

IBM Content Manager OnDemand for Multiplatforms

Windows Client Customization Guide



IBM Content Manager OnDemand for Multiplatforms

Windows Client Customization Guide



Note

Before using this information and the product that it supports, read the information in “Notices” on page 269.

This edition replaces SC27-0837-07.

This edition applies to Version 9 Release 0 of IBM Content Manager OnDemand for z/OS (product number 5697-N93), Version 9 Release 0 of IBM Content Manager OnDemand for Multiplatforms (product number 5724-J33), and Version 7 Release 1 of IBM Content Manager OnDemand for i (product number 5770-RD1) and to all subsequent releases and modifications until otherwise indicated in new editions.

This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).

© **Copyright IBM Corporation 1996, 2012.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this publication vii

Who should use this publication	vii
ibm.com and related resources	vii
Support and assistance	vii
Information Center.	vii
PDF publications	vii
Accessibility information for OnDemand	viii
How to send your comments	viii

Part 1. The OnDemand OLE control 1

Chapter 1. Overview of the Content Manager OnDemand OLE Control. . . . 3

Viewing multiple documents for a single folder. . . .	3
Header File.	3
Return Code	4

Chapter 2. Methods. 5

AboutBox	5
ActivateFolder.	5
AnnotateDoc	6
CancelOperation	7
ChangePassword	8
ClearFolderSearchFields	9
CloseAllFolders	10
CloseDoc	11
CloseFolder	11
CopyBitmap	12
CopyDocPagesToFile	13
CopyText	14
DeleteDoc	15
FindStringInDoc	15
GetAnnotationForDoc	18
GetAnnotationStatus	19
GetControlId	20
GetDocAnnotation	21
GetDocBackgroundColor	23
GetDocCurrentPage	24
GetDocDisplayValue	25
GetDocDisplayValues	26
GetDocImageColor	28
GetDocImageIntensity	29
GetDocNumPages	30
GetDocRotation	31
GetDocScrollPositions	33
GetDocType	34
GetDocZoom	35
GetFolderDisplayFieldName.	36
GetFolderDisplayFieldNames	37
GetFolderFieldName	39
GetFolderFieldNames	40
GetFolderName	41
GetFolderNames.	42
GetFolderSearchFieldName	44

GetFolderSearchFieldNames	45
GetNumDocAnnotations	47
GetNumDocsInList	48
GetNumFolderDisplayFields.	50
GetNumFolderFields	52
GetNumFolders	54
GetNumFolderSearchFields	55
GetNumServerPrinters.	58
GetNumServers	59
GetResourceCacheMode	60
GetServerName	61
GetServerNames.	62
GetServerPrinter.	64
GetServerPrinterInfo	65
GetStoreDocInvalidFieldNum	66
GetTypeForDoc	67
IsDocHorzScrollRequired	68
Logoff	71
Logon	72
OnSysColorChange.	73
OpenDoc	74
OpenFolder	77
PrintDoc	78
RetrieveDoc	80
ScrollDocHorz	82
ScrollDocVert.	85
SearchFolder	88
SetDefaultFolderSearchFields	90
SetDocBackgroundColor	91
SetDocCurrentPage	92
SetDocImageColor	93
SetDocImageIntensity	94
SetDocRotation	95
SetDocZoom	95
SetFolderCloseMemoryRelease	97
SetFolderSearchFieldData.	98
SetLogonReturnOnFailure	100
SetResourceCacheMode	101
SetRightButtonMenu	102
SetSelectionMode	105
SetServerPrinterData	106
SetUserMessageMode	107
ShowFolder	108
ShowWaitCursorDuringCancelableOperation	109
StoreDoc	110
UndoFind	113
UpdateDoc	114
WasOperationCancelled	115

Chapter 3. OLE Events 117

FolderSearchCompleted	117
FolderClosed	117
DocOpened	117
DocClosed	117
AreaSelected.	117

AreaDeselected	117
UserCommand(long CommandID)	118

Part 2. Windows 32-bit GUI customization guide 119

Chapter 4. Content Manager OnDemand customization overview . . 121

Chapter 5. Command line 123

Starting Content Manager OnDemand 32-bit client	123
Parameter Syntax	123
Parameters	123
Change Password — /C new password	123
Code Page Override — /A codepage_number	123
Disable Anticipation — /V	124
Disable Close Folder — /Z	124
Disable Exit — /K	124
Disable Logoff or Password Change — /X	124
Disable Update Servers — /Y	124
Disable User Confirmation — /B	124
Enable DDE Interface — /I number,path,resid	125
Folder Name — /F name	125
Free Memory When Folder Closed — /Q	125
Language ID Override — /L	125
Language Path — /1	125
Logon Password — /P password	126
Logon Server Name — /S name	126
Logon User ID — /U id	126
Maximum Open Folders — /O number	126
Product Title — /T name	126
Window Placement — /W placement	127

Chapter 6. Dynamic Data Exchange (DDE) and DDE Management Library . 129

Invoking the OnDemand 32-bit from another Windows application	129
Content Manager OnDemand invocation and DDEML initialization	130
DDEML termination	132
DDEML transactions	132

Chapter 7. Content Manager OnDemand DDE commands 135

ACTIVATE_DOC	135
ACTIVATE_FOLDER	135
ANNOTATE_DOC	136
ARRANGE_DOCS	137
CHANGE_PASSWORD	138
CLEAR_FIELDS	138
CLOSE_ALL_DOCS	139
CLOSE_ALL_FOLDERS	139
CLOSE_DOC	140
CLOSE_FOLDER	140
COPY_DOC_PAGES	141
DELETE_DOC	142
DESELECT_DOC	143
DISABLE_SWITCH	143
ENABLE_SWITCH	144

EXIT	145
GET_DISPLAY_FIELDS	145
GET_DOC_VALUES	146
GET_FOLDER_FIELDS	147
GET_FOLDERS	148
GET_NUM_DOCS_IN_LIST	148
GET_NUM_DOC_PAGES	149
GET_PRINTERS	149
GET_QUERY_FIELDS	150
GET_SERVERS	151
LOGOFF	151
LOGON	152
OPEN_DOC	153
OPEN_FOLDER	154
PRINT_DOC	157
RESTORE_DEFAULTS	159
RETRIEVE_DOC	159
SEARCH_FOLDER	160
SELECT_DOC	161
SET_FIELD_DATA	162
SET_FOCUS	163
SET_HELP_PATH	164
SET_USER_MSG_MODE	165
SHOW_WINDOW	165
STORE_DOC	166
UPDATE_DOC	168

Chapter 8. Return Codes 171

Chapter 9. DDEML Advise Loop . . . 173

Chapter 10. External Applications and Dynamic Link Libraries 175

Chapter 11. Related documents . . . 183

Chapter 12. Program Information File 187

Chapter 13. Document Audit Facility 189

Overview	189
Creating the DAF control file	189
The AUDIT section	189
The folder section	190
Defining the report	190
Defining the application group	190
Defining the application	191
Defining the Folder	191
Controlling access to the DAF	191
Using the DAF	192

Chapter 14. Modifying client behavior through the Registry 193

Chapter 15. Integration with Monarch Version 5 195

Before you begin	196
Configuring the client	196
Adding the Registry key	196

Exporting the Registry key	201
Using multiple Monarch model files.	201
Configuring Setup.	202
Copying client software	203
Adding subdirectories	203
Copying Monarch files	203
Sharing the installation directory	203
Running Setup	204
Running Monarch from OnDemand.	204
Upgrading your OnDemand client	204

Chapter 16. Installing client software on a network 205

Sharing OnDemand clients among multiple users	205
Installation directories	205
Distribution install	206
Overview.	206
Copying OnDemand software to the server	207
Distributing user-defined files.	207
Configuring the client to connect to a z/OS Version 2.1 server.	207
Step 1: Recommended: Specifying settings in the setup.ini file.	208
Step 2: Required: Adding the required command line parameters.	208
Step 3: Required for OLE: Using the SetFolderCloseMemoryRelease method.	209
Step 4: Recommended: Configuring the client to run in short sockets mode	209
Step 5: Clearing the Attempt Unified Logon check box in the Update Servers window to access a z/OS server	209

Chapter 17. Distributing user-defined files 211

Overview.	211
Copying OnDemand client software to the server	212
Adding subdirectories	212
Storing user-defined files on the server.	213
Installing the OnDemand client	213

Chapter 18. Using response files . . . 215

Introduction.	215
-----------------------	-----

Format of a response file	215
Creating a response file	215
Installing software using a response file	216
Verifying software installation.	216
Using a response file to install OnDemand software	216

| Chapter 19. Full report browse and reprint 217

| Chapter 20. ICU converters 219

Chapter 21. Troubleshooting 221

StoreDoc() API returns error code 2	221
Hints and tips	221
How to allow a user to print AFP files without opening them with the workbench or navigation plug-in	221

Appendix A. Microsoft Visual Basic DDE sample program 223

Global variables used by the sample program	223
Entry point for the sample program	223

Appendix B. Microsoft Visual C++ DDE sample program 237

Global variables used by the sample program	237
---	-----

Appendix C. Microsoft Visual Basic OLE sample program 247

Global variables used by the sample program	247
---	-----

Appendix D. Microsoft Visual C++ OLE sample program 257

Notices 269

Trademarks	271
----------------------	-----

Index 273

About this publication

This book contains information about the IBM® Content Manager OnDemand (Content Manager OnDemand) Object Linking and Embedding (OLE) Control and how to customize the Content Manager OnDemand Windows client by specifying command line parameters, by invoking and manipulating the Content Manager OnDemand Windows client from another Windows application with the Dynamic Data Exchange (DDE) interface, or by creating a Program Information File (PIF). This book also contains information that administrators can use to distribute the OnDemand Windows client software and files to multiple users over a network.

Note: The term Content Manager OnDemand Windows client (or simply, *client*) refers to the Content Manager OnDemand end-user client software that runs under Microsoft Windows. The term *server* refers to systems that are running the Content Manager OnDemand for i, Content Manager OnDemand for Multiplatforms, and Content Manager OnDemand for z/OS.

Who should use this publication

This book is of primary interest to programmers who want to integrate Content Manager OnDemand with other Windows applications and administrators that are responsible for installing and distributing software products.

ibm.com and related resources

Product support and documentation are available from ibm.com®.

Support and assistance

Product support is available on the Web. Click Support from the product Web site at:

Content Manager OnDemand for Multiplatforms

www.ibm.com/software/data/ondemand/mp/support.html

Content Manager OnDemand for z/OS®

www.ibm.com/software/data/ondemand/390/support.html

Content Manager OnDemand for IBM i

www.ibm.com/software/data/ondemand/400/support.html

Information Center

You can view the product documentation in an Eclipse-based information center that you can install when you install the product. By default, the information center runs in a Web server mode that other Web browsers can access. You can also run it locally on your workstation. See the information center at: www.ibm.com/software/data/ondemand/mp/support.html

PDF publications

You can view the PDF files online using the Adobe Acrobat Reader for your operating system. If you do not have Acrobat Reader installed, you can download it from the Adobe Web site at www.adobe.com.

You can find PDF publications at: <http://www.ibm.com/support/docview.wss?rs=129&uid=swg27017159>

Accessibility information for OnDemand

For complete information about accessibility features that are supported by this product, see your *Administration Guide*.

How to send your comments

Your feedback helps IBM to provide quality information. Please send any comments that you have about this publication or other OnDemand documentation. Visit the IBM Data Management Online Reader's Comment Form (RCF) page at www.ibm.com/software/data/rcf.

If you would like to help IBM make Content Manager OnDemand easier to use, take the consumability survey at <http://www.ibm.com/software/data/info/consumability-survey/>.

Be sure to include the name of the product, the version number of the product, and the name of the book. If you are commenting on specific text, please include the location of the text (for example, a chapter and section title, a table number, a page number, or a help topic title).

Part 1. The OnDemand OLE control

Chapter 1. Overview of the Content Manager OnDemand OLE Control

Note: In order to use this part of the book, you should know how to imbed OLE controls in an application.

Content Manager OnDemand makes an OLE (Object Linking and Embedding) Control available for displaying documents from the Content Manager OnDemand database. The OLE Control is implemented in ARSOLE.OCX. During Content Manager OnDemand installation, this file is placed in the same directory as the other Content Manager OnDemand executable files, and the OLE Control is registered with the Windows system. To run your container application from any directory other than where Content Manager OnDemand was installed, you need to add the Content Manager OnDemand directory to your path.

The following rules apply to the use of these controls:

- Each control can display only one document at a time. A document must be closed before another can be displayed.
- Scroll bars to control scrolling of the document data are the responsibility of the container application. These must appear outside the OLE Control window. The OLE Control provides methods to direct the scrolling of the document data. Use of these methods is made easier if the scroll bar ranges are set to ARS_OLE_SCROLL_RANGE.
- Multiple folders may be open simultaneously, but only one of these will be the active folder. The OLE Control provides methods to open, close, and activate a folder.
- The container application can completely control logon, open folder, search folder, close folder, and open document operations or it can cause the normal Content Manager OnDemand dialog boxes to be used for these operations.

Viewing multiple documents for a single folder

Each Content Manager OnDemand OLE Control has a unique runtime control id. This control id can be retrieved with the GetControlId method.

Control ids allow multiple Content Manager OnDemand OLE Controls to simultaneously display documents from a single folder document list. This avoids the overhead of multiple logon, open folder, and search folder operations.

A given application can include more than one Content Manager OnDemand OLE Control. That application could use one of those controls to logon, open a folder, and search the folder to create a list of documents. If the control id for that control is made available, the other controls could reference it when using the OpenDoc method and display documents from the single document list.

Header File

The ARSOLEEX.H header file contains definitions of symbolic values used in the OLE control methods described below. It can be included in C/C++ implementations or used as a reference for other languages.

The header file is installed into the INC subdirectory of the Content Manager OnDemand installation directory. That directory can be added to the include file path or the file can be copied to another directory.

Return Code

Most Content Manager OnDemand OLE Control methods return a short value. A list of the return code values, such as `ARS_OLE_RC_SUCCESS`, can be found in `ARSOLEEX.H`.

Chapter 2. Methods

The following sections describe the methods available for an Content Manager OnDemand OLE Control.

AboutBox

Method:

void **AboutBox()**

Description:

The Content Manager OnDemand About Box is displayed.

Return Value:

None.

Example.

The following example displays the Content Manager OnDemand About Box.

C/C + + Example

```
CArsOle * pArsCtrl;  
  
.  
.  
  
pArsCtrl->AboutBox( );  
  
.  
.
```

Visual Basic Example

```
.  
.  
.  
  
ArsOle.AboutBox  
  
.  
.  
.
```

ActivateFolder

Method:

short ActivateFolder(
 char * **pFolderName**)

Parameters:

pFolderName

Points to a null-terminated character string containing the name of the folder to be activated.

Description:

The named folder becomes the active folder.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

OpenFolder, CloseFolder, CloseAllFolders

Example

The following example activates a folder.

C/C + + Example

```
CArsOle * pArsCtrl;
short rc;

.
.

rc = pArsCtrl->ActivateFolder( "Henry's Folder" );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

.
.
```

Visual Basic Example

```
Dim rc As Integer

.
.
.

rc = ArsOle.ActivateFolder("Henry's Folder")
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

.
.
.
```

AnnotateDoc

Method:

```
short AnnotateDoc(
    long Index,
    char * pText,
    long page,
    boolean Public,
    boolean CanBeCopied )
```

Parameters:**Index**

Specifies the zero-based index of a document within the document list of the active folder. If this value is less than zero, the annotation is associated with the open document.

pText

Points to a null-terminated character string containing the text of the annotation. If the text contains more than 32,700 characters, it is truncated.

page

Specifies the page number to be associated with the annotation.

Public

Indicates whether the annotation is public.

CanBeCopied

Indicates whether the annotation may be copied to other servers.

Description:

An annotation is created in the database and associated with the specified document.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

OpenDoc

Example

The following example creates an annotation for a document.

C/C + + Example

```
CArsOle * pArsCtrl;
short rc;

.
.

rc = pArsCtrl->AnnotateDoc( 3, "This is the text.", 5, TRUE, FALSE );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

.
.
```

Visual Basic Example

```
Dim rc As Integer

.
.
.

rc = ArsOle.AnnotateDoc(3, "This is the text.", 5, True, False)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

.
.
.
```

CancelOperation

Method:

short CancelOperation()

Description:

Cancels an operation that was started by a SearchFolder, OpenDoc, or RetrieveDoc method.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

SearchFolder, OpenDoc, WasOperationCancelled,
ShowWaitCursorDuringCancelableOperation

Example:

The following example cancels an operation.

C/C++ Example

```
CArsOle * pArsCtrl;
short rc;
.
.
rc = pArsCtrl->CancelOperation( );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.
```

Visual Basic Example

```
Dim rc As Integer

.
.

rc = ArsOle.CancelOperation ( )
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

.
.
```

ChangePassword

Method:

```
short ChangePassword(
    char * pCurrentPassword,
    char * pNewPassword1,
    char * pNewPassword2 )
```

Parameters:

pCurrentPassword

Specifies the users current password.

pNewPassword1

Specifies the users new password.

pNewPassword2

Specifies the users new password again. This is for verification.

Description:

OnDemand changes the logon password for the current user.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

Lgon

Example:

C/C++ Example

```
CArsOle * pArsCtrl;  
short rc;  
  
.  
.  
  
rc = pArsCtrl->ChangePassword( "tt1sd",  
                                "sfd45r",  
                                "sfd45r" );  
  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
.  
.
```

Visual Basic Example

```
Dim rc As Integer  
  
.  
.  
  
rc = ArsOle.ChangePassword ( "tt1sd", _  
                             "sfd45r", _  
                             "sfd45r" ) _  
  
if rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If  
  
.  
.
```

ClearFolderSearchFields

Method:

short ClearFolderSearchFields()

Description:

The search fields for the active folder are cleared.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

OpenFolder, SearchFolder

Example:

The following example clears the search fields for the active folder.

C/C + + Example

```
CArsOle * pArsCtrl;  
short rc;  
  
.  
.  
  
rc = pArsCtrl->ClearFolderSearchFields( );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
.  
.
```

Visual Basic Example

```

Dim rc As Integer

.
.
.

rc = Ars01e.ClearFolderSearchFields()
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

.
.
.

```

CloseAllFolders

Method:

short CloseAllFolders()

Description:

All open folders are closed. This causes all open documents associated with the folders to be closed.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

OpenFolder, CloseFolder

Example.

The following example closes all folders.

C/C++ Example

```

CArs01e * pArsCtrl;
short rc;

.
.

rc = pArsCtrl->CloseAllFolders( );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

.
.

```

Visual Basic Example

```

Dim rc As Integer

.
.
.

rc = Ars01e.CloseAllFolders()
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

.
.
.

```

CloseDoc

Method:

short CloseDoc()

Description:

The open document is closed and the control window is repainted with a white background.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

OpenDoc

Example:

The following example closes a document.

C/C + + Example

```
CArsOle * pArsCtrl;  
short rc;  
  
.  
.  
  
rc = pArsCtrl->CloseDoc( );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
.  
.
```

Visual Basic Example

```
Dim rc As Integer  
  
.  
.  
.  
  
rc = ArsOle.CloseDoc()  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If  
  
.  
.  
.
```

CloseFolder

Method:

short CloseFolder()

Description:

The active folder is closed. This causes all open documents associated with the folder to be closed. If any other folders are open, one of them becomes the active folder. If more than one other folder is open, the container application should invoke the ActivateFolder method to specify the folder which is to be active.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

OpenFolder, CloseAllFolders

Example:

The following example closes the active folder.

C/C++ Example

```
CArsOle * pArsCtrl;
short rc;

.
.

rc = pArsCtrl->CloseFolder( );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

.
.
```

Visual Basic Example

```
Dim rc As Integer

.
.

rc = ArsOle.CloseFolder()
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

.
.
```

CopyBitmap

Method:

short CopyBitmap()

Description:

Copies a selected area of the document to the clipboard in bitmap format.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

CopyText, SetSelectionMode

Example:

The following example copies a selected area of the document to the clipboard in bitmap format.

C/C++ Example

```
CArsOle * pArsCtrl;
short rc;

.
.

rc = pArsCtrl->CopyBitmap( );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

.
.
```

Visual Basic Example

```

Dim rc As Integer

.
.

rc = ArsOle.CopyBitmap ( )
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

.
.

```

CopyDocPagesToFile

Method:

```

short CopyDocPagesToFile(
    char * pPath,
    long page,
    boolean AsIs )

```

Parameters:

page

Specifies the page number to be copied. If this parameter is less than or equal to zero, the entire document is copied.

pPath

Points to a null-terminated character string containing the fully qualified name of a file to which the data is to be copied. If the file already exists, the data is appended to the file.

AsIs (for AFP and line data only)

If non-zero, indicates that the data is to be copied "asis"; if zero, that it is to be converted to ASCII.

Description:

The data for the page or pages of the open document is copied to the specified file.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

SearchFolder, GetNumDocsInList

Example:

The following example copies page 5 of the open document to a file in ASCII format.

C/C++ Example

```

CArsOle * pArsCtrl;
short rc;

.
.

rc = pArsCtrl->CopyDocPagesToFile( "C:\\FILES\\MYDATA.FIL", 5, FALSE );
if ( rc != ARS_OLE_RC_SUCCESS )

```

```
ERROR;
```

```
.  
.
```

Visual Basic Example

```
Dim rc As Integer
```

```
.  
.
```

```
rc = ArsOle.CopyDocPagesToFile("C:\FILES\MYDATA.FIL", 5, False)  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If
```

```
.  
.
```

CopyText

Method:

```
short CopyText( )
```

Description:

Copies a selected area of the document to the clipboard in text format.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

CopyBitmap, SetSelectionMode

Example:

The following example copies a selected area of the document to the clipboard in text format.

C/C + + Example

```
CArsOle * pArsCtrl;  
short rc;  
. .  
rc = pArsCtrl->CopyText( );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
. .
```

Visual Basic Example

```
Dim rc As Integer
```

```
.  
.
```

```
rc = ArsOle.CopyText ( )  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If
```

```
.  
.
```

DeleteDoc

Method:

```
short DeleteDoc(  
    long DocIndex )
```

Parameters:

DocIndex

Specifies the zero-based relative document number within the document list of the active folder.

Description:

OnDemand deletes the specified document from the database. Since the document numbers may have changed, information from a previous GetDocDisplayValues method may no longer be valid.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

GetNumDocsInList

Example:

The following example deletes the first document in the document list of the active folder.

C/C++ Example

```
CArsOle * pArsCtrl;  
short rc;  
  
.  
.  
  
rc = pArsCtrl->DeleteDoc( 0 );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
.  
.
```

Visual Basic Example

```
Dim rc As Integer  
  
.  
.  
  
rc = ArsOle.DeleteDoc (0)  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If  
  
.  
.
```

FindStringInDoc

Method:

```
short FindStringInDoc(  
    char * pString,  
    long page,
```

short **Type**,
 boolean **CaseSensitive**,
 VARIANT * **pFound**,
 VARIANT * **pHorzPosition**,
 VARIANT * **pVertPosition**)

Parameters:

pString

Points to a null-terminated character string containing the text to be found.

page

Specifies the page on which the search is to begin. If **Type** specifies **ARS_OLE_FIND_PREV** or **ARS_OLE_FIND_NEXT**, the page must be the same as that on which a current find is highlighted.

Type

Specifies the type of find operation. This must be one of the following type values found in **ARSOLEEX.H**:

ARS_OLE_FIND_FIRST
ARS_OLE_FIND_PREV
ARS_OLE_FIND_NEXT

CaseSensitive

If non-zero, indicates that the search should be case sensitive; if zero, that the case should be ignored.

pFound

Points to a variable to receive a found/not found indication. On return, this variable is set to type **VT_I2**.

pHorzPosition

Points to a variable to receive the new horizontal scroll position. On return, this variable is set to type **VT_I2**.

pVertPosition

Points to a variable to receive the new vertical scroll position. On return, this variable is set to type **VT_I2**.

Description:

A search is conducted for the text string beginning on the specified page. The variable pointed to by **pFound** is set to non-zero if the search succeeds; zero if it fails. If the search is successful, the page on which the string is found is made the current page, the string is highlighted and scrolled into view, and the new scroll positions are returned in the specified variables. The scroll positions assume that the scroll ranges have been set to **ARS_OLE_SCROLL_RANGE**.

The search will always “wrap” the document from end to beginning or beginning to end. A previous or next find will never fail. If there is a single occurrence in the document, these will find the same string.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

OpenDoc, UndoFind

Example:

The following example performs a search.

C/C + + Example

```
CArsOle * pArsCtrl;
CScrollBar * pHorzScrollBar, * pVertScrollBar;
VARIANT found, horz_position, vert_position;
char * pString;
short rc;

.
.
.

rc = pArsCtrl->FindStringInDoc( pString,
                                1,
                                ARS_OLE_FIND_FIRST,
                                FALSE,
                                &found,
                                &horz_position,
                                &vert_position );

if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

if ( found.iVal )
{
    pHorzScrollBar->SetScrollPos( (int)horz_position.iVal );
    pVertScrollBar->SetScrollPos( (int)vert_position.iVal );

    .
    .
}
else
{
    .
    .
}

.
.
```

Visual Basic Example

```
Dim rc As Integer
Dim found, horz_pos, vert_pos As Variant
Dim Temp As String

.
>
.
rc = ArsOle.FindStringInDoc( Temp,
                              1,
                              ARS_OLE_FIND_FIRST,
                              False,
                              found,
                              horz_pos,
                              vert_pos )

If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

If found <> 0 Then
    hScrollBar.Value = horz_pos
    vScrollBar.Value = vert_pos
```

End If

.
.

GetAnnotationForDoc

Note: This method intended for use with Visual Basic.

Method:

```
short GetAnnotationForDoc(  
    short Index,  
    BSTR * pText,  
    BSTR * pUserId,  
    BSTR * pDateTime,  
    VARIANT * pPage,  
    VARIANT * pPublic,  
    VARIANT * pCanBeCopied )
```

Parameters:

Index

Specifies the zero-based index of the annotation to be returned. It must be a number greater than or equal to zero and less than the value returned by GetNumDocAnnotations.

pText

Points to a BSTR to receive the text of the annotation.

pUserId

Points to a BSTR to receive the userid for the annotation.

pDateTime

Points to a BSTR to receive the date and time for the annotation.

pPage

Points to a variable to receive the document page number for the annotation. On return, this variable is set to type VT_I4.

pPublic

Points to a variable to receive a Boolean flag indicating whether the annotation is public or private. On return, this variable is set to type VT_I2.

pCanBeCopied

Points to a variable to receive a Boolean flag indicating whether the annotation can be copied to another server. On return, this variable is set to type VT_I2.

Description:

The annotation is retrieved.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

GetNumDocAnnotations, GetDocAnnotation

Example

The following example retrieves an annotation for a document.

```
Dim rc, j As Integer
Dim num_notes, page, ispublic, canbecopied As Variant
Dim text As String
Dim userid As String
Dim datetime As String

rc = ArsOle.GetNumDocAnnotations( num_notes )
if rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

For j = 0 To num_notes -1
    rc = ArsOle.GetAnnotationForDoc( j,
                                     text,
                                     userid,
                                     datetime,
                                     page,
                                     ispublic,
                                     canbecopied )

    if rc <> ARS_OLE_RC_SUCCESS Then
        MsgBox "ERROR"
    End
End If
Next j
```

GetAnnotationStatus

Method:

```
short GetAnnotationStatus(
    long Index,
    VARIANT * pStatus )
```

Parameters:

Index

Specifies the zero-based index of a document within the document list of the active folder. If this value is less than zero, status is returned for the open document.

pStatus

Points to a variable to receive the annotation status. This will be one of the annotation status values, such as ARS_OLE_ANNOTATION_YES, found in ARSOLEEX.H. On return, this variable is set to type VT_I2.

Description:

The annotation status is returned in the specified variable.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

AnnotateDoc, GetNumDocAnnotations, GetDocAnnotation, GetAnnotationForDoc

Example

The following example gets the annotation status for a document.

C/C + + Example

```

VARIANT status;
CArsOle * pArsCtrl;
short rc;

.
.
.

rc = pArsCtrl->GetAnnotationStatus( -1, &status );

if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

.
.
.

```

Visual Basic Example

```

Dim rc As Integer
Dim status As Variant

.
.
.

rc = ArsOle.GetAnnotationStatus( -1, status )
if rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

.
.
.

```

GetControlId

Method:

```

short GetControlId(
    VARIANT * pControlId )

```

Parameters:

pControlId

Points to a variable to receive the control id. On return, this variable is set to type VT_I4.

Description:

The identifier of the control is returned in the specified variable. This control identifier can be used to reference information associated with a different Content Manager OnDemand OLE Control. Refer to “Viewing multiple documents for a single folder” on page 3 for a discussion of control ids.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

OpenDoc

Example:

The following example retrieves the control id.

C/C + + Example

```

Long ControlId;

.
.

CArsOle * pArsCtrl;
VARIANT control_id;
short rc;

.
.

rc = pArsCtrl->GetControlId( &control_id );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

ControlId = control_id.lVal;

.
.

```

Visual Basic Example

```

Dim rc As Integer
Dim control_id As Variant

.
.

rc = ArsOle.GetControlId (control_id)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

.
.

```

GetDocAnnotation

Note: This method intended for use with C/C++.

Method:

```

short GetDocAnnotation(
    short Index,
    LPUNKNOWN pText,
    LPUNKNOWN pUserId,
    LPUNKNOWN pDateTime,
    VARIANT * pPage,
    VARIANT * pPublic,
    VARIANT * pCanBeCopied )

```

Parameters:

Index

Specifies the zero-based index of the annotation to be returned. It must be a number greater than or equal to zero and less than the value returned by GetNumDocAnnotations.

pText

Points to a string to receive the text of the annotation.

pUserId

Points to a string to receive the userid for the annotation.

pDateTime

Points to a string to receive the date and time for the annotation.

pPage

Points to a variable to receive the document page number for the annotation. On return, this variable is set to type VT_I4.

pPublic

Points to a variable to receive a Boolean flag indicating whether the annotation is public or private. On return, this variable is set to type VT_I2.

pCanBeCopied

Points to a variable to receive a Boolean flag indicating whether the annotation can be copied to another server. On return, this variable is set to type VT_I2.

Description:

The annotation data is retrieved.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

GetNumDocAnnotations, GetAnnotationForDoc

Example

The following example retrieves an annotation for a document.

```
VARIANT num_notes, page, ispublic, canbecopied;
CArsOle * pArsCtrl;
short rc, j;
char * pText, userid[100], datetime[200];

rc = pArsCtrl->GetNumDocAnnotations( &num_notes );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

pText = new char[35000];

for ( j = 0; j < num_notes.iVal; j++ )
{
    rc = pArsCtrl->GetDocAnnotation( j,
                                    (LPUNKNOWN)pText,
                                    (LPUNKNOWN)userid,
                                    (LPUNKNOWN)datetime,
                                    &page,
                                    &ispublic,
                                    &canbecopied );

    if ( rc != ARS_OLE_RC_SUCCESS )
        ERROR;

    // Process annotation
}

delete pText;
```

GetDocBackgroundColor

Method:

```
short GetDocBackgroundColor(  
    VARIANT * pColor,  
    VARIANT * pChangeable )
```

Parameters:

pColor

Points to a variable to receive the current document background color. This will be one of the color values, such as ARS_OLE_COLOR_WHITE, found in ARSOLEEX.H. On return, this variable is set to type VT_I2.

pChangeable

Points to a variable to receive an indication of whether the document background color can be changed. On return, this variable contains a non-zero value if the color is changeable; otherwise, zero. On return, this variable is set to type VT_I2.

Description:

The current document background color and a changeability indicator are returned.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

SetDocBackgroundColor

Example:

The following example retrieves the current document background color and disables a menu item if the color cannot be changed.

C/C++ Example

```
CArsOle * pArsCtrl;  
CMenu * pSubMenu;  
short rc, back_color;  
VARIANT current_color, changeable;  
  
:  
:  
  
rc = pArsCtrl->GetDocBackgroundColor( &current_color, &changeable );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
back_color = current_color.iVal;  
  
pSubMenu->EnableMenuItem(  
    ID_VIEW_BKGD_COLOR,  
    MF_BYCOMMAND | ( changeable.iVal ? MF_ENABLED : MF_GRAYED ) );  
  
:  
:
```

Visual Basic Example

```
Dim rc As Integer  
Dim back_color, changeable As Variant  
  
:  
:  
  
rc = ArsOle.GetDocBackgroundColor (back_color, changeable)
```

```

If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

If changeable <> 0 Then
    menuBackgroundColor.Enabled = True
Else
    menuBackgroundColor.Enabled = False
End If

.
.

```

GetDocCurrentPage

Method:

```

short GetDocCurrentPage(
    VARIANT * pPage )

```

Parameters:

pPage

Points to a variable to receive the current page number of the open document. On return, this variable is set to type VT_I4.

Description:

The current page number of the open document is returned in the specified variable.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

SetDocCurrentPage, GetDocNumPages

Example:

The following example retrieves the current page number of the open document.

C/C++ Example

```

CArsOle * pArsCtrl;
VARIANT vari;
long page_num;
short rc;

.
.

rc = pArsCtrl->GetDocCurrentPage( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

page_num = var.lVal;

.
.

```

Visual Basic Example

```

Dim rc As Integer
Dim page_num As Variant

.
.

```

```

rc = ArsOle.GetDocCurrentPage (page_num)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

.
.

```

GetDocDisplayValue

Note: This method intended for use with Visual Basic.

Method:

```

short GetDocDisplayValue(
    long DocIndex,
    short ValueIndex,
    BSTR * pValue )

```

Parameters:

DocIndex

Specifies the zero-based index of a document within the document list of the active folder.

ValueIndex

Specifies the zero-based index of the value to be returned. It must be a number greater than or equal to zero and less than the value returned by GetNumFolderDisplayFields.

pValue

Points to a BSTR to receive the value.

Description:

The specified value is returned in pValue.

GetDocDisplayValue or GetDocDisplayValues can be used to retrieve the document display values. An application should use the one which is more convenient in its environment.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

GetNumDocsInList, GetNumFolderDisplayFields,
GetFolderDisplayFieldNames, GetDocDisplayValues, OpenDoc

Example:

The following example creates a list box of the folder document list names and associated values and opens the document selected by a user.

```

Dim rc, count, i, j As Integer
Dim num_fields, num_docs As Variant
Dim Names() As String
Dim Line As String
Dim Temp As String

.
.
.
rc = ArsOle.GetNumFolderDisplayFields(num_fields)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"

```

```

        End
    End If

    ReDim Names(num_fields)

    For count = 0 To num_fields -1
        rc = ArsOle.GetFolderDisplayFieldName(count, Temp)
        Names(count) = Temp
    Next count

    rc = ArsOle.GetNumDocsInList(num_docs)
    If rc <> ARS_OLE_RC_SUCCESS Then
        MsgBox "ERROR"
        End
    End If

    For j = 0 To num_docs -1
        For i = 0 To num_fields -1
            rc = ArsOle.GetDocDisplayValue(j, i, Temp)

            Line = Line + Names(i) + " = " + Temp
            If i < num_fields Then
                Line = Line + ", "
            End If
        Next i

        lbDocs.AddItem Line
    Next j

    .
    .
    .
    'During OK button processing:

    rc = ArsOle.OpenDoc (lbDocs.List(lbDocs.ListIndex), "", 0)
    If rc <> ARS_OLE_RC_SUCCESS Then
        MsgBox "ERROR"
        End
    End If

```

GetDocDisplayValues

Note: This method intended for use with C/C++.

Method:

```

short GetDocDisplayValues(
    long Index,
    IUnknown * pValues,
    short MaxValues )

```

Parameters:

Index

Specifies the zero-based index of a document within the document list of the active folder.

pValues

Points to an array of ArsOleValues to receive the values of the folder display fields for the document specified with Index. There are the same number of values as of display fields. The array must have at least MaxValues elements.

MaxValues

Specifies the maximum number of values to be returned.

Description:

The values of the folder display fields for the document, up to a maximum of MaxValues, are returned in pValues. Each name is a null-terminated character string.

The values are placed in the array in the same sequence that the display field names are returned by the GetFolderDisplayFieldNames method.

GetDocDisplayValue or GetDocDisplayValues can be used to retrieve the document display values. An application should use the one which is more convenient in its environment.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

GetNumDocsInList, GetNumFolderDisplayFields,
GetFolderDisplayFieldNames, GetDocDisplayValue, OpenDoc

Example:

The following example creates a list box of the folder document list names and associated values and opens the document selected by a user.

C/C++ example

```
CArsOle * pArsCtrl;  
ArsOleName * pNames;  
ArsOleValue * pValues;  
CListBox * pDocList;  
char * pLine;  
short rc, k, opr, num_fields;  
long j, num_docs;  
int size;  
VARIANT vari;  
.  
.  
// During dialog initialization:  
  
rc = pArsCtrl->GetNumFolderDisplayFields( &vari );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
num_fields = var.iVal;  
  
pNames = new ArsOleName[ max( num_fields, 1 ) ];  
rc = pArsCtrl->GetFolderDisplayFieldNames( (IUnknown*)pNames, num_fields );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
rc = pArsCtrl->GetNumDocsInList( &vari );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
num_docs = var.lVal;  
  
pValues = new ArsOleValue[ max( num_fields, 1 ) ];  
  
size = num_fields * ( sizeof(ArsOleName) + sizeof(ArsOleValue) + 5 );  
pLine = new char[ size ];  
for ( j = 0, pLine[0] = '\\0'; j < num_docs; j++ )  
{  
    rc = pArsCtrl->GetDocDisplayValues( j, pValues, num_fields );  
    if ( rc != ARS_OLE_RC_SUCCESS )  
        ERROR;  
    .  
    .  
}
```

```

        .
        .
        for ( k = 0; k < num_fields; k++ )
        {
            strcat( pLine, pNames[k] );
            strcat( pLine, " = " );
            strcat( pLine, pValues[k] );
            if ( k < num_fields - 1 )
                strcat( pLine, ", " );
        }
        pDocList->InsertString( -1, pLine );
    }
    pDocList->SetCurSel( 0 );
    .
    .
    // During OK button processing:

    rc = pArsCtrl->OpenDoc( (long)pDocList->GetCurSel( ) , NULL, 0 );
    if ( rc != ARS_OLE_RC_SUCCESS )
        ERROR;

    .
    .

```

GetDocImageColor

Method:

```

short GetDocImageColor(
    VARIANT * pColor,
    VARIANT * pChangeable )

```

Parameters:

pColor

Points to a variable to receive the current document image color. This will be one of the color values, such as ARS_OLE_COLOR_BLACK, found in ARSOLEEX.H. On return, this variable is set to type VT_I2.

pChangeable

Points to a variable to receive an indication of whether the document image color can be changed. On return, this variable contains a non-zero value if the color is changeable; otherwise, zero. On return, this variable is set to type VT_I2.

Description:

The current document image color and a changeability indicator are returned.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

SetDocImageColor

Example:

The following example retrieves the current document image color and disables a menu item if the color cannot be changed.

C/C++ Example

```

CArsOle * pArsCtrl;
CMenu * pSubMenu;
short rc, image_color;
VARIANT current_color, changeable;

```

```

        .
        .

rc = pArsCtrl->GetDocImageColor( &current_color, &changeable );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

image_color = current_color.iVal;

pSubMenu->EnableMenuItem(
    ID_VIEW_IMAGE_COLOR,
    MF_BYCOMMAND | ( changeable.iVal ? MF_ENABLED : MF_GRAYED ) );
        .
        .

```

Visual Basic Example

```

Dim rc As Integer
Dim current_color, changeable As Variant

        .
        .

rc = ArsOle.GetDocImageColor (current_color, changeable)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

If changeable <> 0 Then
    menuImageColor.Enabled = True
Else
    menuImageColor.Enabled = False
End If

        .
        .

```

GetDocImageIntensity

Method:

```

short GetDocImageIntensity(
    VARIANT * pIntensity,
    VARIANT * pChangeable )

```

Parameters:

pIntensity

Points to a variable to receive the current document image intensity. This will be one of the intensity values, such as ARS_OLE_INTENSITY_NORMAL, found in ARSOLEEX.H. On return, this variable is set to type VT_I2.

pChangeable

Points to a variable to receive an indication of whether the document image intensity can be changed. On return, this variable contains a non-zero value if the intensity is changeable; otherwise, zero. On return, this variable is set to type VT_I2.

Description:

The current document image intensity and a changeability indicator are returned.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

SetDocImageIntensity

Example:

The following example retrieves the current document image intensity and disables a menu item if the intensity cannot be changed.

C/C++ Example

```
CArsOle * pArsCtrl;
CMenu * pSubMenu;
short rc, image_intensity;
VARIANT current_intensity, changeable;

:

rc = pArsCtrl->GetDocImageIntensity( &current_intensity, &changeable );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

image_intensity = current_intensity.iVal;

pSubMenu->EnableMenuItem(
    ID_VIEW_IMAGE_INTENSITY,
    MF_BYCOMMAND | ( changeable.iVal ? MF_ENABLED : MF_GRAYED ) );

:
```

Visual Basic Example

```
Dim rc As Integer
Dim current_intensity, changeable As Variant

:

rc = ArsOle.GetDocImageIntensity (current_intensity, changeable)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

If changeable <> 0 Then
    menuImageIntensity.Enabled = True
Else
    menuImageIntensity.Enabled = False
End If

:
```

GetDocNumPages

Method:

```
short GetDocNumPages(
    VARIANT * pNumPages )
```

Parameters:

pNumPages

Points to a variable to receive the number of pages in the open document. On return, this variable is set to type VT_I4.

Description:

The number of pages in the open document is returned in the specified variable.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

OpenDoc, GetDocCurrentPage, SetDocCurrentPage

Example:

The following example retrieves the number of pages in the open document.

C/C++ Example

```
CArsOle * pArsCtrl;
VARIANT vari;
long num_pages;
short rc;

:

rc = pArsCtrl->GetDocNumPages( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

num_pages = var.lVal;

:
```

Visual Basic Example

```
Dim rc As Integer
Dim num_pages As Variant

:

rc = ArsOle.GetDocNumPages (num_pages)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

:
```

GetDocRotation

Method:

```
short GetDocRotation(
    VARIANT * pRotation,
    VARIANT * pChangeable )
```

Parameters:**pRotation**

Points to a variable to receive the current document rotation. This will be one of the rotation values, such as ARS_OLE_ROTATION_0, found in ARSOLEEX.H. On return, this variable is set to type VT_I2.

pChangeable

Points to a variable to receive an indication of whether the document rotation can be changed. On return, this variable contains a non-zero value if the rotation is changeable; otherwise, zero. On return, this variable is set to type VT_I2.

Description:

The current document rotation and a changeability indicator are returned.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

SetDocRotation

Example:

The following example retrieves the current document rotation and disables a menu item if the rotation cannot be changed.

C/C++ Example

```
CArsOle * pArsCtrl;
CMenu * pSubMenu;
short rc, rotation;
VARIANT current_rotation, changeable;

:
:

rc = pArsCtrl->GetDocRotation( &current_rotation, &changeable );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

rotation = current_rotation.iVal;

pSubMenu->EnableMenuItem(
    ID_VIEW_ROTATION,
    MF_BYCOMMAND | ( changeable.iVal ? MF_ENABLED : MF_GRAYED ) );

:
:
```

Visual Basic Example

```
Dim rc As Integer
Dim rotation, changeable As Variant

:
:

rc = ArsOle.GetDocRotation (rotation, changeable)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

If changeable <> 0 Then
    menuRotation.Enabled = True
Else
    menuRotation.Enabled = False
End If

:
:
```

GetDocScrollPositions

Method:

```
short GetDocScrollPositions(  
    VARIANT * pHorzPosition,  
    VARIANT * pVertPosition )
```

Parameters:

pHorzPosition

Points to a variable to receive the new horizontal scroll position. On return, this variable is set to type VT_I2.

pVertPosition

Points to a variable to receive the new vertical scroll position. On return, this variable is set to type VT_I2.

Description:

The current scroll positions are returned in the specified variables. The scroll positions assume that the scroll ranges have been set to ARS_OLE_SCROLL_RANGE.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

Example:

The following example sets the current page number of the open document and updates the current scroll positions.

C/C + + Example

```
CArsOle * pArsCtrl;  
CScrollBar * pHorzScrollBar, * pVertScrollBar;  
short rc;  
VARIANT horz_position, vert_position;  
.  
.  
rc = pArsCtrl->SetDocCurrentPage( 46 );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
.  
.  
rc = pArsCtrl->GetDocScrollPositions( &horz_position, &vert_position );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
pHorzScrollBar->SetScrollPos( (int)horz_position.iVal );  
pVertScrollBar->SetScrollPos( (int)vert_position.iVal );  
.  
.
```

Visual Basic Example

```
Dim rc As Integer  
Dim horz_pos, vert_post As Variant  
.  
.  
  
rc = ArsOle.SetDocCurrentPage( 46 )  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End
```

```

End If

rc = ArsOle.GetDocScrollPositions( horz_pos, vert_pos )
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

sbHorz.Value = horz_pos
sbVert.Value = vert_pos

.
.

```

GetDocType

Note: This method intended for use with C/C++.

Method:

```

short GetDocType(
    long Index,
    VARIANT * pType,
    LPUNKNOWN pExtension )

```

Parameters:

Index

Specifies the zero-based index of a document within the document list of the active folder. If this value is less than zero, then the open document is used.

pType

Points to a variable to receive the document type of the specified document. The document type will be one of the document type values found in ARSOLEEX.H, such as ARS_OLE_DOC_TYPE_AFP.

pExtension

Points to a string to receive the file extension of the document. This value is returned only if the document type is ARS_OLE_DOC_TYPE_USER_DEF.

Description:

Retrieves the document type. If the document type is ARS_OLE_DOC_TYPE_USER_DEF, then the file extension is also retrieved.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

GetTypeForDoc

Example

The following example retrieves the document type for the third item in the document list.

```

VARIANT type;
char ext[ 20 ];
CArsOle * pArsCtrl;
short rc;

.
.
.

```

```

rc = pArsCtrl->GetDocType( 2, &type, extension );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

.
.
.

```

GetDocZoom

Method:

```

short GetDocZoom(
    VARIANT * pCurrentZoomPercent,
    VARIANT * pMinZoomPercent,
    VARIANT * pMaxZoomPercent )

```

Parameters:

pCurrentZoomPercent

Points to a variable to receive the current zoom percent. On return, this variable is set to type VT_I2.

pMinZoomPercent

Points to a variable to receive the minimum zoom percent. On return, this variable is set to type VT_I2.

pMaxZoomPercent

Points to a variable to receive the maximum zoom percent. On return, this variable is set to type VT_I2.

Description:

The current, minimum, and maximum zoom percents for the document are returned in the specified variables.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

SetDocZoom

Example:

The following example retrieves the current, minimum, and maximum zoom percents.

C/C++ Example

```

CArsOle * pArsCtrl;
short rc, current_zoom, min_zoom, max_zoom;
VARIANT var1, var2, var3;

.
.

rc = pArsCtrl->GetDocZoom( &var1, &var2, &var3 );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

current_zoom = var1.iVal;
min_zoom = var2.iVal;

```

```
max_zoom = var3.iVal;
```

```
.  
.
```

Visual Basic Example

```
Dim rc As Integer  
Dim current_zoom, min_zoom, max_zoom As Variant  
  
.  
.  
  
rc = ArsOle.GetDocZoom (current_zoom, min_zoom, max_zoom)  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If  
  
.  
.
```

GetFolderDisplayFieldName

Note: This method intended for use with Visual Basic.

Method:

```
short GetFolderDisplayFieldName(  
    short Index,  
    BSTR * pName )
```

Parameters:

Index

Specifies the zero-based index of the name to be returned. It must be a number greater than or equal to zero and less than the value returned by GetNumFolderDisplayFields.

pName

Points to a BSTR to receive the name of the field.

Description:

The specified field name is returned in pName.

GetFolderDisplayFieldName or GetFolderDisplayFieldNames can be used to retrieve the folder display field names. An application should use the one which is more convenient in its environment.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

GetNumFolderDisplayFields, GetFolderDisplayFieldNames

Example:

The following example creates a list box of the folder document list names and associated values and opens the document selected by a user.

Visual Basic Example

```
Dim rc, count, i, j As Integer  
Dim num_fields, num_docs As Variant  
Dim Names() As String  
Dim Line As String  
Dim Temp As String
```

```

.
.
.
rc = ArsOle.GetNumFolderDisplayFields(num_fields)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

ReDim Names(num_fields)

For count = 0 To num_fields -1
    rc = ArsOle.GetFolderDisplayFieldName(count, Temp)
    Names(count) = Temp
Next count

rc = ArsOle.GetNumDocsInList(num_docs)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

For j = 0 To num_docs -1
    For i = 0 To num_fields -1
        rc = ArsOle.GetDocDisplayValue(j, i, Temp)

        Line = Line + Names(i) + " = " + Temp
        If i < num_fields Then
            Line = Line + ", "
        End If
    Next i

    lbDocs.AddItem Line
Next j

.
.
.
'During OK button processing:

rc = ArsOle.OpenDoc (lbDocs.List(lbDocs.ListIndex), "", 0)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

```

GetFolderDisplayFieldNames

Note: This method intended for use with C/C++.

Method:

```

short GetFolderDisplayFieldNames(
    IUnknown * pNames,
    short MaxNames )

```

Parameters:

pNames

Points to an array of ArsOleNames to receive the names of the display fields for the active folder. The array must have at least MaxNames elements.

MaxNames

Specifies the maximum number of names to be returned.

Description:

The names of the display fields for the active folder, up to a maximum of MaxNames, are returned in pNames. Each name is a null-terminated character string.

GetFolderDisplayFieldName or GetFolderDisplayFieldNames can be used to retrieve the folder display field names. An application should use the one which is more convenient in its environment.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

GetNumFolderDisplayFields, GetFolderDisplayFieldName

Example:

The following example creates a list box of the folder document list names and associated values and opens the document selected by a user.

C/C++ Example

```
CArsOle * pArsCtrl;
ArsOleName * pNames;
ArsOleValue * pValues;
CListBox * pDocList;
char * pLine;
short rc, k, opr, num_fields;
long j, num_docs;
int size;
VARIANT vari;
.
.
// During dialog initialization:

rc = pArsCtrl->GetNumFolderDisplayFields( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_fields = var.iVal;
.
.
.
pNames = new ArsOleName[ max( num_fields, 1 ) ];
rc = pArsCtrl->GetFolderDisplayFieldNames( (IUnknown*)pNames, num_fields );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

rc = pArsCtrl->GetNumDocsInList( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_docs = var.lVal;

pValues = new ArsOleValue[ max( num_fields, 1 ) ];

size = num_fields * ( sizeof(ArsOleName) + sizeof(ArsOleValue) + 5 );
pLine = new char[ size ];
for ( j = 0, pLine[0] = '\0'; j < num_docs; j++ )
{
    rc = pArsCtrl->GetDocDisplayValues( j, pValues, num_fields );
    if ( rc != ARS_OLE_RC_SUCCESS )
        ERROR;

    for ( k = 0; k < num_fields; k++ )
    {
        strcat( pLine, pNames[k] );
        strcat( pLine, " = " );
    }
}
```

```

        strcat( pLine, pValues[k] );
        if ( k < num_fields - 1 )
            strcat( pLine, ", " );
    }
    pDocList->InsertString( -1, pLine );
}
pDocList->SetCurSel( 0 );
.
.
// During OK button processing:

rc = pArsCtrl->OpenDoc( (long)pDocList->GetCurSel( ) , NULL, 0 );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.

```

GetFolderFieldName

Note: This method intended for use with Visual Basic.

Method:

```

short GetFolderFieldName(
    short Index,
    BSTR * pName )

```

Parameters:

Index

Specifies the zero-based index of the value to be returned. It must be a number greater than or equal to zero and less than the value returned by GetNumFolderFields.

pName

Points to a BSTR to receive the field name.

Description:

The specified value is returned in pName.

GetFolderFieldName or GetFolderFieldNames can be used to retrieve the folder field names. An application should use the one which is more convenient in its environment.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

GetFolderFieldNames, GetNumFolderFields, StoreDoc

Example:

Visual Basic Example

```

Dim rc, count As Integer
Dim num_fields As Variant
Dim FieldNames() As String
Dim Temp As String

.
.

rc = ArsOle.GetNumFolderFields (num_fields)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End

```

```

End If

ReDim FieldNames (num_fields - 1)

For count = 0 To num_fields - 1
    rc = ArsOle.GetFolderFieldName (count, Temp)
    FieldNames(count) = Temp
Next count

.
.

```

GetFolderFieldNames

Note: This method intended for use with C/C++.

Method:

```

short GetFolderFieldNames(
    IUnknown * pNames,
    short MaxNames )

```

Parameters:

pNames

Points to an array of ArsOleValues to receive the folder field names. The array must have at least MaxNames elements.

MaxNames

Specifies the maximum number of names to be returned.

Description:

The names of the folder fields, up to a maximum of MaxNames, are returned in pNames. Each name is a null-terminated character string.

The names are placed in the array in the same sequence that should be used with the method StoreDoc.

GetFolderFieldName or GetFolderFieldNames can be used to retrieve the folder field names. An application should use the one which is more convenient in its environment.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

GetFolderFieldName, GetNumFolderFields, StoreDoc

Example:

The following example demonstrates the StoreDoc method. First the folder fields are displayed along with entry fields so that the user can enter field values. Then those values are used to store a new document into Content Manager OnDemand.

C/C++ Example

```

VARIANT var;
CArsOle * pArsCtrl;
ArsOleName * pNames;
short rc, j;

.
.

```

```

rc = pArsCtrl->GetNumFolderFields( &var );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

// m_NumFolderFields is a class variable
m_NumFolderFields = var.iVal;

pNames = new ArsOleName[ max( m_NumFolderFields, 1 ) ];
rc = pArsCtrl->GetFolderFieldNames( (IUnknown*)pNames,
                                   m_NumFolderFields );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
for ( j = 0; j < m_NumFolderFields; j++ )
    GetDlgItem( IDC_FIELD1_TEXT + j )->SetWindowText( pNames[j] );

// During OK button processing

CArsOle * pArsCtrl;
short rc, j;
CString fields[16];
SAFEARRAY * pSA;
VARIANT var;
BSTR bstrElement;
long i;

pSA = SafeArrayCreateVector(VT_BSTR, 0, m_NumFolderFields);
if ( pSA == NULL )
    ERROR;

for ( j = 0; j < m_NumFolderFields; j++ )
    GetDlgItem( IDC_FIELD1_EDIT + j )->GetWindowText( fields[j] );

for ( i = 0; i < m_NumFolderFields; i++ )
{
    bstrElement = fields[i].AllocSysString();
    if (bstrElement == NULL)
        ERROR;
    SafeArrayPutElement (pSA, &i, bstrElement);
}

var.vt = VT_ARRAY | VT_BSTR;
var.parray = pSA;

rc = pArsCtrl->StoreDoc( "G:\\download\\file.afp",
                       "BKH-CRD",
                       "BKH-CRD",
                       &var );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

```

GetFolderName

Note: This method intended for use with Visual Basic.

Method:

```

short GetFolderName(
    short Index,
    BSTR * pName )

```

Parameters:

Index

Specifies the zero-based index of the name to be returned. It must be a number greater than or equal to zero and less than the value returned by GetNumFolders.

pName

Points to a BSTR to receive the name of the folder.

Description:

The specified folder name is returned in pName.

GetFolderName or GetFolderNames can be used to retrieve the folder names. An application should use the one which is more convenient in its environment.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

GetNumFolders, GetFolderNames, OpenFolder

Example:

Visual Basic Example

```
Dim rc, count As Integer
Dim num_folders As Variant
Dim Temp As String

.
.

rc = ArsOle.GetNumFolders (num_folders)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

For count = 0 To num_folders -1
    rc = ArsOle.GetFolderName(count, Temp)
    lbFolders.AddItem Temp
Next count

.
.

' During OK button processing

rc = ArsOle.OpenFolder (lbFolders.List(lbFolders.ListItem))
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If
```

GetFolderNames

Note: This method intended for use with C/C++.

Method:

```
short GetFolderNames(
    IUnknown * pNames,
    short MaxNames )
```

Parameters:

pNames

Points to an array of `ArsOleNames` to receive the names of the folders available for the current server. The array must have at least `MaxNames` elements.

MaxNames

Specifies the maximum number of names to be returned.

Description:

The names of the folders available for the current server, up to a maximum of `MaxNames`, are returned in `pNames`. Each name is a null-terminated character string.

`GetFolderName` or `GetFolderNames` can be used to retrieve the folder names. An application should use the one which is more convenient in its environment.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

`GetNumFolders`, `GetFolderName`, `OpenFolder`

Example:

The following example retrieves the names of all folders available for the current server, puts them in a `ComboBox` control, retrieves the chosen folder, and performs an open for that folder.

C/C + + Example

```
CArsOle * pArsCtrl;
ArsOleName * pFolderNames;
CComboBox * pFoldersList;
char folder[ sizeof( ArsOleName ) ];
short rc, j, num_folders;
int index;
VARIANT vari;
.
.

// During dialog initialization:

rc = pArsCtrl->GetNumFolders( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_folders = var.iVal;

pFolderNames = new ArsOleName[ max( num_folders, 1 ) ];
rc = pArsCtrl->GetFolderNames( (IUnknown*)pFolderNames, num_folders );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

for ( j = 0; j < num_folders; j++ )
    index = pFoldersList->AddString( pFolderNames[j] );
pFoldersList->SetCurSel( 0 );
.
.

// During OK button processing:

pFoldersList->GetWindowText( folder, sizeof(folder) );

rc = pArsCtrl->OpenFolder( folder );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.
```

GetFolderSearchFieldName

Note: This method intended for use with Visual Basic.

Method:

```
short GetFolderSearchFieldName(  
    short Index,  
    BSTR * pName )
```

Parameters:

Index

Specifies the zero-based index of the name to be returned. It must be a number greater than or equal to zero and less than the value returned by GetNumFolderSearchFields.

pName

Points to a BSTR to receive the name of the field.

Description:

The specified field name is returned in pName.

GetFolderSearchFieldName or GetFolderSearchFieldNames can be used to retrieve the folder search field names. An application should use the one which is more convenient in its environment.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

GetNumFolderSearchFields, GetFolderSearchFieldNames,
SetFolderSearchFieldData, SearchFolder

Example:

Visual Basic Example

```
Dim rc, count, i, j As Integer  
Dim num_fields, num_docs As Variant  
Dim Names() As String  
Dim Line As String  
Dim Temp As String  
Dim Oprs As Variant  
.  
.  
.  
Oprs = Array ("Equal", "Not Equal", ..., "Like", "Not Like")  
  
rc = ArsOle.GetNumFolderSearchFields(num_fields)  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If  
  
ReDim Names(num_fields -1)  
  
For count = 0 To num_fields -1  
    rc = ArsOle.GetFolderSearchFieldName(count, Temp)  
    Names(count) = Temp  
Next count  
  
for count = 0 To num_fields -1  
    lbFieldList.AddItem Names(count)  
Next count  
  
for count = 0 To UBound(Oprs)
```

```

        lbOprList.AddItem (Oprs(count))
    Next count
    .
    .
    .
    ' During SET FIELD button processing
    rc = ArsOle.SetFolderSearchFieldData (lbFieldList.List(lbFieldList.ListIndex),
                                          lbOprList.ListIndex,
                                          txtValue1.Value,
                                          txtValue2.Value)

    If rc <> ARS_OLE_RC_SUCCESS Then
        MsgBox "ERROR"
    End
End If

'During OK button processing:

rc = ArsOle.SearchFolder (False)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

```

GetFolderSearchFieldNames

Note: This method intended for use with C/C++.

Method:

```

short GetFolderSearchFieldNames(
    IUnknown * pNames,
    short MaxNames )

```

Parameters:

pNames

Points to an array of ArsOleNames to receive the names of the search fields for the active folder. The array must have at least MaxNames elements.

MaxNames

Specifies the maximum number of names to be returned.

Description:

The names of the search fields for the active folder, up to a maximum of MaxNames, are returned in pNames. Each name is a null-terminated character string.

GetFolderSearchFieldName or GetFolderSearchFieldNames can be used to retrieve the folder search field names. An application should use the one which is more convenient in its environment.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

GetNumFolderSearchFields, GetFolderSearchFieldName,
SetFolderSearchFieldData, SearchFolder

Example:

The following example retrieves the names of the active folder search fields, gives a user the opportunity to set the values for these fields, and initiates a search of the folder.

C/C++ Example

```

CArsOle * pArsCtrl;
ArsOleName * pNames;
CListBox * pFieldList, * pOprList;
CEdit * pValue1, * pValue2;
char name[ sizeof( ArsOleName ) ];
char value1[ sizeof( ArsOleValue ) ];
char value2[ sizeof( ArsOleValue ) ];
short rc, j, opr, num_fields;
VARIANT vari;

.
.
struct _OprMap
{
    short code;
    char * pText;
} OprMap

.
.
.
static OprMap Oprs[] =
{ { ARS_OLE_OPR_EQUAL,          "Equal" },
  { ARS_OLE_OPR_NOT_EQUAL,      "Not Equal" },
  .
  { ARS_OLE_OPR_LIKE,           "Like" },
  { ARS_OLE_OPR_NOT_LIKE,       "Not Like" } };

#define NUM_OPRS ( sizeof(Oprs) / sizeof(OprMap) )

// During dialog initialization:

rc = pArsCtrl->GetNumFolderSearchFields( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_fields = var.iVal;

pNames = new ArsOleName[ max( num_fields, 1 ) ];
rc = pArsCtrl->GetFolderSearchFieldNames( (IUnknown*)pNames, num_fields );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

for ( j = 0; j < num_fields; j++ )
    pFieldList->InsertString( -1, pNames[j] );
pFieldList->SetCurSel( 0 );

for ( j = 0; j < NUM_OPRS; j++ )
{
    pOprList->InsertString( -1, Oprs[j].pText );
    pOprList->SetItemData( j, (DWORD)Oprs[j].code );
}
pOprList->SetCurSel( 0 );

.
.
.
.
// During SET FIELD button processing:

pFieldList->GetText( pFieldList->GetCurSel( ), name );
opr = (short)pOprList->GetItemData( pOprList->GetCurSel( ) );
pValue1->GetWindowText( value1, sizeof(value1) );
pValue2->GetWindowText( value2, sizeof(value2) );

rc = pArsCtrl->SetFolderSearchFieldData( name, opr, value1, value2 );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

```

```

        .
        .
        // During OK button processing:

        rc = pArsCtrl->SearchFolder( FALSE );
        if ( rc != ARS_OLE_RC_SUCCESS )
            ERROR;
        .
        .

```

GetNumDocAnnotations

Method:

```

short GetNumDocAnnotations(
    VARIANT * pNumAnnotations )

```

Parameters:

pNumAnnotations

Points to a variable to receive the number of annotations attached to the document. On return, this variable is set to type VT_I4.

Description:

The number of annotations attached to the document is returned in the specified variable.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

GetDocAnnotation, GetAnnotationForDoc

Example:

The following example retrieves the annotations for a document.

C/C++ Example

```

VARIANT num_notes, page, ispublic, canbecopied;
CArsOle * pArsCtrl;
short rc, j;
char * pText, userid[100], datetime[200];

rc = pArsCtrl->GetNumDocAnnotations( &num_notes );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

pText = new char[35000];

for ( j = 0; j < num_notes.iVal; j++ )
{
    rc = pArsCtrl->GetDocAnnotation( j,
                                    (LPUNKNOWN)pText,
                                    (LPUNKNOWN)userid,
                                    (LPUNKNOWN)datetime,
                                    &page,
                                    &ispublic,
                                    &canbecopied );

    if ( rc != ARS_OLE_RC_SUCCESS )
        ERROR;

    // Process annotation
}

delete pText;

```

Visual Basic Example

```
Dim rc, j As Integer
Dim num_notes, page, ispublic, canbecopied As Variant
Dim text As String
Dim userid As String
Dim datetime As String

rc = ArsOle.GetNumDocAnnotations( num_notes );
if rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

For j = 0 To num_notes -1
    rc = ArsOle.GetAnnotationForDoc( j,
                                     text,
                                     userid,
                                     datetime,
                                     page,
                                     ispublic,
                                     canbecopied )

    if rc <> ARS_OLE_RC_SUCCESS Then
        MsgBox "ERROR"
    End
End If

    ' Process Annotation

Next j
```

GetNumDocsInList

Method:

```
short GetNumDocsInList(
    VARIANT * pNumDocs )
```

Parameters:

pNumDocs

Points to a variable to receive the number of documents in the document list of the active folder. On return, this variable is set to type VT_I4.

Description:

The number of documents in the document list of the active folder is returned in the specified variable.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

GetDocDisplayValues, OpenDoc

Example:

The following example creates a list box of the folder document list names and associated values and opens the document selected by a user.

C/C++ Example

```
CArsOle * pArsCtrl;
ArsOleName * pNames;
ArsOleValue * pValues;
CListBox * pDocList;
char * pLine;
short rc, k, opr, num_fields;
long j, num_docs;
```

```

int size;
VARIANT vari;
.
.
// During dialog initialization:

rc = pArsCtrl->GetNumFolderDisplayFields( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_fields = var.iVal;

pNames = new ArsOleName[ max( num_fields, 1 ) ];
rc = pArsCtrl->GetFolderDisplayFieldNames( (IUnknown*)pNames, num_fields );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

rc = pArsCtrl->GetNumDocsInList( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_docs = var.lVal;

pValues = new ArsOleValue[ max( num_fields, 1 ) ];
.
.
.
.
size = num_fields * ( sizeof(ArsOleName) + sizeof(ArsOleValue) + 5 );
pLine = new char[ size ];
for ( j = 0, pLine[0] = '\0'; j < num_docs; j++ )
{
    rc = pArsCtrl->GetDocDisplayValues( j, pValues, num_fields );
    if ( rc != ARS_OLE_RC_SUCCESS )
        ERROR;

    for ( k = 0; k < num_fields; k++ )
    {
        strcat( pLine, pNames[k] );
        strcat( pLine, " = " );
        strcat( pLine, pValues[k] );
        if ( k < num_fields - 1 )
            strcat( pLine, ", " );
    }
    pDocList->InsertString( -1, pLine );
}
pDocList->SetCurSel( 0 );
.
.
// During OK button processing:

rc = pArsCtrl->OpenDoc( (long)pDocList->GetCurSel( ) , NULL, 0 );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.

```

Visual Basic Example

```

Dim rc, count, i, j As Integer
Dim num_fields, num_docs As Variant
Dim Names() As String
Dim Line As String
Dim Temp As String
.
.
.
rc = ArsOle.GetNumFolderDisplayFields(num_fields)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"

```

```

        End
    End If

    ReDim Names(num_fields -1)

    For count = 0 To num_fields -1
        rc = ArsOle.GetFolderDisplayFieldName(count, Temp)
        Names(count) = Temp
    Next count

    rc = ArsOle.GetNumDocsInList(num_docs)
    If rc <> ARS_OLE_RC_SUCCESS Then
        MsgBox "ERROR"
    End
    End If

    For j = 0 To num_docs -1
        For i = 0 To num_fields -1
            rc = ArsOle.GetDocDisplayValue(j, i, Temp)

            Line = Line + Names(i) + " = " + Temp
            If i < num_fields Then
                Line = Line + ", "
            End If
        Next i

        lbDocs.AddItem Line
    Next j

    .
    .
    .
    'During OK button processing:

    rc = ArsOle.OpenDoc (lbDocs.List(lbDocs.ListIndex), "", 0)
    If rc <> ARS_OLE_RC_SUCCESS Then
        MsgBox "ERROR"
    End
    End If

```

GetNumFolderDisplayFields

Method:

```

short GetNumFolderDisplayFields(
    VARIANT * pNumFields )

```

Parameters:

pNumFields

Points to a variable to receive the number of display fields for the active folder. On return, this variable is set to type VT_I2.

Description:

The number of display fields for the active folder is returned in the specified variable.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

GetFolderDisplayFieldNames

Example:

The following example creates a list box of the folder document list names and associated values and opens the document selected by a user.

C/C++ Example

```
CArsOle * pArsCtrl;
ArsOleName * pNames;
ArsOleValue * pValues;
CListBox * pDocList;
char * pLine;
short rc, k, opr, num_fields;
long j, num_docs;
int size;
VARIANT vari;
.
.
// During dialog initialization:

rc = pArsCtrl->GetNumFolderDisplayFields( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_fields = var.iVal;

pNames = new ArsOleName[ max( num_fields, 1 ) ];
rc = pArsCtrl->GetFolderDisplayFieldNames( (IUnknown*)pNames, num_fields );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

rc = pArsCtrl->GetNumDocsInList( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_docs = var.lVal;
.
.
.
pValues = new ArsOleValue[ max( num_fields, 1 ) ];
size = num_fields * ( sizeof(ArsOleName) + sizeof(ArsOleValue) + 5 );
pLine = new char[ size ];
for ( j = 0, pLine[0] = '\\0'; j < num_docs; j++ )
{
    rc = pArsCtrl->GetDocDisplayValues( j, pValues, num_fields );
    if ( rc != ARS_OLE_RC_SUCCESS )
        ERROR;

    for ( k = 0; k < num_fields; k++ )
    {
        strcat( pLine, pNames[k] );
        strcat( pLine, " = " );
        strcat( pLine, pValues[k] );
        if ( k < num_fields - 1 )
            strcat( pLine, ", " );
    }
    pDocList->InsertString( -1, pLine );
}
pDocList->SetCurSel( 0 );
.
.
// During OK button processing:

rc = pArsCtrl->OpenDoc( (long)pDocList->GetCurSel( ) , NULL, 0 );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.
```

Visual Basic Example

```
Dim rc, count, i, j As Integer
Dim num_fields, num_docs As Variant
Dim Names() As String
```

```

Dim Line As String
Dim Temp As String
.
.
.
rc = ArsOle.GetNumFolderDisplayFields(num_fields)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

ReDim Names(num_fields -1)

For count = 0 To num_fields -1
    rc = ArsOle.GetFolderDisplayFieldName(count, Temp)
    Names(count) = Temp
Next count

rc = ArsOle.GetNumDocsInList(num_docs)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

For j = 0 To num_docs -1
    For i = 0 To num_fields -1
        rc = ArsOle.GetDocDisplayValue(j, i, Temp)

        Line = Line + Names(i) + " = " + Temp
        If i < num_fields Then
            Line = Line + ", "
        End If
    Next i

    lbDocs.AddItem Line
Next j
.
.
.
'During OK button processing:

rc = ArsOle.OpenDoc (lbDocs.List(lbDocs.ListIndex), "", 0)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

```

GetNumFolderFields

Method:

```

short GetNumFolderFields(
    VARIANT * pNumfields )

```

Parameters:

pNumfields

Points to a variable to receive the number of folder fields for the active folder. On return, this variable is set to type VT_I2.

Description:

The number of folder fields for the active folder is returned in the specified variable.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

GetFolderFieldName, GetFolderFieldNames

Example:

The following example demonstrates the StoreDoc method. First the folder fields are displayed along with entry fields so that the user can enter field values. Then those values are used to store a new document into OnDemand.

C/C++ Example

```
VARIANT var;
CArsOle * pArsCtrl;
ArsOleName * pNames;
short rc, j;
.
.
rc = pArsCtrl->GetNumFolderFields( &var );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

// m_NumFolderFields is a class variable
m_NumFolderFields = var.iVal;

pNames = new ArsOleName[ max( m_NumFolderFields, 1 ) ];
rc = pArsCtrl->GetFolderFieldNames( (IUnknown*)pNames,
    m_NumFolderFields );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

for ( j = 0; j < m_NumFolderFields; j++ )
    GetDlgItem( IDC_FIELD1_TEXT + j )->SetWindowText( pNames[j] );

// During OK button processing
CArsOle * pArsCtrl;
short rc, j;
CString fields[16];
SAFEARRAY * pSA;
VARIANT var;
BSTR bstrElement;
long i;
pSA = SafeArrayCreateVector(VT_BSTR, 0, m_NumFolderFields);
if ( pSA == NULL )
    ERROR;

for ( j = 0; j < m_NumFolderFields; j++ )
    GetDlgItem( IDC_FIELD1_EDIT + j )->GetWindowText( fields[j] );

for ( i = 0; i < m_NumFolderFields; i++ )
{
    bstrElement = fields[i].AllocSysString();
    if (bstrElement == NULL)
        ERROR;
    SafeArrayPutElement (pSA, &i, bstrElement);
}

var.vt = VT_ARRAY | VT_BSTR;
var.parray = pSA;

rc = pArsCtrl->StoreDoc( "G:\\download\\file.afp",
    "BKH-CRD",
    "BKH-CRD",
    &var );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
```

Visual Basic Example

```

Dim rc, count As Integer
Dim num_fields As Variant
Dim FieldNames() As String
Dim Temp As String
.
.

rc = ArsOle.GetNumFolderFields (num_fields)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

ReDim FieldNames (num_fields - 1)

For count = 0 To num_fields - 1
    rc = ArsOle.GetFolderFieldName (count, Temp)
    FieldNames(count) = Temp
Next count

.
.

```

GetNumFolders

Method:

```

short GetNumFolders(
    VARIANT * pNumFolders )

```

Parameters:

pNumFolders

Points to a variable to receive the number of folders available for the current server. On return, this variable is set to type VT_I2.

Description:

The number of folders available for the current server is returned in the specified variable. This value can be used with the GetFolderNames method to prepare for opening a folder.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

GetFolderNames, OpenFolder

Example:

The following example retrieves the names of all folders available for the current server, puts them in a ComboBox control, retrieves the chosen folder, and performs an open for that folder.

C/C++ Example

```

CArsOle * pArsCtrl;
ArsOleName * pFolderNames;
CComboBox * pFoldersList;
char folder[ sizeof( ArsOleName ) ];
short rc, j, num_folders;
int index;
VARIANT vari;
.
.
// During dialog initialization:

rc = pArsCtrl->GetNumFolders( &vari );

```

```

if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_folders = var.iVal;

pFolderNames = new ArsOleName[ max( num_folders, 1 ) ];
rc = pArsCtrl->GetFolderNames( (IUnknown*)pFolderNames, num_folders );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

for ( j = 0; j < num_folders; j++ )
    index = pFoldersList->AddString( pFolderNames[j] );
pFoldersList->SetCurSel( 0 );
.
.

// During OK button processing:

pFoldersList->GetWindowText( folder, sizeof(folder) );

rc = pArsCtrl->OpenFolder( folder );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.

```

Visual Basic Example

```

Dim rc, count As Integer
Dim num_folders As Variant
Dim Temp As String

.
.

rc = ArsOle.GetNumFolders (num_folders)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

ReDim FolderNames(num_folders -1)

For count = 0 To num_folders -1
    rc = ArsOle.GetFolderName(count, Temp)
    lbFolders.AddItem Temp
Next count

.
.

' During OK button processing

rc = ArsOle.OpenFolder (lbFolders.List(lbFolders.ListItem))
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

```

GetNumFolderSearchFields

Method:

```

short GetNumFolderSearchFields(
    VARIANT * pNumFields )

```

Parameters:

pNumFields

Points to a variable to receive the number of search fields for the active folder. On return, this variable is set to type VT_I2.

Description:

The number of search fields for the active folder is returned in the specified variable. This value can be used with the GetFolderSearchFieldNames method to prepare for setting the search field values for a folder.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

GetFolderSearchFieldNames, SetFolderSearchFieldData, SearchFolder

Example:

The following example retrieves the names of the active folder search fields, gives a user the opportunity to set the values for these fields, and initiates a search of the folder.

C/C++ Example

```
CArsOle * pArsCtrl;
ArsOleName * pNames;
CListBox * pFieldList, * pOprList;
CEdit * pValue1, * pValue2;
char name[ sizeof( ArsOleName ) ];
char value1[ sizeof( ArsOleValue ) ];
char value2[ sizeof( ArsOleValue ) ];
short rc, j, opr, num_fields;
VARIANT vari;

.
.
struct _OprMap
{
    short code;
    char * pText;
} OprMap

static OprMap Oprs[] =
{ { ARS_OLE_OPR_EQUAL,          "Equal" },
  { ARS_OLE_OPR_NOT_EQUAL,     "Not Equal" },
  .
  { ARS_OLE_OPR_LIKE,          "Like" },
  { ARS_OLE_OPR_NOT_LIKE,      "Not Like" } };

#define NUM_OPRS ( sizeof(Oprs) / sizeof(OprMap) )

.
.
// During dialog initialization:

rc = pArsCtrl->GetNumFolderSearchFields( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_fields = vari.iVal;

pNames = new ArsOleName[ max( num_fields, 1 ) ];
rc = pArsCtrl->GetFolderSearchFieldNames( (IUnknown*)pNames, num_fields );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

for ( j = 0; j < num_fields; j++ )
    pFieldList->InsertString( -1, pNames[j] );
pFieldList->SetCurSel( 0 );
```

```

for ( j = 0; j < NUM_OPRS; j++ )
{
    pOprList->InsertString( -1, Oprs[j].pText );
    pOprList->SetItemData( j, (DWORD)Oprs[j].code );
}
pOprList->SetCurSel( 0 );
.
.
// During SET FIELD button processing:

pFieldList->GetText( pFieldList->GetCurSel( ), name );
opr = (short)pOprList->GetItemData( pOprList->GetCurSel( ) );
pValue1->GetWindowText( value1, sizeof(value1) );
pValue2->GetWindowText( value2, sizeof(value2) );

rc = pArsCtrl->SetFolderSearchFieldData( name, opr, value1, value2 );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.
// During OK button processing:

rc = pArsCtrl->SearchFolder( FALSE );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.

```

Visual Basic Example

```

Dim rc, count, i, j As Integer
Dim num_fields, num_docs As Variant
Dim Names() As String
Dim Line As String
Dim Temp As String
Dim Oprs As Variant
.
.
.
Oprs = Array ("Equal", "Not Equal", ..., "Like", "Not Like")

rc = ArsOle.GetNumFolderSearchFields(num_fields)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

ReDim Names(num_fields -1)

For count = 0 To num_fields -1
    rc = ArsOle.GetFolderSearchFieldName(count, Temp)
    Names(count) = Temp
Next count

for count = 0 To num_fields -1
    lbFieldList.AddItem Names(count)
Next count

for count = 0 To UBound(Oprs)
    lbOprList.AddItem (Oprs(count))
Next count
.
.
.
' During SET FIELD button processing
rc = ArsOle.SetFolderSearchFieldData (lbFieldList.List(lbFieldList.ListIndex),
                                       lbOprList.ListIndex,
                                       txtValue1.Value,
                                       txtValue2.Value)

```

```

If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If
'During OK button processing:

rc = ArsOle.SearchFolder (False)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

```

GetNumServerPrinters

Method:

```

short GetNumServerPrinters(
    VARIANT * pNumServerPrinters )

```

Parameters:

pNumServerPrinters

Points to a variable to receive the number of server printers available for the current server. On return, this variable is set to type VT_I2.

Description:

The number of server printers available is returned in the specified variable. This value can be used with the GetServerPrinter and GetServerPrinterInfo methods.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

GetServerPrinter, GetServerPrinterInfo

Example:

The following example retrieves the names and attributes of the available server printers.

C/C++ Example

```

CArsOle * pArsCtrl;

short rc, j, num_prts;
char name[ 200 ];
VARIANT vari, type;
.
.
.

rc = pArsCtrl->GetNumServerPrinters( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_prts = vari.iVal;

for ( j = 0; j < num_prts; j++ )
{
    rc = pArsCtrl->GetServerPrinter( j, name, type );
    if ( rc != ARS_OLE_RC_SUCCESS )
        ERROR;

    // Process name and type
}

```

```

}
.
.
.

```

Visual Basic Example

```

Dim rc, count As Integer
Dim num_prts, type As Variant
Dim name As String

.
.

rc = ArsOle.GetNumServers (num_prts)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

For count = 0 To num_prts -1
    rc = ArsOle.GetServerPrinterInfo (count, name, type)
    ' Process name and type
Next count

.
.

```

GetNumServers

Method:

```

short GetNumServers(
    VARIANT * pNumServers )

```

Parameters:

pNumServers

Points to a variable to receive the number of servers available for logon. On return, this variable is set to type VT_I2.

Description:

The number of servers available for logon is returned in the specified variable. This value can be used with the GetServerNames method to prepare for a logon.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

GetServerNames, Logon

Example:

The following example retrieves the names of all servers available for logon, puts them in a ComboBox control, retrieves the chosen server, userid, and password, and performs a logon.

C/C++ Example

```

CArsOle * pArsCtrl;
ArsOleName * pServerNames;
CComboBox * pServersList;
CEdit * pUserId;
CEdit * pPassword;
char server[ sizeof( ArsOleName ) ];
char user[ sizeof( ArsOleName ) ];
char password[ sizeof( ArsOleName ) ];

```

```

short rc, j, num_servers;
int index;
VARIANT vari;
.
.
// During dialog initialization:

rc = pArsCtrl->GetNumServers( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_servers = var.iVal;

pServerNames = new ArsOleName[ max( num_servers, 1 ) ];
rc = pArsCtrl->GetServerNames( (IUnknown*)pServerNames, num_servers );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

for ( j = 0; j < num_servers; j++ )
    index = pServersList->AddString( pServerNames[j] );
pServersList->SetCurSel( 0 );
.
.
// During OK button processing:

pServersList->GetWindowText( server, sizeof(server) );
pUserId->GetWindowText( user, sizeof(user) );
pPassword->GetWindowText( password, sizeof(password) );

rc = pArsCtrl->Logon( server, user, password );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.

```

Visual Basic Example

```

Dim rc, count As Integer
Dim num_servers As Variant
Dim Temp As String

.
.

rc = ArsOle.GetNumServers (num_servers)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

For count = 0 To num_servers -1
    rc = ArsOle.GetServerName(count, Temp)
    lbServers.AddItem Temp
Next count

.
.

' During OK button processing

rc = ArsOle.Logon (lbServers.List(lbServers.ListItem), txtUser.Value, txtPasswd.Value)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

```

GetResourceCacheMode

Method:

```

short SetResourceCacheMode(
    short Mode )

```

Parameters:

Mode Specifies the new resource cache mode. This must be one of the following mode values found in ARSOLEEX.H:

ARS_OLE_RES_MODE_RETAIN

ARS_OLE_RES_MODE_ERASE_AFTER_RETRIEVE

ARS_OLE_RES_MODE_RETAIN specifies that any resource group retrieved during execution of a RetrieveDoc method be retained for use by subsequent RetrieveDoc or OpenDoc methods. Files containing these resource groups will not be erased until all instances of the control have been terminated. This is the default mode.

ARS_OLE_RES_MODE_ERASE_AFTER_RETRIEVE specifies that the file for any resource group retrieved during execution of a RetrieveDoc method be erased before returning to the caller. If subsequent RetrieveDoc or OpenDoc methods require access to the same resource group, another retrieval will be required.

Description:

The requested mode is set.

Return Value:

Refer to "Return Code" in the *Windows Client Customization Guide* for an explanation of the return code.

See Also:

RetrieveDoc

Example:

The following example sets the resource cache mode to retain.

C/C++ Example

```
CArsOle * pArsCtrl;
short rc;
.
.
rc = pArsCtrl->SetResourceCacheMode( ARS_OLE_RES_MODE_RETAIN );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.
```

Visual Basic Example

```
Dim rc As Integer
.
.
rc = ArsOle.SetResourceCacheMode (ARS_OLE_RES_MODE_RETAIN)
if rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If
.
.
```

GetServerName

Note: This method intended for use with Visual Basic.

Method:

```
short GetServerName(
    short Index,
```

BSTR * pName)

Parameters:

Index

Specifies the zero-based index of the name to be returned. It must be a number greater than or equal to zero and less than the value returned by GetNumServers.

pName

Points to a BSTR to receive the name of the server.

Description:

The specified server name is returned in pName.

GetServerName or GetServerNames can be used to retrieve the server names. An application should use the one which is more convenient in its environment.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

GetNumServers, GetServerNames, Logon

Example:

Visual Basic Example

```
Dim rc, count As Integer
Dim num_servers As Variant
Dim Temp As String

.
.

rc = ArsOle.GetNumServers (num_servers)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

For count = 0 To num_servers -1
    rc = ArsOle.GetServerName(count, Temp)
    lbServers.AddItem Temp
Next count

.
.

' During OK button processing

rc = ArsOle.Logon (lbServers.List(lbServers.ListItem), txtUser.Value, txtPasswd.Value)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If
```

GetServerNames

Note: This method intended for use with C/C++.

Method:

```
short GetServerNames(
    IUnknown * pNames,
    short MaxNames )
```

Parameters:**pNames**

Points to an array of `ArsOleNames` to receive the names of the servers available for logon. The array must have at least `MaxNames` elements.

MaxNames

Specifies the maximum number of names to be returned.

Description:

The names of the servers available for logon, up to a maximum of `MaxNames`, are returned in `pNames`. Each name is a null-terminated character string.

`GetServerName` or `GetServerNames` can be used to retrieve the server names. An application should use the one which is more convenient in its environment.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

`GetNumServers`, `GetServerName`, `Logon`

Example:

The following example retrieves the names of all servers available for logon, puts them in a `ComboBox` control, retrieves the chosen server, `userid`, and `password`, and performs a logon.

C/C++ Example

```
CArsOle * pArsCtrl;
ArsOleName * pServerNames;
CComboBox * pServersList;
CEdit * pUserId;
CEdit * pPassword;
char server[ sizeof( ArsOleName ) ];
char user[ sizeof( ArsOleName ) ];
char password[ sizeof( ArsOleName ) ];
short rc, j, num_servers;
int index;
VARIANT vari;
.
.
// During dialog initialization:

rc = pArsCtrl->GetNumServers( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_servers = var.iVal;

pServerNames = new ArsOleName[ max( num_servers, 1 ) ];
rc = pArsCtrl->GetServerNames( (IUnknown*)pServerNames, num_servers );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

for ( j = 0; j < num_servers; j++ )
    index = pServersList->AddString( pServerNames[j] );
pServersList->SetCurSel( 0 );
.
.

// During OK button processing:

pServersList->GetWindowText( server, sizeof(server) );
pUserId->GetWindowText( user, sizeof(user) );
```

```

pPassword->GetWindowText( password, sizeof(password) );

rc = pArsCtrl->Logon( server, user, password );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.

```

GetServerPrinter

Note: This method intended for use with C/C++.

Method:

```

short GetServerPrinter(
    short Index,
    LPUNKNOWN pName,
    VARIANT * pType )

```

Parameters:

Index

Specifies the zero-based index of the printer to be returned. It must be a number greater than or equal to zero and less than the value returned by GetNumServerPrinters.

pName

Points to a string to receive the name of the server printer.

pType

Points to a variable to receive the type of the server printer. It will be one of the following type values found in ARSOLEEX.H:

```

ARS_OLE_SERVER_PRINTER_PRINT
ARS_OLE_SERVER_PRINTER_PRINT_WITH_INFO
ARS_OLE_SERVER_PRINTER_FAX

```

On return, this variable is set to type VT_I2.

Description:

The server printer information is retrieved.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

GetNumServerPrinters, GetServerPrinterInfo

Example

The following example retrieves the names and attributes of the available server printers.

```

CArsOle * pArsCtrl;

short rc, j, num_prts;
char name[ 200 ];
VARIANT vari, type;
.
.
.

rc = pArsCtrl->GetNumServerPrinters( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

```

```

num_prts = vari.iVal;

for ( j = 0; j < num_prts; j++ )
{
    rc = pArsCtrl->GetServerPrinter( j, name, type );
    if ( rc != ARS_OLE_RC_SUCCESS )
        ERROR;

    // Process name and type
}
.
.
.

```

GetServerPrinterInfo

Note: This method intended for use with Visual Basic.

Method:

```

short GetServerPrinterInfo(
    short Index,
    BSTR * pName,
    VARIANT * pType )

```

Parameters:

Index

Specifies the zero-based index of the printer to be returned. It must be a number greater than or equal to zero and less than the value returned by GetNumServerPrinters.

pName

Points to a BSTR to receive the name of the server printer.

pType

Points to a variable to receive the type of the server printer. It will be one of the following type values found in ARSOLEEX.H:

```

    ARS_OLE_SERVER_PRINTER_PRINT
    ARS_OLE_SERVER_PRINTER_PRINT_WITH_INFO
    ARS_OLE_SERVER_PRINTER_FAX

```

Description:

The server printer information is retrieved.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

GetNumServerPrinters, GetServerPrinter

Example

The following example retrieves the names and attributes of the available server printers.

```

Dim rc, count As Integer
Dim num_prts, type As Variant
Dim name As String

.
.

```

```

rc = ArsOle.GetNumServers (num_prts)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

For count = 0 To num_prts -1
    rc = ArsOle.GetServerPrinterInfo (count, name, type)
    ' Process name and type
Next count

.
.

```

GetStoreDocInvalidFieldNum

Method:

```

short GetStoreDocInvalidFieldNum(
    VARIANT * pFieldNum )

```

Parameters:

pFieldNum

Points to a variable to receive the number of the invalid field. On return, this variable is set to type VT_I2.

Description:

If a StoreDoc method was previously used and the return code was one of the following:

```

ARS_OLE_RC_INVALID_DATE_FIELD
ARS_OLE_RC_INVALID_INTEGER_FIELD
ARS_OLE_RC_INVALID_DECIMAL_FIELD
ARS_OLE_RC_TOO_MANY_VALUE_CHARS
ARS_OLE_RC_INVALID_APPLGRP_FIELD_TYPE,

```

the field number returned is the zero-based index of the folder field which caused the problem. If a previous StoreDoc method did not return one of the listed codes, the value received is unpredictable.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

StoreDoc

Example

The following example retrieves the invalid field number following a StoreDoc.

C/C++ Example

```

CArsOle * pArsCtrl;
short rc;
SAFEARRAY * pSA;
VARIANT var;
BSTR bstrElement;
long i;
.
.
pSA = SafeArrayCreateVector(VT_BSTR, 0, 2);
if ( pSA == NULL )
    ERROR;

bstrElement = SysAllocStringByteLen ("255-546-667", 11);

```

```

i = 0;
SafeArrayPutElement (pSA, &i, bstrElement);
bstrElement = SysAllocStringByteLen ("06/07/94", 8);
i = 1;
SafeArrayPutElement (pSA, &i, bstrElement);

var.vt = VT_ARRAY | VT_BSTR;
var.parray = pSA;
rc = pArsCtrl->StoreDoc( "g:\\download \\file.afp",
                        "BKH-CRD",
                        "BKH-CRD",
                        &var );
if ( rc != ARS_OLE_RC_SUCCESS )
{
    if ( rc == ARS_OLE_RC_INVALID_DATE_FIELD |
        rc == ARS_OLE_RC_INVALID_INTEGER_FIELD |
        rc == ARS_OLE_RC_INVALID_DECIMAL_FIELD |
        rc == ARS_OLE_RC_TOOMANY_VALUE_CHARS |
        rc == ARS_OLE_RC_INVALID_APPLGRP_FIELD_TYPE )
    {
        VARIANT var1;
        rc = pArsCtrl->GetStoreDocInvalidFieldNum(&var1);
        .
        .
        .
    }
    ERROR;
}

```

Visual Basic Example

```

Dim values(2) As String
Dim rc As Integer
.
.
.
values(0) = "255-546-667"
values(1) = "06/07/94"
var = values

rc = ArsOle.StoreDoc ("g:\download \file.afp", _
                    "BKH-CRD", _
                    "BKH-CRD", _
                    var)
if rc <> ARS_OLE_RC_SUCCESS Then
    if rc = ARS_OLE_RC_INVALID_DATE_FIELD or
        rc = ARS_OLE_RC_INVALID_INTEGER_FIELD or
        rc = ARS_OLE_RC_INVALID_DECIMAL_FIELD or
        rc = ARS_OLE_RC_TOOMANY_VALUE_CHARS or
        rc = ARS_OLE_RC_INVALID_APPLGRP_FIELD_TYPE then
        rc = ArsOle.GetStoreDocInvalidFieldNum (var1)
    End If
    MsgBox "ERROR"
End
End If
.
.
.

```

GetTypeForDoc

Note: This method intended for use with Visual Basic.

Method:

```

short GetTypeForDoc(
    • long Index,

```

- VARIANT * **pType**,
- BSTR * **pExtension**)

Parameters:

Index

Specifies the zero-based index of a document within the document list of the active folder. If this value is less than zero, then the open document is used.

pType

Points to a variable to receive the document type of the specified document. The document type will be one of the document type values found in ARSOLEEX.H, such as ARS_OLE_DOC_TYPE_AFP.

pExtension

Points to a BSTR to receive the file extension of the document. This value is returned only if the document type is ARS_OLE_DOC_TYPE_USER_DEF.

Description:

Retrieves the document type. If the document type is ARS_OLE_DOC_TYPE_USER_DEF, then the file extension is also retrieved.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

GetDocType

Example

The following example retrieves the document type for the third item in the document list.

```
DIM rc As Integer
DIM type As Variant
DIM ext As String

.
.
.

rc = pArsCtrl->GetTypeForDoc(2, type, ext);
if rc <> ARS_OLE_RC_SUCCESS THEN
    MsgBox "ERROR"
End
Endif

.
.
.
```

IsDocHorzScrollRequired

Method:

short IsDocHorzScrollRequired(
VARIANT * **pRequired**)

Parameters:

pRequired

Points to a variable to receive the result. On return, this variable is set to type VT_I2.

Description:

If the width of the document data exceeds the width of the control window, the result variable is set to a non-zero value; otherwise, to zero. The displayed width of the document depends on the inherent width of its data, the type of data (that is, AFP verses Linedata), and the zoom factor.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

ScrollDocHorz

Example:

The following example initializes the horizontal scroll bar range, shows or hides the scroll bar after a document is opened or the zoom value is changed, and processes WM_HSCROLL messages.

C/C++ Example

```
CArsOle * pArsCtrl;
CScrollBar * pHorzScrollBar;
short rc, scroll_code;
VARIANT scroll_position, required;
.
.
// During initialization:

pHorzScrollBar->SetScrollRange( 0, ARS_OLE_SCROLL_RANGE );
pHorzScrollBar->ShowScrollBar( FALSE );

// After a document is opened or changing the zoom value:

rc = pArsCtrl->IsDocHorzScrollRequired( &required );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

pHorzScrollBar->ShowScrollBar( required.iVal );
.
.
// While processing a WM_HSCROLL message:

scroll_code = (short)LOWORD(wParam);
.
.
.
switch ( scroll_code )
{
    case SB_LINELEFT:
        scroll_code = ARS_OLE_SCROLL_LINELEFT;
        break;
    case SB_LINERIGHT:
        scroll_code = ARS_OLE_SCROLL_LINERIGHT;
        break;
    case SB_PAGELEFT:
        scroll_code = ARS_OLE_SCROLL_PAGELEFT;
        break;
    case SB_PAGERIGHT:
        scroll_code = ARS_OLE_SCROLL_PAGERIGHT;
        break;
    case SB_LEFT:
        scroll_code = ARS_OLE_SCROLL_LEFT;
        break;
    case SB_RIGHT:
        scroll_code = ARS_OLE_SCROLL_RIGHT;
        break;
```

```

        case SB_THUMBPOSITION:
            scroll_code = ARS_OLE_SCROLL_THUMBPOSITION;
            break;
        case SB_THUMBTRACK:
            scroll_code = ARS_OLE_SCROLL_THUMBTRACK;
            break;
        default:
            scroll_code = ARS_OLE_SCROLL_ENDSCROLL;
    }

    if ( scroll_code == (short)ARS_OLE_SCROLL_THUMBPOSITION ||
        scroll_code == (short)ARS_OLE_SCROLL_THUMBTRACK )
    {
        scroll_position.vt = VT_I2;
        scroll_position.iVal = (short)HIWORD(wParam);
    }

    rc = pArsCtrl->ScrollDocHorz( scroll_code, &scroll_position );
    if ( rc != ARS_OLE_RC_SUCCESS )
        ERROR;

    pHorzScrollBar->SetScrollPos( (int)scroll_position.iVal );
    .
    .

```

Visual Basic Example

```

Dim rc As Integer
Dim scroll_pos, required As Variant

.
.

' During initialization

sbHorz.Min = 0
sbHorz.Max = ARS_OLE_SCROLL_RANGE
sbHorz.Visible = False

' After a document is opened or changing the zoom value

rc = ArsOle.IsDocHorzScrollRequired (required)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

If required <> 0 Then
    sbHorz.Visible = True
End If

.
.

' During scroll bar Change method

Private Sub sbHorz_Change()
    Dim Diff As Integer
    Dim rc As Integer
    Dim ScrollCode As Integer
    Dim NewPos As Variant

    NewPos = 0
    Diff = sbHorz.Value - HorzScrollOld
    If Diff = sbHorz.LargeChange Then
        ScrollCode = ARS_OLE_SCROLL_PAGEDOWN
        rc = ArsOle.ScrollDocHorz(ScrollCode, NewPos)
        HorzScrollOld = NewPos
        sbHorz.Value = NewPos
    End If
End Sub

```

```

ElseIf Diff = -sbHorz.LargeChange Then
    ScrollCode = ARS_OLE_SCROLL_PAGEUP
    rc = ArsOle.ScrollDocHorz(ScrollCode, NewPos)
    HorzScroll101d = NewPos
    sbHorz.Value = NewPos
ElseIf Diff = sbHorz.SmallChange Then
    ScrollCode = ARS_OLE_SCROLL_LINEDOWN
    rc = ArsOle.ScrollDocHorz(ScrollCode, NewPos)
    HorzScroll101d = NewPos
    sbHorz.Value = NewPos
ElseIf Diff = -sbHorz.SmallChange Then
    ScrollCode = ARS_OLE_SCROLL_LINEUP
    rc = ArsOle.ScrollDocHorz(ScrollCode, NewPos)
    HorzScroll101d = NewPos
    sbHorz.Value = NewPos
Else
    ScrollCode = ARS_OLE_SCROLL_THUMBPOSITION
    NewPos = sbHorz.Value
    rc = ArsOle.ScrollDocHorz(ScrollCode, NewPos)
    HorzScroll101d = sbHorz.Value
End If
End Sub

```

Logoff

Method:

short Logoff()

Description:

Logs off from the current server.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

Logon

Example:

The following example performs a logoff.

C/C + + Example

```

CArsOle * pArsCtrl;
short rc;
.
.
rc = pArsCtrl->Logoff( );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.

```

Visual Basic Example

```

Dim rc As Integer
.
.

rc = ArsOle.Logoff ( )
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

.
.

```

Logon

Method:

```
short Logon(  
    char * pServerName,  
    char * pUserId,  
    char * pPassword )
```

Parameters:

pServerName

Points to a null-terminated character string containing the name of the server.

pUserId

Points to a null-terminated character string containing the user id.

pPassword

Points to a null-terminated character string containing the password.

Description:

If each of pServerName, pUserId, and pPassword are non-NULL and point to a string other than an empty string, a logon is performed with the specified data.

If pServerName and pUserId point to meaningful strings and pPassword points to a single space character, the logon will proceed with no password for the user.

If any of pServerName, pUserId, and pPassword are NULL or point to an empty string, the normal Content Manager OnDemand Logon dialog box is displayed with the partial information provided. The user can then enter the missing information and complete the logon.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

GetNumServers, GetServerNames, Logoff, SetLogonReturnOnFailure

Example:

The following example retrieves the names of all servers available for logon, puts them in a ComboBox control, retrieves the chosen server, userid, and password, and performs a logon.

C/C++ Example

```
CArsOle * pArsCtrl;  
ArsOleName * pServerNames;  
CComboBox * pServersList;  
CEdit * pUserId;  
CEdit * pPassword;  
char server[ sizeof( ArsOleName ) ];  
char user[ sizeof( ArsOleName ) ];  
char password[ sizeof( ArsOleName ) ];  
short rc, j, num_servers;  
int index;  
VARIANT vari;  
:  
:  
// During dialog initialization:  
  
rc = pArsCtrl->GetNumServers( &vari );
```

```

if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_servers = var.iVal;

pServerNames = new ArsOleName[ max( num_servers, 1 ) ];
rc = pArsCtrl->GetServerNames( (IUnknown*)pServerNames, num_servers );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

for ( j = 0; j < num_servers; j++ )
    index = pServersList->AddString( pServerNames[j] );
pServersList->SetCurSel( 0 );
.
.
// During OK button processing:

pServersList->GetWindowText( server, sizeof(server) );
pUserId->GetWindowText( user, sizeof(user) );
pPassword->GetWindowText( password, sizeof(password) );

rc = pArsCtrl->Logon( server, user, password );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.

```

Visual Basic Example

```

Dim rc, count As Integer
Dim num_servers As Variant
Dim Temp As String

.
.

rc = ArsOle.GetNumServers (num_servers)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

For count = 0 To num_servers -1
    rc = ArsOle.GetServerName(count, Temp)
    lbServers.AddItem Temp
Next count

.
.

' During OK button processing

rc = ArsOle.Logon (lbServers.List(lbServers.ListItem), txtUser.Value, txtPasswd.Value)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

```

OnSysColorChange

Note: This method intended for use with C/C++.

Method:

```
void OnSysColorChange( )
```

Description:

Colors used for Content Manager OnDemand dialog boxes are updated to the current system colors. The colors used for the document data are not affected.

This method should be called whenever the main window for the OLE container application receives a WM_SYSCOLORCHANGE message.

Return Value:

None.

Example:

The following example notifies an Content Manager OnDemand OLE Control when a system color change has occurred.

C/C++ Example

```
CArsOle * pArsCtrl;  
.  
.  
// During WM_SYSCOLORCHANGE message handling:  
  
pArsCtrl->OnSysColorChange( );  
.  
.
```

OpenDoc

Method:

```
short OpenDoc(  
    long Index,  
    char * pPath,  
    long ControlId )
```

Parameters:

Index

Specifies the zero-based index of a document within the document list of the active folder.

pPath

Points to a null-terminated character string containing the fully-qualified path of a file containing the document data. If this parameter is NULL, the document data is retrieved from the Content Manager OnDemand database; if not NULL, the data will be taken from the specified file, but a resource group will be retrieved from the database if required.

ControlId

Specifies the control id of an Content Manager OnDemand OLE Control. If the value is zero, the control id of this control is used. Refer to "Viewing multiple documents for a single folder" on page 3 for a discussion of control ids.

Description:

The document associated with the specified index in the document list of the active folder of the specified Content Manager OnDemand OLE Control is opened and displayed in the window of this control.

The reference to a different Content Manager OnDemand OLE Control allows several windows to simultaneously display documents from a single document list. This avoids the overhead of multiple logon, open folder, and search folder operations. If only one Content Manager OnDemand OLE Control is being used within an application, the ControlId should always be set to zero.

The document retrieval may be cancelled by using the CancelOperation method. If a cancel facility is made available to the user, it may be desirable to call the OpenDoc method on a background thread and allow the user interface thread to monitor the cancellation signal.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

GetNumFolderDisplayFields, GetFolderDisplayFieldNames, GetNumDocsInList, GetDocDisplayValues, CloseDoc, CancelOperation, WasOperationCancelled, ShowWaitCursorDuringCancelableOperation

Example:

The following example creates a list box of the folder document list names and associated values and opens the document selected by a user.

C/C++ Example

```
CArsOle * pArsCtrl;
ArsOleName * pNames;
ArsOleValue * pValues;
CListBox * pDocList;
char * pLine;
short rc, k, opr, num_fields;
long j, num_docs;
int size;
VARIANT vari;
.
.
// During dialog initialization:

rc = pArsCtrl->GetNumFolderDisplayFields( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_fields = vari.iVal;

pNames = new ArsOleName[ max( num_fields, 1 ) ];
rc = pArsCtrl->GetFolderDisplayFieldNames( (IUnknown*)pNames, num_fields );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

rc = pArsCtrl->GetNumDocsInList( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_docs = var.lVal;
.
.
.
pValues = new ArsOleValue[ max( num_fields, 1 ) ];

size = num_fields * ( sizeof(ArsOleName) + sizeof(ArsOleValue) + 5 );
pLine = new char[ size ];
for ( j = 0, pLine[0] = '\0'; j < num_docs; j++ )
{
    rc = pArsCtrl->GetDocDisplayValues( j, pValues, num_fields );
    if ( rc != ARS_OLE_RC_SUCCESS )
        ERROR;

    for ( k = 0; k < num_fields; k++ )
    {
        strcat( pLine, pNames[k] );
        strcat( pLine, " = " );
        strcat( pLine, pValues[k] );
        if ( k < num_fields - 1 )
```

```

        strcat( pLine, ", " );
    }
    pDocList->InsertString( -1, pLine );
}
pDocList->SetCurSel( 0 );
.
.
// During OK button processing:

rc = pArsCtrl->OpenDoc( (long)pDocList->GetCurSel( ) , NULL, 0 );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.

```

Visual Basic Example

```

Dim rc, count, i, j As Integer
Dim num_fields, num_docs As Variant
Dim Names() As String
Dim Line As String
Dim Temp As String

.
.

rc = ArsOle.GetNumFolderDisplayFields(num_fields)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

ReDim Names(num_fields -1)

For count = 0 To num_fields -1
    rc = ArsOle.GetFolderDisplayFieldName(count, Temp)
    Names(count) = Temp
Next count

rc = ArsOle.GetNumDocsInList(num_docs)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

For j = 0 To num_docs -1
    For i = 0 To num_fields -1
        rc = ArsOle.GetDocDisplayValue(j, i, Temp)

        Line = Line + Names(i) + " = " + Temp
        If i < num_fields Then
            Line = Line + ", "
        End If
    Next i

    lbDocs.AddItem Line
Next j

.
.

'During OK button processing:

rc = ArsOle.OpenDoc (lbDocs.List(lbDocs.ListIndex), "", 0)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

```

OpenFolder

Method:

```
short OpenFolder(  
    char * pFolderName )
```

Parameters:

pFolderName

Points to a null-terminated character string containing the name of the folder.

Description:

If pFolderName is non-NULL and points to a string other than an empty string, the named folder is opened.

If pFolderName is NULL or points to an empty string, the normal Content Manager OnDemand Open Folder dialog box is displayed. The user can then select the folder and complete the open.

The opened folder becomes the active folder. The Content Manager OnDemand Folder dialog box is initially hidden. It can be displayed by using the ShowFolder method.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

GetNumFolders, GetFolderNames, CloseFolder, CloseAllFolders

Example:

The following example retrieves the names of all folders available for the current server, puts them in a ComboBox control, retrieves the chosen folder, and performs an open for that folder.

C/C + + Example

```
CArsOle * pArsCtrl;  
ArsOleName * pFolderNames;  
CComboBox * pFoldersList;  
char folder[ sizeof( ArsOleName ) ];  
short rc, j, num_folders;  
int index;  
VARIANT vari;  
.  
.  
// During dialog initialization:  
  
rc = pArsCtrl->GetNumFolders( &vari );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
num_folders = var.iVal;  
  
pFolderNames = new ArsOleName[ max( num_folders, 1 ) ];  
rc = pArsCtrl->GetFolderNames( (IUnknown*)pFolderNames, num_folders );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
for ( j = 0; j < num_folders; j++ )  
    index = pFoldersList->AddString( pFolderNames[j] );  
pFoldersList->SetCurSel( 0 );  
.  
.  
// During OK button processing:  
  
pFoldersList->GetWindowText( folder, sizeof(folder) );
```

```

rc = pArsCtrl->OpenFolder( folder );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.

```

Visual Basic Example

```

Dim rc, count As Integer
Dim num_folders As Variant
Dim Temp As String

.
.

rc = ArsOle.GetNumFolders (num_folders)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

For count = 0 To num_folders -1
    rc = ArsOle.GetFolderName(count, Temp)
    lbFolders.AddItem Temp
Next count

.
.

' During OK button processing

rc = ArsOle.OpenFolder (lbFolders.List(lbFolders.ListItem))
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

```

PrintDoc

Method:

```

short PrintDoc(
    long Index,
    long page,
    char * pPrinterName,
    boolean LocalPrinter,
    short Copies,
    short Orientation,
    float TopMargin,
    float BottomMargin,
    float LeftMargin,
    float RightMargin,
    boolean MarginsInMillimeters )

```

Parameters:

Index

Specifies the zero-based index of a document within the document list of the active folder. If this value is less than zero, the open document is printed. If a local printer is specified, only the open document may be printed.

page

Specifies the page number to be printed. If this parameter is less than or equal to zero, the entire document is printed. If a server printer is specified, this parameter is ignored and the entire document is printed.

pPrinterName

Points to a null-terminated character string containing the name of a local or server printer. For local printers, the name should be the same as that typically displayed in a printer selection list (for example, IBM Laser Printer 4019 on LPT1:). For server printers, the name should be one of those defined for the current server in the Content Manager OnDemand database.

LocalPrinter

If non-zero, indicates that the name specified for pPrinterName is a local printer; if zero, that it specifies a server printer. If local, only the open document may be printed. If server, the orientation and margin values are ignored and the entire document is printed.

Copies

Specifies the number of copies to be printed. This value must be between 1 and 100.

Orientation

Specifies the page orientation. This must be one of the following orientation values found in ARSOLEEX.H:

```
ARS_OLE_ORIENTATION_PORTRAIT  
ARS_OLE_ORIENTATION_LANDSCAPE  
ARS_OLE_ORIENTATION_BEST_FIT  
ARS_OLE_ORIENTATION_ASIS
```

This parameter is ignored if a server printer is specified.

TopMargin

Specifies the amount of space for the top page margin. This parameter is ignored if a server printer is specified.

BottomMargin

Specifies the amount of space for the bottom page margin. This parameter is ignored if a server printer is specified.

LeftMargin

Specifies the amount of space for the left page margin. This parameter is ignored if a server printer is specified.

RightMargin

Specifies the amount of space for the right page margin. This parameter is ignored if a server printer is specified.

MarginsInMillimeters

If non-zero, indicates that the margin values are specified in millimeters; if zero, that they are specified in inches. This parameter is ignored if a server printer is specified.

Description:

The page(s) of the specified document are printed.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

GetNumDocsInList, OpenDoc

Example:

The following example prints page 5 of the open document on a local printer.

C/C++ Example

```
CArsOle * pArsCtrl;
short rc;
.
.
rc = pArsCtrl->PrintDoc( -1,
                        5,
                        "Acrobat PDFWriter on DISK:",
                        TRUE,
                        1,
                        ARS_OLE_ORIENTATION_BEST_FIT,
                        0.5,
                        0.5,
                        0.5,
                        0.5,
                        FALSE );

if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.
```

Visual Basic Example

```
Dim rc As Integer
.
.

rc = ArsOle.PrintDoc (-1,
                    5,
                    "Acrobat PDFWriter on DISK:",
                    True,
                    1,
                    ARS_OLE_ORIENTATION_BEST_FIT,
                    0.5,
                    0.5,
                    0.5,
                    0.5,
                    False)

If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

.
.
```

RetrieveDoc

Method:

```
short RetrieveDoc(
    long Index,
    char * pDocPath,
    char * pResGrpPath,
```

char * pCombinedPath)

Parameters:

Index

Specifies the zero-based index of a document within the document list of the active folder.

pDocPath

Points to a null-terminated character string containing the fully-qualified path of a file to contain the document data. If this parameter is NULL, no data is written.

pResGrpPath

Points to a null-terminated character string containing the fully-qualified path of a file to contain the resource group data. If this parameter is NULL, no data is written.

pCombinedPath

Points to a null-terminated character string containing the fully-qualified path of a file to contain the combined resource group and document data. If this parameter is NULL, no data is written.

Description:

The data for the indicated document and its associated resource group, if any, are retrieved from the Content Manager OnDemand database and written to the indicated files. If any of the files already exist, the data is appended to the files. In the combined file, the data for the document is placed after the data for the resource group. Any combination of pDocPath, pResGrpPath, and pCombinedPath may be specified.

The document retrieval may be cancelled by using the CancelOperation method. If a cancel facility is made available to the user, it may be desirable to call the RetrieveDoc method on a background thread and allow the user interface thread to monitor the cancellation signal.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

GetNumDocsInList, SetResourceCacheMode

Example:

The following example retrieves the data for all files in a document list, copying all document data to one file and the resource group data to a different file.

C/C++ Example

```
CArsOle * pArsCtrl;
long num_docs, doc_num;
short rc;
.
.
rc = pArsCtrl->GetNumDocsInList( &num_docs );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

for ( doc_num = 0; doc_num < num_docs; doc_num++ )
{
    rc = pArsCtrl->RetrieveDoc( doc_num,
                              "C:\\FILES\\DATA.DOC",
                              doc_num == 0 ? "C:\\FILES\\DATA.RG" : NULL,
```

```

        NULL );
    if ( rc != ARS_OLE_RC_SUCCESS )
        ERROR;
}
.
.

Visual Basic Example

Dim rc, count As Integer
Dim num_docs As Variant

.
.

rc = ArsOle.GetNumDocsInList (num_docs)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

For count = 0 To num_docs - 1
    If count = 0 Then
        rc = ArsOle.RetrieveDoc (count, _
                                "C:\FILES\DATA.DOC", _
                                "C:\FILES\DATA.RG", _
                                "")
        If rc <> ARS_OLE_RC_SUCCESS Then
            MsgBox "ERROR"
        End
        End If
    Else
        rc = ArsOle.RetrieveDoc (count, _
                                "C:\FILES\DATA.DOC", _
                                "", _
                                "")
        If rc <> ARS_OLE_RC_SUCCESS Then
            MsgBox "ERROR"
        End
        End If
    End If
End If
Next count

.
.

```

ScrollDocHorz

Method:

```

short ScrollDocHorz(
    short Type,
    VARIANT * pPosition )

```

Parameters:

Type

Specifies a scroll bar code that identifies the type of scrolling required. This must be one of the following scroll types found in ARSOLEEX.H:

```

ARS_OLE_SCROLL_LINELEFT
ARS_OLE_SCROLL_LINERIGHT
ARS_OLE_SCROLL_PAGELEFT
ARS_OLE_SCROLL_PAGERIGHT
ARS_OLE_SCROLL_LEFT

```

```

ARS_OLE_SCROLL_RIGHT
ARS_OLE_SCROLL_THUMBPOSITION
ARS_OLE_SCROLL_THUMBTRACK
ARS_OLE_SCROLL_ENDSCROLL

```

pPosition

Points to a variable that contains and/or will contain the scroll position. When Type is ARS_OLE_SCROLL_THUMBPOSITION or ARS_OLE_SCROLL_THUMBTRACK, the variable must contain the position to which to scroll. For all types, on return this variable contains the current scroll position. This variable should be set to type VT_I2.

Description:

The document data is horizontally scrolled as indicated by Type and the current scroll position is returned in the variable pointed to by pPosition. On input and on return, the position value assumes that the horizontal scroll range has been set to ARS_OLE_SCROLL_RANGE. If a different value is used, the units should be converted before and after the call.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

IsDocHorzScrollRequired, ScrollDocVert

Example:

The following example initializes the horizontal scroll bar range, shows or hides the scroll bar after a document is opened or the zoom value is changed, and processes WM_HSCROLL messages.

C/C++ Example

```

CArsOle * pArsCtrl;
CScrollBar * pHorzScrollBar;
short rc, scroll_code;
VARIANT scroll_position, required;
.
.
// During initialization:

pHorzScrollBar->SetScrollRange( 0, ARS_OLE_SCROLL_RANGE );
pHorzScrollBar->ShowScrollBar( FALSE );
.
.
// After a document is opened or changing the zoom value:

rc = pArsCtrl->IsDocHorzScrollRequired( &required );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

pHorzScrollBar->ShowScrollBar( required.iVal );
.
.
// While processing a WM_HSCROLL message:

scroll_code = (short)LOWORD(wParam);

switch ( scroll_code )
{
    case SB_LINELEFT:
        scroll_code = ARS_OLE_SCROLL_LINELEFT;
        break;
    case SB_LINERIGHT:
        scroll_code = ARS_OLE_SCROLL_LINERIGHT;
        break;
}

```

```

case SB_PAGELEFT:
    scroll_code = ARS_OLE_SCROLL_PAGELEFT;
    break;
case SB_PAGERIGHT:
    scroll_code = ARS_OLE_SCROLL_PAGERIGHT;
    break;
case SB_LEFT:
    scroll_code = ARS_OLE_SCROLL_LEFT;
    break;
case SB_RIGHT:
    scroll_code = ARS_OLE_SCROLL_RIGHT;
    break;
case SB_THUMBPOSITION:
    scroll_code = ARS_OLE_SCROLL_THUMBPOSITION;
    break;
case SB_THUMBTRACK:
    scroll_code = ARS_OLE_SCROLL_THUMBTRACK;
    break;
default:
    scroll_code = ARS_OLE_SCROLL_ENDSCROLL;
}

if ( scroll_code == (short)ARS_OLE_SCROLL_THUMBPOSITION ||
    scroll_code == (short)ARS_OLE_SCROLL_THUMBTRACK )
{
    scroll_position.vt = VT_I2;
    scroll_position.iVal = (short)HIWORD(wParam);
}

rc = pArsCtrl->ScrollDocHorz( scroll_code, &scroll_position );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

pHorzScrollBar->SetScrollPos( (int)scroll_position.iVal );
.
.

```

Visual Basic Example

```

Dim rc As Integer
Dim scroll_pos, required As Variant
.
.
' During initialization

sbHorz.Min = 0
sbHorz.Max = ARS_OLE_SCROLL_RANGE
sbHorz.Visible = False

' After a document is opened or changing the zoom value

rc = ArsOle.IsDocHorzScrollRequired (required)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

If required <> 0 Then
    sbHorz.Visible = True
End If

.
.

' During scroll bar Change method

Private Sub sbHorz_Change()

```

```

Dim Diff As Integer
Dim rc As Integer
Dim ScrollCode As Integer
Dim NewPos As Variant

NewPos = 0
Diff = sbHorz.Value - HorzScroll101d
If Diff = sbHorz.LargeChange Then
    ScrollCode = ARS_OLE_SCROLL_PAGEDOWN
    rc = Ars01e.ScrollDocHorz(ScrollCode, NewPos)
    HorzScroll101d = NewPos
    sbHorz.Value = NewPos
ElseIf Diff = -sbHorz.LargeChange Then
    ScrollCode = ARS_OLE_SCROLL_PAGEUP
    rc = Ars01e.ScrollDocHorz(ScrollCode, NewPos)
    HorzScroll101d = NewPos
    sbHorz.Value = NewPos
ElseIf Diff = sbHorz.SmallChange Then
    ScrollCode = ARS_OLE_SCROLL_LINEDOWN
    rc = Ars01e.ScrollDocHorz(ScrollCode, NewPos)
    HorzScroll101d = NewPos
    sbHorz.Value = NewPos
ElseIf Diff = -sbHorz.SmallChange Then
    ScrollCode = ARS_OLE_SCROLL_LINEUP
    rc = Ars01e.ScrollDocHorz(ScrollCode, NewPos)
    HorzScroll101d = NewPos
    sbHorz.Value = NewPos
Else
    ScrollCode = ARS_OLE_SCROLL_THUMBPOSITION
    NewPos = sbHorz.Value
    rc = Ars01e.ScrollDocHorz(ScrollCode, NewPos)
    HorzScroll101d = sbHorz.Value
End If
End Sub

```

ScrollDocVert

Method:

```

short ScrollDocVert(
    short Type,
    VARIANT * pPosition )

```

Parameters:

Type

Specifies a scroll bar code that identifies the type of scrolling required. This must be one of the following scroll types found in ARSOLEEX.H:

```

ARS_OLE_SCROLL_LINEUP
ARS_OLE_SCROLL_LINEDOWN
ARS_OLE_SCROLL_PAGEUP
ARS_OLE_SCROLL_PAGEDOWN
ARS_OLE_SCROLL_TOP
ARS_OLE_SCROLL_BOTTOM
ARS_OLE_SCROLL_THUMBPOSITION
ARS_OLE_SCROLL_THUMBTRACK
ARS_OLE_SCROLL_ENDSCROLL

```

pPosition

Points to a variable that contains and/or will contain the scroll position. When **Type** is ARS_OLE_SCROLL_THUMBPOSITION or ARS_OLE_SCROLL_THUMBTRACK, the variable must contain the position to which to scroll. For all types, on return this variable contains the current scroll position. This variable should be set to type VT_I2.

Description:

The document data is vertically scrolled as indicated by Type and the current scroll position is returned in the variable pointed to by pPosition. On input and on return, the position value assumes that the vertical scroll range has been set to ARS_OLE_SCROLL_RANGE. If a different value is used, the units should be converted before and after the call.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

ScrollDocHorz

Example:

The following example initializes the vertical scroll bar range and processes WM_VSCROLL messages.

C/C + + Example

```
CArsOle * pArsCtrl;
CScrollBar * pVertScrollBar;
short rc, scroll_code;
VARIANT scroll_position;
.
.
// During initialization:

pVertScrollBar->SetScrollRange( 0, ARS_OLE_SCROLL_RANGE );
pVertScrollBar->ShowScrollBar( TRUE );

// While processing a WM_VSCROLL message:

scroll_code = (short)LOWORD(wParam);

switch ( scroll_code )
{
    case SB_LINEUP:
        scroll_code = ARS_OLE_SCROLL_LINEUP;
        break;
    case SB_LINEDOWN:
        scroll_code = ARS_OLE_SCROLL_LINEDOWN;
        break;
    case SB_PAGEUP:
        scroll_code = ARS_OLE_SCROLL_PAGEUP;
        break;
    case SB_PAGEDOWN:
        scroll_code = ARS_OLE_SCROLL_PAGEDOWN;
        break;
    case SB_TOP:
        scroll_code = ARS_OLE_SCROLL_TOP;
        break;
    case SB_BOTTOM:
        scroll_code = ARS_OLE_SCROLL_BOTTOM;
        break;
    case SB_THUMBPOSITION:
        scroll_code = ARS_OLE_SCROLL_THUMBPOSITION;
        break;
    case SB_THUMBTRACK:
        scroll_code = ARS_OLE_SCROLL_THUMBTRACK;
        break;
    default:
        scroll_code = ARS_OLE_SCROLL_ENDSCROLL;
}
.
.
```

```

        .
        .
    if ( scroll_code == (short)ARS_OLE_SCROLL_THUMBPOSITION ||
        scroll_code == (short)ARS_OLE_SCROLL_THUMBTRACK )
    {
        scroll_position.vt = VT_I2;
        scroll_position.iVal = (short)HIWORD(wParam);
    }

    rc = pArsCtrl->ScrollDocVert( scroll_code, &scroll_position );
    if ( rc != ARS_OLE_RC_SUCCESS )
        ERROR;

    pVertScrollBar->SetScrollPos( (int)scroll_position.iVal );
    .
    .

```

Visual Basic Example

```

Dim rc As Integer
Dim scroll_pos, required As Variant

```

```

        .
        .

' During initialization

sbVert.Min = 0
sbVert.Max = ARS_OLE_SCROLL_RANGE
sbVert.Visible = True

        .
        .

' During scroll bar Change method

Private Sub sbVert_Change()
    Dim Diff As Integer
    Dim rc As Integer
    Dim ScrollCode As Integer
    Dim NewPos As Variant
    NewPos = 0
    Diff = sbVert.Value - VertScroll101d
    If Diff = sbVert.LargeChange Then
        ScrollCode = ARS_OLE_SCROLL_PAGEDOWN
        rc = Ars01e.ScrollDocVert(ScrollCode, NewPos)
        VertScroll101d = NewPos
        sbVert.Value = NewPos
    ElseIf Diff = -sbVert.LargeChange Then
        ScrollCode = ARS_OLE_SCROLL_PAGEUP
        rc = Ars01e.ScrollDocVert(ScrollCode, NewPos)
        VertScroll101d = NewPos
        sbVert.Value = NewPos
    ElseIf Diff = sbVert.SmallChange Then
        ScrollCode = ARS_OLE_SCROLL_LINEDOWN
        rc = Ars01e.ScrollDocVert(ScrollCode, NewPos)
        VertScroll101d = NewPos
        sbVert.Value = NewPos
    ElseIf Diff = -sbVert.SmallChange Then
        ScrollCode = ARS_OLE_SCROLL_LINEUP
        rc = Ars01e.ScrollDocVert(ScrollCode, NewPos)
        VertScroll101d = NewPos
        sbVert.Value = NewPos
    Else
        ScrollCode = ARS_OLE_SCROLL_THUMBPOSITION
        NewPos = sbVert.Value
    End If
End Sub

```

```

        rc = ArsOle.ScrollDocVert(ScrollCode, NewPos)
        VertScroll101d = sbVert.Value
    End If
End Sub

```

SearchFolder

Method:

```

short SearchFolder(
    boolean Append )

```

Parameters:

Append

If non-zero, indicates that the results of the search are to be appended to the existing document list; otherwise, that the results of the search are to replace the existing document list.

Description:

The active folder is searched with the current field values. These values were set by default, by the user, or by the SetFolderSearchFieldData method.

The search operation may be cancelled by using the CancelOperation method. If a cancel facility is made available to the user, it may be desirable to call the SearchFolder method on a background thread and allow the user interface thread to monitor the cancellation signal.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

OpenFolder, GetNumFolderSearchFields, GetFolderSearchFieldNames, SetFolderSearchFieldData, CancelOperation, WasOperationCancelled, ShowWaitCursorDuringCancelableOperation

Example:

The following example retrieves the names of the active folder search fields, gives a user the opportunity to set the values for these fields, and initiates a search of the folder.

C/C++ Example

```

CArsOle * pArsCtrl;
ArsOleName * pNames;
CListBox * pFieldList, * pOprList;
CEdit * pValue1, * pValue2;
char name[ sizeof( ArsOleName ) ];
char value1[ sizeof( ArsOleValue ) ];
char value2[ sizeof( ArsOleValue ) ];
short rc, j, opr, num_fields;
VARIANT vari;
.
.
struct _OprMap
{
    short code;
    char * pText;
} OprMap

static OprMap Oprs[] =
{ { ARS_OLE_OPR_EQUAL, "Equal" },
  { ARS_OLE_OPR_NOT_EQUAL, "Not Equal" },
  .

```

```

        { ARS_OLE_OPR_LIKE,                "Like" },
        { ARS_OLE_OPR_NOT_LIKE,            "Not Like" } };

#define NUM_OPRS ( sizeof(Oprs) / sizeof(OprMap) )
.
.
// During dialog initialization:

rc = pArsCtrl->GetNumFolderSearchFields( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_fields = var.iVal;

pNames = new ArsOleName[ max( num_fields, 1 ) ];
rc = pArsCtrl->GetFolderSearchFieldNames( (IUnknown*)pNames, num_fields );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

for ( j = 0; j < num_fields; j++ )
    pFieldList->InsertString( -1, pNames[j] );
pFieldList->SetCurSel( 0 );

for ( j = 0; j < NUM_OPRS; j++ )
{
    pOprList->InsertString( -1, Oprs[j].pText );
    pOprList->SetItemData( j, (DWORD)Oprs[j].code );
}
pOprList->SetCurSel( 0 );
.
.
// During SET FIELD button processing:

pFieldList->GetText( pFieldList->GetCurSel( ), name );
opr = (short)pOprList->GetItemData( pOprList->GetCurSel( ) );
pValue1->GetWindowText( value1, sizeof(value1) );
pValue2->GetWindowText( value2, sizeof(value2) );

rc = pArsCtrl->SetFolderSearchFieldData( name, opr, value1, value2 );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.
// During OK button processing:

rc = pArsCtrl->SearchFolder( FALSE );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.

```

Visual Basic Example

```

Dim rc, count, i, j As Integer
Dim num_fields, num_docs As Variant
Dim Names() As String
Dim Line As String
Dim Temp As String
Dim Oprs As Variant

.
.

Oprs = Array ("Equal", "Not Equal", ..., "Like", "Not Like")

rc = ArsOle.GetNumFolderSearchFields(num_fields)
If rc <> ARS_OLE_RC_SUCCESS Then

```

```

        MsgBox "ERROR"
    End
End If

ReDim Names(num_fields -1)

For count = 0 To num_fields -1
    rc = ArsOle.GetFolderSearchFieldName(count, Temp)
    Names(count) = Temp
Next count

for count = 0 To num_fields -1
    lbFieldList.AddItem Names(count)
Next count

for count = 0 To UBound(Oprs) -1
    lbOprList.AddItem (Oprs(count))
Next count
' During SET FIELD button processing
rc = ArsOle.SetFolderSearchFieldData (lbFieldList.List(lbFieldList.ListIndex),
                                     lbOprList.ListIndex,
                                     txtValue1.Value,
                                     txtValue2.Value)

If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If
.
.
.
'During OK button processing:

rc = ArsOle.SearchFolder (False)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

```

SetDefaultFolderSearchFields

Method:

short SetDefaultFolderSearchFields()

Description:

The search fields for the active folder are set to their default values.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

OpenFolder, SearchFolder

Example:

The following example sets the search fields for the active folder to their default values.

C/C++ Example

```

CArsOle * pArsCtrl;
short rc;
.
.
rc = pArsCtrl->SetDefaultFolderSearchFields( );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.

```

Visual Basic Example

```
Dim rc As Integer

.
.

rc = ArsOle.SetDefaultFolderSearchFields ()
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

.
.
```

SetDocBackgroundColor

Method:

```
short SetDocBackgroundColor(
    short Color )
```

Parameters:

Color

Specifies the new document background color. This must be one of the following color values found in ARSOLEEX.H:

```
ARS_OLE_COLOR_WHITE
ARS_OLE_COLOR_BLACK
ARS_OLE_COLOR_RED
ARS_OLE_COLOR_BLUE
ARS_OLE_COLOR_GREEN
ARS_OLE_COLOR_YELLOW
ARS_OLE_COLOR_GREY
```

Description:

The document is displayed with the new background color.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

SetDocBackgroundColor

Example:

The following example sets the document background color to grey.

C/C++ Example

```
CArsOle * pArsCtrl;
short rc;

.
.

rc = pArsCtrl->SetDocBackgroundColor( ARS_OLE_COLOR_GREY );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

.
.
```

Visual Basic Example

```
Dim rc As Integer

.
.

rc = ArsOle.SetDocBackgroundColor (ARS_OLE_COLOR_GREY)
```

```

If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

.
.

```

SetDocCurrentPage

Method:

```

short SetDocCurrentPage(
    long page )

```

Parameters:

page

Specifies the page number to be made the current page number of the open document.

Description:

The current page number of the open document is set to the specified page and the control window is repainted with the data for that page.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

GetDocCurrentPage, GetDocNumPages

Example:

The following example sets the current page number of the open document and updates the current scroll positions.

C/C++ Example

```

CArsOle * pArsCtrl;
CScrollBar * pHorzScrollBar, * pVertScrollBar;
short rc;
VARIANT horz_position, vert_position;

.
.
rc = pArsCtrl->SetDocCurrentPage( 46 );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

.
.

rc = pArsCtrl->GetDocScrollPositions( &horz_position, &vert_position );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

pHorzScrollBar->SetScrollPos( (int)horz_position.iVal );
pVertScrollBar->SetScrollPos( (int)vert_position.iVal );

.
.

```

Visual Basic Example

```

Dim rc As Integer
Dim horz_pos, vert_pos As Variant

.
.

rc = ArsOle.SetDocCurrentPage( 46 )
If rc <> ARS_OLE_RC_SUCCESS Then

```

```

        MsgBox "ERROR"
    End
End If

rc = ArsOle.GetDocScrollPositions( horz_pos, vert_pos )
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

sbHorz.Value = horz_pos
sbVert.Value = vert_pos

.
.

```

SetDocImageColor

Method:

```

short SetDocImageColor(
    short Color )

```

Parameters:

Color

Specifies the new document image color. This must be one of the following color values found in ARSOLEEX.H:

```

ARS_OLE_COLOR_BLACK
ARS_OLE_COLOR_RED
ARS_OLE_COLOR_BLUE
ARS_OLE_COLOR_GREEN
ARS_OLE_COLOR_YELLOW
ARS_OLE_COLOR_GREY
ARS_OLE_COLOR_MAGENTA
ARS_OLE_COLOR_CYAN

```

Description:

The document is displayed with the new image color.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

GetDocImageColor

Example:

The following example sets the document image color to red.

C/C++ Example

```

CArsOle * pArsCtrl;
short rc;

.
.
rc = pArsCtrl->SetDocImageColor( ARS_OLE_COLOR_RED );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.

```

Visual Basic Example

```

Dim rc As Integer

.
.

```

```

rc = ArsOle.SetDocImageColor (ARS_OLE_COLOR_RED)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

.
.

```

SetDocImageIntensity

Method:

```

short SetDocImageIntensity(
    short Intensity )

```

Parameters:

Intensity

Specifies the new document image intensity. This must be one of the following intensity values found in ARSOLEEX.H:

```

ARS_OLE_INTENSITY_NORMAL
ARS_OLE_INTENSITY_LIGHT
ARS_OLE_INTENSITY_NONE

```

Description:

The document is displayed with the new image intensity.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

GetDocImageIntensity

Example:

The following example sets the document image intensity to light.

C/C + + Example

```

CArsOle * pArsCtrl;
short rc;

.
.
rc = pArsCtrl->SetDocImageIntensity( ARS_OLE_INTENSITY_LIGHT );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

.
.

```

Visual Basic Example

```

Dim rc As Integer

.
.

rc = ArsOle.SetDocImageIntensity (ARS_OLE_INTENSITY_LIGHT)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

.
.

```

SetDocRotation

Method:

```
short SetDocRotation(  
    short Rotation )
```

Parameters:

Rotation

Specifies the new document rotation. This must be one of the following rotation values found in ARSOLEEX.H:

```
ARS_OLE_ROTATION_0  
ARS_OLE_ROTATION_90  
ARS_OLE_ROTATION_180  
ARS_OLE_ROTATION_270
```

Description:

The document is displayed at the new rotation.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

GetDocRotation

Example:

The following example rotates the document to 90 degrees.

C/C + + Example

```
CArsOle * pArsCtrl;  
short rc;  
  
.  
.  
rc = pArsCtrl->SetDocRotation( ARS_OLE_ROTATION_90 );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
.  
.
```

Visual Basic Example

```
Dim rc As Integer  
  
.  
.  
  
rc = ArsOle.SetDocRotation (ARS_OLE_ROTATION_90)  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If  
  
.  
.
```

SetDocZoom

Method:

```
short SetDocZoom(  
    short ZoomPercent,  
    VARIANT * pHorzPosition,  
    VARIANT * pVertPosition )
```

Parameters:**ZoomPercent**

Specifies the new zoom percent to be set.

pHorzPosition

Points to a variable to receive the new horizontal scroll position. On return, this variable is set to type VT_I2.

pVertPosition

Points to a variable to receive the new vertical scroll position. On return, this variable is set to type VT_I2.

Description:

The document is displayed at the new zoom percent and the new scroll positions are returned in the specified variables. The scroll positions assume that the scroll ranges have been set to ARS_OLE_SCROLL_RANGE.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

GetDocZoom

Example:

The following example sets a new zoom value and repositions the scroll bars.

C/C + + Example

```
CArsOle * pArsCtrl;
CScrollBar * pHorzScrollBar, * pVertScrollBar;
short rc;
VARIANT horz_position, vert_position;
.
.
// During initialization:

pHorzScrollBar->SetScrollRange( 0, ARS_OLE_SCROLL_RANGE );
pVertScrollBar->SetScrollRange( 0, ARS_OLE_SCROLL_RANGE );
.
.
// When required to double the zoom factor:

rc = pArsCtrl->SetDocZoom( 200, &horz_position, &vert_position );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

pHorzScrollBar->SetScrollPos( (int)horz_position.iVal );
pVertScrollBar->SetScrollPos( (int)vert_position.iVal );
.
.
```

Visual Basic Example

```
Dim rc As Integer
Dim horz_pos, vert_pos As Variant
.
.
' During initialization

sbHorz.Min = 0
sbHorz.Max = ARS_OLE_SCROLL_RANGE
```

```

sbVert.Min = 0
sbVert.Max = ARS_OLE_SCROLL_RANGE

.
.

' When required to double the zoom factor

rc = Ars01e.SetDocZoom (200, horz_pos, vert_pos)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

sbHorz.Value = horz_pos
sbVert.Value = vert_pos

```

SetFolderCloseMemoryRelease

Method:

```

short SetFolderCloseMemoryRelease(
    boolean Release )

```

Parameters:

Release

If non-zero, indicates that all memory associated with a folder is to be released when the folder is closed; otherwise, that the memory is to be retained.

Description:

Determines whether memory is to be released when a folder is closed. By default, the setting is false.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

CloseFolder

Example:

The following example causes memory to be released when a folder is closed.

C/C + + Example

```

CArs01e * pArsCtrl;
.
.
.
pArsCtrl->SetFolderCloseMemoryRelease( TRUE );
.
.
.

```

Visual Basic Example

```

.
.
.
Ars01e.SetFolderCloseMemoryRelease (True)
.
.
.

```

SetFolderSearchFieldData

Method:

```
short SetFolderSearchFieldData(  
    char * pFieldName,  
    short operator,  
    char * pValue1,  
    char * pValue2 )
```

Parameters:

pFieldName

Points to a null-terminated character string containing the name of a search field for the active folder.

operator

Specifies the search operator to be used. This must be one of the following operator values found in ARSOLEEX.H:

```
ARS_OLE_OPR_EQUAL  
ARS_OLE_OPR_NOT_EQUAL  
ARS_OLE_OPR_LESS_THAN  
ARS_OLE_OPR_LESS_THAN_OR_EQUAL  
ARS_OLE_OPR_GREATER_THAN  
ARS_OLE_OPR_GREATER_THAN_OR_EQUAL  
ARS_OLE_OPR_BETWEEN  
ARS_OLE_OPR_NOT_BETWEEN  
ARS_OLE_OPR_IN  
ARS_OLE_OPR_NOT_IN  
ARS_OLE_OPR_LIKE  
ARS_OLE_OPR_NOT_LIKE
```

pValue1

Points to a null-terminated character string containing the first, or only, value to be set for the field.

pValue2

Points to a null-terminated character string containing the second value to be set for the field. This parameter is ignored unless the operator is ARS_OLE_OPR_BETWEEN or ARS_OLE_OPR_NOT_BETWEEN.

Description:

The search operator and value(s) are set for the specified field for the active folder.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

GetNumFolderSearchFields, GetFolderSearchFieldNames, SearchFolder

Example:

The following example retrieves the names of the active folder search fields, gives a user the opportunity to set the values for these fields, and initiates a search of the folder.

C/C++ Example

```
CArsOle * pArsCtrl;  
ArsOleName * pNames;  
CListBox * pFieldList, * pOpList;  
CEdit * pValue1, * pValue2;
```

```

char name[ sizeof( ArsOleName ) ];
char value1[ sizeof( ArsOleValue ) ];
char value2[ sizeof( ArsOleValue ) ];
short rc, j, opr, num_fields;
VARIANT vari;
.
.
struct _OprMap
{
    short code;
    char * pText;
} OprMap

static OprMap Oprs[] =
{ { ARS_OLE_OPR_EQUAL,          "Equal" },
  { ARS_OLE_OPR_NOT_EQUAL,      "Not Equal" },
  .
  { ARS_OLE_OPR_LIKE,           "Like" },
  { ARS_OLE_OPR_NOT_LIKE,       "Not Like" } };

#define NUM_OPRS ( sizeof(Oprs) / sizeof(OprMap) )
.
.
// During dialog initialization:

rc = pArsCtrl->GetNumFolderSearchFields( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_fields = var.iVal;

pNames = new ArsOleName[ max( num_fields, 1 ) ];
rc = pArsCtrl->GetFolderSearchFieldNames( (IUnknown*)pNames, num_fields );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

for ( j = 0; j < num_fields; j++ )
    pFieldList->InsertString( -1, pNames[j] );
pFieldList->SetCurSel( 0 );

for ( j = 0; j < NUM_OPRS; j++ )
{
    pOprList->InsertString( -1, Oprs[j].pText );
    pOprList->SetItemData( j, (DWORD)Oprs[j].code );
}
pOprList->SetCurSel( 0 );
.
.
// During SET FIELD button processing:

pFieldList->GetText( pFieldList->GetCurSel( ), name );
opr = (short)pOprList->GetItemData( pOprList->GetCurSel( ) );
pValue1->GetWindowText( value1, sizeof(value1) );
pValue2->GetWindowText( value2, sizeof(value2) );

rc = pArsCtrl->SetFolderSearchFieldData( name, opr, value1, value2 );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.
// During OK button processing:

rc = pArsCtrl->SearchFolder( FALSE );

```

```

if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.

```

Visual Basic Example

```

Dim rc, count, i, j As Integer
Dim num_fields, num_docs As Variant
Dim Names() As String
Dim Line As String
Dim Temp As String
Dim Oprs As Variant

.
.

Oprs = Array ("Equal", "Not Equal", ..., "Like", "Not Like")

rc = ArsOle.GetNumFolderSearchFields(num_fields)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

ReDim Names(num_fields)

For count = 1 To num_fields
    rc = ArsOle.GetFolderSearchFieldName(count, Temp)
    Names(count) = Temp
Next count

for count = 1 To num_fields
    lbFieldList.AddItem Names(count)
Next count

for count = 1 To UBound(Oprs)
    lbOprList.AddItem (Oprs(count))
Next count

.
.
.
' During SET FIELD button processing
rc = ArsOle.SetFolderSearchFieldData (lbFieldList.List(lbFieldList.ListIndex),
                                      lbOprList.ListIndex,
                                      txtValue1.Value,
                                      txtValue2.Value)

If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

'During OK button processing:

rc = ArsOle.SearchFolder (False)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

```

SetLogonReturnOnFailure

Method:

```

short SetLogonReturnOnFailure(
    boolean Return )

```

Parameters:

Return

If non-zero, indicates that control is to be returned when a logon fails; otherwise, that the OnDemand Logon dialog box is to be displayed when a logon fails.

Description:

Determines the action to be taken when a logon fails. By default, the setting is false.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

Logon

Example:

The following example sets the action to be taken when a logon fails.

C/C + + Example

```
CArsOle * pArsCtrl;  
.  
.  
.  
pArsCtrl->SetLogonReturnOnFailure( TRUE );  
.  
.  
.
```

Visual Basic Example

```
.  
.  
.  
ArsOle.SetLogonReturnOnFailure (True)  
.  
.  
.
```

SetResourceCacheMode

Method:

```
short SetResourceCacheMode(  
    short Mode )
```

Parameters:

Mode

Specifies the new resource cache mode. This must be one of the following mode values found in ARSOLEEX.H:

```
ARS_OLE_RES_MODE_RETAIN  
ARS_OLE_RES_MODE_ERASE_AFTER_RETRIEVE
```

ARS_OLE_RES_MODE_RETAIN specifies that any resource group retrieved during execution of a RetrieveDoc method be retained for use by subsequent RetrieveDoc or OpenDoc methods. Files containing these resource groups will not be erased until all instances of the control have been terminated. This is the default mode.

ARS_OLE_RES_MODE_ERASE_AFTER_RETRIEVE specifies that the file for any resource group retrieved during execution of a RetrieveDoc method be erased before returning to the caller. If subsequent RetrieveDoc or OpenDoc methods require access to the same resource group, another retrieval will be required.

Description:

The requested mode is set.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

RetrieveDoc

Example:

The following example sets the resource cache mode to retain.

C/C++ Example

```
CArsOle * pArsCtrl;
short rc;
.
.
rc = pArsCtrl->SetResourceCacheMode( ARS_OLE_RES_MODE_RETAIN );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.
```

Visual Basic Example

```
Dim rc As Integer
.
.
rc = ArsOle.SetResourceCacheMode (ARS_OLE_RES_MODE_RETAIN)
if rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If
.
.
```

SetRightButtonMenu

Method:

```
short SetRightButtonMenu(
    char * pMenuData,
    VARIANT * pErrorPosition )
```

Parameters:**pMenuData**

Specifies the data to be used to create a menu. If this parameter is null, any existing menu is deleted.

pErrorPosition

Points to a variable to receive an error position if the menu cannot be created. In that case, the variable is set to type VT_I4.

Description:

OnDemand creates a menu to be displayed when the user clicks the right mouse button in the document window. This menu replaces any existing menu. The format of the menu is determined by the data specified by the pMenuData parameter. Complex menus containing commands, separators, and submenus can be created.

The menu data consists of a set of menu items separated by a newline character (X'0A'). There can be a maximum of ARS_OLE_MAX_MENU_ITEMS items, each of which can contain no more

than ARS_OLE_MAX_MENU_ITEM_LEN characters. The items must appear in the order in which they are to be displayed in the menu.

Each item is described by keywords and values. A keyword and its associated value must be separated by an equal sign. Each keyword/value pair must be separated by at least one space. The recognized keywords are:

level A number that indicates the nesting level of the item. The first item must be level 0 (zero). Each subsequent nesting level must add 1 (one). The nesting level can be increased only when specifying a popup submenu.

This keyword must be provided.

id The user command number to be associated with the item. The id is the number that will be reported for the UserCommand event when the user chooses that menu item.

Two special ids are defined:

- If id is specified as ARS_OLE_MENU_ID_SEPARATOR, a separator item is created.
- If id is specified as ARS_OLE_MENU_ID_POPUP, a popup submenu is created. In this case, the following item should have a level one greater than the level specified for this item.

Other ids must have a minimum of ARS_OLE_MIN_MENU_ID and a maximum value of ARS_OLE_MAX_MENU_ID.

This keyword must be provided.

enabled

The enabled keyword must have a value of 1 (one) or 0 (zero). If 1 (one), the item is enabled; if 0 (zero), disabled.

This keyword is optional and is ignored for separator items. The default is for the item to be enabled.

checked

The checked keyword must have a value of 1 (one) or 0 (zero). If 1 (one), the item is checked; if 0 (zero), unchecked.

This keyword is optional and is ignored for separator items. The default is for the item to be unchecked.

text The text keyword specifies the text of the item. The text, which may include embedded blanks, must be contained in single quotes. If a single quote is part of the text, two consecutive single quotes must be specified.

The text may contain the normal Windows special characters, such as an & (ampersand) to indicate that the following character is to be underlined.

This keyword is optional and is ignored for separator items.

If the menu data is not valid, an error position is returned via the pErrorPosition parameter. The position is zero-based and is relative to the first character of the menu data. The position will normally identify the first character of the item which contains an error.

The following examples show the use of the keywords.

Data	Menu
level=0 id=368 text='Copy Text'	Copy Text
level=0 id=44 text='Item1'\n	Item1
level=0 id=1\n	
level=0 id=45 text='Item2'	Item2
level=0 id=32457 text='Item1'\n	Item1
level=0 id=0 text='Popup1'\n	Popup1 ▶ Item2
level=1 id=32458 text='Item2'\n	
level=1 id=32459 text='Item3'\n	Item3
level=0 id=1\n	Popup2 ▶
level=0 id=0 text='Popup2'\n	
level=1 id=32460 enabled=0 text='Item4'\n	Item1
level=1 id=1\n	Popup1 ▶
level=1 id=32461 checked=1 text='Item5'	Popup2 ▶ Item4
	✓ Item5

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

UserCommand event

Example:

The following example shows how to create a right button menu that contains the command Copy Text.

C/C + + Example

```

CArsOle * pArsCtrl;
short rc;
VARIANT var;
.
.
.

rc = pArsCtrl->SetRightButtonMenu( "level=0 id=368 text='Copy Text'", var );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

.
.

```

Visual Basic Example

```

Dim rc As Integer
Dim var As Variant

.
.

rc = ArsOle.SetRightButtonMenu ( "level=0 id=368 text='Copy Text'", var)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

```

SetSelectionMode

Method:

```
short SetSelectionMode(  
    short Mode )
```

Parameters:

Mode

Specifies the new selection mode. This must be one of the following selection mode values found in ARSOLEEX.H:

```
ARS_OLE_SELECTION_MODE_NONE  
ARS_OLE_SELECTION_MODE_AREA  
ARS_OLE_SELECTION_MODE_TEXT
```

ARS_OLE_SELECTION_MODE_NONE specifies that the user cannot select a portion of the document with the mouse. This is the default mode when a document is opened. Setting this mode removes any existing selection.

ARS_OLE_SELECTION_MODE_AREA specifies that user selection is performed in the same manner as Options->Selection Mode->Area with the OnDemand client GUI.

ARS_OLE_SELECTION_MODE_TEXT specifies that user selection is performed in the same manner as Options->Selection Mode->Text with the OnDemand client GUI.

Description:

Subsequent mouse selection by the user is performed in the specified mode.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

CopyBitmap, CopyText

Example:

The following example sets the selection mode to area selection.

C/C + + Example

```
CArsOle * pArsCtrl;  
short rc;  
.  
.  
rc = pArsCtrl->SetSelectionMode( ARS_OLE_SELECTION_MODE_AREA );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
.  
.
```

Visual Basic Example

```
Dim rc As Integer  
.  
.  
  
rc = ArsOle.SetSelectionMode (ARS_OLE_SELECTION_MODE_AREA)  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End
```

End If

.
.

SetServerPrinterData

Method:

```
short SetServerPrinterData(  
    short Index,  
    char * pData )
```

Parameters:

Index

Specifies the server printer data index. It must be one of the following values found in ARSLOEEX.H:

```
ARS_OLE_SERVER_PRINTER_DATA_FAX_RECEIVER_NAME  
ARS_OLE_SERVER_PRINTER_DATA_FAX_RECEIVER_COMPANY  
ARS_OLE_SERVER_PRINTER_DATA_FAX_RECEIVER_FAX_NUMBER  
ARS_OLE_SERVER_PRINTER_DATA_FAX_SENDER_NAME  
ARS_OLE_SERVER_PRINTER_DATA_FAX_SENDER_COMPANY  
ARS_OLE_SERVER_PRINTER_DATA_FAX_SENDER_TEL_NUMBER  
ARS_OLE_SERVER_PRINTER_DATA_FAX_SENDER_FAX_NUMBER  
ARS_OLE_SERVER_PRINTER_DATA_FAX_SENDER_COVER_PAGE  
ARS_OLE_SERVER_PRINTER_DATA_FAX_SUBJECT  
ARS_OLE_SERVER_PRINTER_DATA_FAX_NOTES  
ARS_OLE_SERVER_PRINTER_DATA_INFO_FROM  
ARS_OLE_SERVER_PRINTER_DATA_INFO_TO
```

pData

Points to a null-terminated character string containing the data to be associated with the index. This parameter may be null.

Description:

The data associated with each index defaults to the empty string. Any data set using this method is retained until changed. If pData is null, the data is reset to the empty string. When the PrintDoc method is used and a server printer is specified, the type of printer (such as FAX) is determined and the appropriate data is sent along with the document.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

PrintDoc

Example

The following example sets the FAX receiver name for server printers.

C/C++ Example

```
CArsOle * pArsCtrl;  
short rc;  
.  
.  
.  
  
rc = pArsCtrl->SetServerPrinterData(  
    ARS_OLE_SERVER_PRINTER_DATA_FAX_RECEIVER_NAME,  
    "John Doe" );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;
```

```

.
.
.

```

Visual Basic Example

```
Dim rc As Integer
```

```

.
.

```

```

rc = ArsOle.SetServerPrinterData (
    ARS_OLE_SERVER_PRINTER_DATA_FAX_RECEIVER_NAME,
    "John Doe");
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

```

```

.
.

```

SetUserMessageMode

Method:

```

short SetUserMessageMode(
    short Mode )

```

Parameters:

Mode

Specifies the new user message mode. This must be one of the following message mode values found in ARSOLEEX.H:

```

ARS_OLE_USER_MSG_MODE_SHOW
ARS_OLE_USER_MSG_MODE_SUPPRESS

```

ARS_OLE_USER_MSG_MODE_SHOW indicates that all Content Manager OnDemand exception messages resulting from OLE Control operations are to be displayed to the user. ARS_OLE_USER_MSG_MODE_SUPPRESS indicates that all Content Manager OnDemand exception messages resulting from OLE Control operations are to be suppressed.

Description:

Subsequent Content Manager OnDemand exception messages resulting from OLE Control operations are displayed or suppressed as requested.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

Example:

The following example suppresses Content Manager OnDemand exception messages resulting from OLE Control operations.

C/C++ Example

```

CArsOle * pArsCtrl;
short rc;
.
.
rc = pArsCtrl->SetUserMessageMode( ARS_OLE_USER_MSG_MODE_SUPPRESS );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.

```

Visual Basic Example

```
Dim rc As Integer

.  
.  
  
rc = ArsOle.SetUserMessageMode (ARS_OLE_USER_MSG_MODE_SUPPRESS)  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If  
  
.  
.
```

ShowFolder

Method:

```
short ShowFolder(  
    boolean Show,  
    short Left,  
    short Top,  
    short Right,  
    short Bottom )
```

Parameters:

Show

If non-zero, indicates that the Content Manager OnDemand Folder dialog box is to be displayed; otherwise, that it is to be hidden.

Left

Specifies the X coordinate of the upper left point at which the dialog box is to be displayed. If Show is zero, this parameter is ignored. If Show is non-zero and this parameter is less than zero, the dialog box is displayed at its current size and location.

Top

Specifies the Y coordinate of the upper left point at which the dialog box is to be displayed.

Right

Specifies the X coordinate of the lower right point at which the dialog box is to be displayed.

Bottom

Specifies the Y coordinate of the lower right point at which the dialog box is to be displayed.

Description:

The Content Manager OnDemand Folder dialog box is displayed or hidden.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

OpenFolder, ActivateFolder

Example:

The following example shows the Content Manager OnDemand Folder dialog box centered in the screen with a border equal to ten percent of the screen size.

C/C + + Example

```
CArsOle * pArsCtrl;
short rc, left, top, right, bottom;
int screen_width, screen_height;

.
.
screen_width = GetSystemMetrics( SM_CXSCREEN );
screen_height = GetSystemMetrics( SM_CYSCREEN );
left = screen_width / 10;
top = screen_height / 10;
right = screen_width * 8 / 10;
bottom = screen_height * 8 / 10;

rc = pArsCtrl->ShowFolder( TRUE, left, top, right, bottom );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

.
.
```

Visual Basic Example

```
Dim rc As Integer

.
.

rc = ArsOle.ShowFolder (True, _
                        Screen.Width / 10, _
                        Screen.Height / 10, _
                        Screen.Width * 8 / 10, _
                        Screen.Height * 8 / 10)

If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

.
.
```

ShowWaitCursorDuringCancelableOperation

Method:

```
short ShowWaitCursorDuringCancelableOperation(
    boolean Show )
```

Parameters:

Show

If zero, indicates that OnDemand should not display a wait cursor during SearchFolder or OpenDoc operations. If non-zero, indicates that a wait cursor should be displayed. This is the default behavior.

Description:

During cancelable operations, OnDemand normally displays a wait cursor. If an application provides a cancel button or similar facility to allow the user to initiate a cancel, it may be desirable to prevent this display. This allows the application to show its own cursor, such as an hourglass and arrow, to tell the user that a cancel may be requested.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

CancelOperation

Example:

The following example causes a wait cursor not to be displayed during a cancelable operation.

C/C++ Example

```
CArsOle * pArsCtrl;
.
.
.
pArsCtrl->ShowWaitCursorDuringCancelableOperation( FALSE );
.
.
.
```

Visual Basic Example

```
.
.
.
ArsOle.ShowWaitCursorDuringCancelableOperation (False)
.
.
.
```

StoreDoc

Method:

```
short StoreDoc(
    char * DocPath,
    char * ApplGrpName,
    char * ApplName,
    VARIANT * Values )
```

Parameters:**DocPath**

Specifies the fully-qualified path of a file containing the document data to be stored in the OnDemand database. This parameter is required.

ApplGrpName

Specifies the name of an Application Group within the folder. It is the responsibility of the caller to know the Application Group names associated with the active folder. This parameter is required. In order for StoreDoc() to succeed, the following requirements must be met:

- The Application Group's database organization must have the attribute Multiple Loads per Table. This attribute is set in the General/Advanced tab in the Properties window of the Application Group.
- The Application Group's Expiration Type in Storage Management must be set to Expiration Type: Segment. This attribute is set in the Storage Management tab in the Properties window of the Application Group.
- Storage management for the application group needs to be configured to use cache. Cache is required to allow the OnDemand server to build a larger storage object and append to the data in the cache. Then this

| storage object can be migrated to Tivoli Storage Manager (TSM). Storing
| data directly into TSM is not recommended and can cause TSM
| performance issues.

ApplName

Specifies the name of an Application within the specified Application Group. It is the responsibility of the caller to know the Application names associated with the specified Application Group. This parameter is required.

Values

Points to a SafeArray of folder values. Each value is a character string which will be converted to data of the field type (that is, integer, date, and so on).

The number and order of folder fields can be determined by using the GetFolderFieldName(s) methods. For more information on these methods, see "GetFolderFieldName" and "GetFolderFieldNames" in the *Windows Client Customization Guide*.

Any folder fields not specified are given an empty string for string fields or zero for numeric fields. If extraneous fields are specified, they are ignored.

Date fields must be provided in the format required for the field (for example, specifying 02/03/96 is invalid when February 3, 1996 is required).

Description:

OnDemand converts the folder field values to application group fields and stores the data from the specified file in the database as a document associated with the specified Application Group and Application.

If the return code is one of the following:

```
ARS_OLE_RC_INVALID_DATE_FIELD  
ARS_OLE_RC_INVALID_INTEGER_FIELD  
ARS_OLE_RC_INVALID_DECIMAL_FIELD  
ARS_OLE_RC_TOO_MANY_VALUE_CHARS  
ARS_OLE_RC_INVALID_APPLGRP_FIELD_TYPE
```

The GetStoreDocInvalidFieldNum method can be used to determine the folder field which caused the problem.

Return Value:

Refer to "Return Code" in the *Windows Client Customization Guide* for an explanation of the return code.

See Also:

GetNumFolderFields, GetFolderFieldName, GetFolderFieldNames, GetStoreDocInvalidFieldNum

Example:

C/C++ Example

```
CArsOle * pArsCtrl;  
short rc;  
SAFEARRAY * pSA;  
VARIANT var;  
BSTR bstrElement;  
long i;  
.  
.
```

```

pSA = SafeArrayCreateVector(VT_BSTR,0,2);
if ( pSA == NULL )
    ERROR;

bstrElement = SysAllocStringByteLen ("255-546-667",11);
i = 0;
SafeArrayPutElement (pSA, &i,bstrElement);
bstrElement = SysAllocStringByteLen ("06/07/94",8);
i = 1;
SafeArrayPutElement (pSA, &i,bstrElement);

var.vt = VT_ARRAY | VT_BSTR;
var.parray = pSA;

rc = pArsCtrl->StoreDoc("g:\\download \\file.afp",
                        "BKH-CRD",
                        "BKH-CRD",
                        &var);
if (rc != ARS_OLE_RC_SUCCESS)
    ERROR;
.
.
.

```

Visual Basic Example

```
Dim values(2) As String
Dim rc As Integer
.
.
.
values(0) = "255-546-667"
values(1) = "06/07/94"
var = values
rc = ArsOle.StoreDoc ("g:\download \file.afp", _
                    "BKH-CRD", _
                    "BKH-CRD", _
                    var)
if rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If
.
.
.
```

UndoFind

Method:

short UndoFind()

Description:

Highlighting is removed from the current found string.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

FindStringInDoc

Example:

The following example removes highlighting from a found string.

C/C + + Example

```
CArsOle * pArsCtrl;
short rc;
.
.
rc = pArsCtrl->UndoFind( );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.
```

Visual Basic Example

```
Dim rc As Integer
.
.

rc = ArsOle.UndoFind ( )
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If
.
.
```

UpdateDoc

Method:

```
short UpdateDoc(  
    long * DocIndex,  
    char * pFieldName,  
    char * pValue )
```

Parameters:

DocIndex

Specifies zero-based relative document number within the document list of the active folder. This parameter is required.

pFieldName

Specifies the name of a folder field. This parameter is required.

pValue

Specifies the value to be stored in the specified folder field. This value is a character string which will be converted to data of the field type (that is, integer, date and so on).

Date fields must be provided in the format required for the field (for example, specifying 02/03/96 is invalid when February 3, 1996 is required).

This parameter is required.

Description:

Content Manager OnDemand converts the folder field value to an application group field and updates the data from the specified value in the database.

Return Value:

Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

GetNumFolderFields, GetFolderFieldName, GetFolderFieldNames

Example:

The following example updates the "Balance" field of the first document within the document list of the active folder.

C/C++ Example

```
CArsOle * pArsCtrl;  
short rc;  
  
.  
.  
  
rc = pArsCtrl->UpdateDoc( 0,  
                           "Balance",  
                           "123.45" );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
.  
.
```

Visual Basic Example

```

Dim rc As Integer

.
.

rc = ArsOle.UpdateDoc ( 0,
                        "Balance", -
                        "123.45" ) -
if rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

.
.

```

WasOperationCancelled

Method:

```

short WasOperationCancelled(
    VARIANT * pCancelled )

```

Parameters:

pCancelled

Points to a variable to receive the result. On return, this variable is set to type VT_I2.

Description:

If the most recent SearchFolder, OpenDoc, or RetrieveDoc operation was cancelled, the result variable is set to a non-zero value; otherwise, to zero.

Return Value:

Refer to “Return Code” on page 4 for an explanation of the return code.

See Also:

CancelOperation

Example:

The following example searches a folder and determines whether the search was cancelled.

C/C++ Example

```

CArsOle * pArsCtrl;
short rc;
VARIANT cancelled;

.
.

rc = pArsCtrl->SearchFolder( FALSE );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

rc = pArsCtrl->WasOperationCancelled( &cancelled );
if ( cancelled.iVAL )
    CANCELLATION LOGIC;

.
.

```

Visual Basic Example

```

Dim rc As Integer
Dim cancelled As Variant

```

```

.
.

rc = Ars01e.SearchFolder (False)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

rc = Ars01e.WasOperationCancelled (cancelled)
If cancelled <> 0 Then
    CANCELLATION LOGIC
End If

.
.

```

Chapter 3. OLE Events

The following events are fired by an Content Manager OnDemand OLE Control.

FolderSearchCompleted

This event is fired when a folder search has completed. The search may have been initiated

- By the container application invoking the SearchFolder method; or
- By the user if the ShowFolder method was used to display the Content Manager OnDemand Folder dialog box.

FolderClosed

This event is fired when a folder is closed. The close may have been initiated

- By the container application invoking the CloseFolder method;
- By the user if the ShowFolder method was used to display the Content Manager OnDemand Folder dialog box;
- By a logoff; or
- By destruction of the control.

DocOpened

This event is fired when a document is opened. The open may have been initiated

- By the container application invoking the OpenDoc method; or
- By the user if the ShowFolder method was used to display the Content Manager OnDemand Folder dialog box.

DocClosed

This event is fired when a document is closed. The close may have been initiated

- By the container application invoking the CloseDoc method;
- By the folder associated with the document being closed;
- By a logoff; or
- By destruction of the control.

AreaSelected

This event is fired when the user selects an area of the screen. This is possible only if the SetSelectionMode method has been used to set the selection mode to ARS_OLE_SELECTION_MODE_AREA or ARS_OLE_SELECTION_MODE_TEXT.

AreaDeselected

This event is fired when a selection is removed. The removal may have been caused

- By the user double clicking the mouse outside of the selection;
- By the SetSelectionMode method specifying ARS_OLE_SELECTION_MODE_NONE;

- By the document being closed; or
- For certain documents, by changing the page, scroll position, zoom, or other presentation attributes.

UserCommand(long CommandID)

This event is fired when the user clicks the right mouse button and chooses a menu item. The command ID associated with the menu item is passed as a parameter.

See the `SetRightButtonMenu` method for information on creating menus.

Part 2. Windows 32-bit GUI customization guide

Chapter 4. Content Manager OnDemand customization overview

The Content Manager OnDemand application can be customized by specifying command line parameters, by invoking and manipulating it from another application with the Dynamic Data Exchange interface, by invoking other applications and DLL files from the client, by retrieving related documents, by creating a Program Information File, and by auditing documents.

Chapter 5. Command line

This section describes:

- Starting Content Manager OnDemand.
- The parameter syntax rules used for the command line parameters.
- The parameters recognized by the Content Manager OnDemand 32-bit Windows Client.

Starting Content Manager OnDemand 32-bit client

Content Manager OnDemand can be started by double-clicking on the program icon on your desktop.

Parameter Syntax

The following syntax rules apply to command line parameters:

- At least one space character must follow the executable name.
- A parameter is identified by a slash followed immediately by a single character.
- The parameter character is not case sensitive.
- The associated parameter value, if any, begins with the first non-space character following the parameter character and ends with the last non-space character preceding the next parameter or the end of the command line.
- If a slash character is to be included in the parameter value, two consecutive slash characters must be specified.
- Keyword parameter values are not case sensitive.
- Unrecognized parameters are ignored.
- If the same parameter appears more than once, the last occurrence is used.

Parameters

The following parameters are recognized by the Content Manager OnDemand Windows client.

Change Password — /C new password

If this parameter is specified, Content Manager OnDemand changes the password for the user after a successful logon. This parameter can only be specified if the /S, /U, and /P parameters have also been specified.

Example

```
C:\ARS32\ARSGUI32 /S pikes /U user1 /P pass1 /C newpass
```

Code Page Override — /A codepage_number

If this parameter is specified, OnDemand uses the code page number included with the parameter to override the default code page for the user who is logged on in the current session. The code page supplied with this parameter must be a valid code page number of up to five digits.

Example

```
C:\ARS32\ARSGUI32 /A 1392
```

Disable Anticipation — /V

If this parameter is specified, Content Manager OnDemand does not attempt to anticipate the next user request, such as displaying the Logon dialog box after initialization or displaying the Open Folder dialog box after the current folder is closed.

Example

```
C:\ARS32\ARSGUI32 /V
```

Disable Close Folder — /Z

If this parameter is specified, Content Manager OnDemand disables the Close Folder menu item, the Close Folder button on the Search Criteria and Document List dialog box, and the Close menu item on the system menu of the Search Criteria and Document List dialog box.

Example

```
C:\ARS32\ARSGUI32 /Z
```

Disable Exit — /K

If this parameter is specified, Content Manager OnDemand disables the Exit menu item.

Example

```
C:\ARS\ARSGUI32 /K
```

Disable Logoff or Password Change — /X

Use the X parameter to indicate if Content Manager OnDemand disables logoff or a change to the logon password as follows:

- If X is specified with no options, Content Manager OnDemand disables the Logoff and Change Logon Password menu items.
- If X is specified with the L option, Content Manager OnDemand disables the Logoff menu item.
- If X is specified with the P option, Content Manager OnDemand disables the Change Logon Password menu item.

Example

```
C:\ARS32\ARSGUI32 /X
```

Disable Update Servers — /Y

If this parameter is specified, Content Manager OnDemand disables the Update Servers button on the Logon dialog box.

Example

```
C:\ARS32\ARSGUI32 /Y
```

Disable User Confirmation — /B

If this parameter is specified, Content Manager OnDemand does not request confirmation of the user's action when there is an open document during folder close, logoff, or exit.

Example

```
C:\ARS32\ARSGUI32 /B
```

Enable DDE Interface — /I number,path,resid

If this parameter is specified, Content Manager OnDemand enables the Dynamic Data Exchange interface and prepares to accept commands from another application.

number, **path**, and **resid** are interpreted as follows:

- **number** must be an integer between 1 and 5. It indicates the number of DDE-application switch menu items to be provided (for more information, see “ENABLE_SWITCH” on page 144). The default value is 1.
- **path** specifies the fully-qualified name of a resource DLL containing a bitmap for toolbar buttons to be associated with the DDE-application switch menu items. The size of the bitmap depends on the number of menu items. Each button requires a 16 X 15 pel image. If there is a single menu item, the bitmap should be 16 X 15; if two menu items, 16 X 30; and so forth.
- **resid** specifies the bitmap resource identifier within the DLL. It must be an integer value.

Example

```
C:\ARS\ARSGUI32 /I 3,C:\MYAPP\BITMAP.DLL,81
```

Folder Name — /F name

If this parameter is specified, Content Manager OnDemand uses the value as the folder name of the initial folder to be opened. If the value is a valid folder name, the initial Open a Folder dialog box is not displayed.

This parameter may be used in combination with the Logon Server Name, Logon User ID, Logon Password, Disable Logoff, and Password Change, and Maximum Open Folders parameters to limit a user to operations with a single folder on a single server.

Example

```
C:\ARS32\ARSGUI32 /F Credit Card Statements
```

Free Memory When Folder Closed — /Q

If this parameter is specified, Content Manager OnDemand frees all memory associated with a folder when it is closed and refreshes the folder information from the server when it is reopened.

Example

```
C:\ARS32\ARSGUI32 /Q
```

Language ID Override — /L

If this parameter is specified, OnDemand uses the language ID included with this parameter to override the language ID from the Windows registry for the user in the current session. Valid language IDs are three-letter language codes such as ENU, JPN, or CHS.

Example

```
C:\ARS32\ARSGUI32 /L JPN
```

Language Path — /1

This parameter identifies the full path to the files that support the national language environment of the client.

Example

```
C:\ARS32\ARSGUI32 /1 C:\ARS32\ARSGUI32\LOCALE\ENU
```

Logon Password — /P password

If this parameter is specified, Content Manager OnDemand uses the value as the password for the initial user logon.

This parameter may be used in combination with the Logon Server Name and Logon User ID parameters to logon the user during initialization. If all three parameters are specified and are valid, the initial Logon dialog box is not displayed.

Example

```
C:\ARS32\ARSGUI32 /S pikes /U user1 /P pass1
```

Logon Server Name — /S name

If this parameter is specified, Content Manager OnDemand uses the value as the server name for the initial user logon. The name must be one of the server names defined in the Registry. (Use the Update Servers command to add server names to the Registry.)

This parameter may be used in combination with the Logon User ID and Logon Password parameters to logon the user during initialization. If all three parameters are specified and are valid, the initial Logon dialog box is not displayed.

Example

```
C:\ARS32\ARSGUI32 /S pikes /U user1 /P pass1
```

Logon User ID — /U id

If this parameter is specified, Content Manager OnDemand uses the value as the user id for the initial user logon.

This parameter may be used in combination with the Logon Server Name and Logon Password parameters to logon the user during initialization. If all three parameters are specified and are valid, the initial Logon dialog box is not displayed.

Example

```
C:\ARS32\ARSGUI32 /S pikes /U user1 /P pass1
```

Maximum Open Folders — /O number

If this parameter is specified, Content Manager OnDemand limits the number of open folders to no more than the indicated value. This parameter may be used in combination with the Free Memory When Folder Closed parameter to cause folder information to be refreshed from the server each time that a folder is opened.

Example

```
C:\ARS32\ARSGUI32 /O 1 /Q
```

Product Title — /T name

If this parameter is specified, Content Manager OnDemand replaces the default product title at the top of the main Content Manager OnDemand window and on

a number of menu items. This title takes precedence over a title supplied in a Program Information File (refer to Chapter 12, “Program Information File,” on page 187).

Example

C:\ARS32\ARSGUI32 /T Joe's Windows Application

Window Placement — /W placement

If this parameter is specified, Content Manager OnDemand displays the main window as indicated by the placement. The placement must be one of the following:

- **Z** to indicate that the main window is to be *zoomed* (maximized).
- **I** to indicate that the main window is to be *iconized* (minimized).
- **N** to indicate that the main window is *not* to be displayed (made invisible). This option should be specified only if Content Manager OnDemand is to be manipulated with the DDE interface.
- ***x,y,w,h*** to indicate the main window placement, where
 - *x* is the left origin in percent of the screen width.
 - *y* is the top origin in percent of the screen height.
 - *w* is the width in percent of the screen width.
 - *h* is the height in percent of the screen height.

If this parameter is not specified or if no placement value is specified, the main window appears as it did when Content Manager OnDemand was last terminated.

Example

C:\ARS32\ARSGUI32 /W 5,10,90,80

Chapter 6. Dynamic Data Exchange (DDE) and DDE Management Library

Dynamic Data Exchange (DDE) is an interprocess communication mechanism supported by Windows. Two Windows applications carry on a “conversation” with DDE protocols. Content Manager OnDemand can function as a server in such a conversation, providing access to data and performing actions at the direction of a client application. The client application is able to manipulate Content Manager OnDemand by “remote control.”

DDE is based on the messaging system built into Windows. The applications communicate by posting messages to each other. This original DDE communication protocol can be difficult to understand and implement. Windows has subsequently hidden many of the complexities of DDE by providing the DDE Management Library (DDEML), which is a function call interface.

Either the original DDE protocol or the DDEML can be used by a client application when communicating with Content Manager OnDemand. The description and examples in the following sections use the DDEML, and assume a general familiarity with its operation. Refer to the *Microsoft Windows Software Development Kit* or an appropriate textbook for a more complete description of DDE and DDEML.

Invoking the OnDemand 32-bit from another Windows application

This invocation method is typically used when the DDE interface is required.

OnDemand 32-bit can be invoked from another Windows application by using the *CreateProcess* Windows API. The prototype for this function is:

```
BOOL CreateProcess(  
    lpszImageName,  
    lpszCmdLine,  
    lpsaProcess,  
    lpsaThread,  
    fInheritHandles,  
    fdwCreate,  
    lpvEnvironment,  
    lpszCurDir,  
    lpsiStartInfo,  
    lppiProcInfo )
```

lpszCmdLine is a null-terminated character string containing the name of the Content Manager OnDemand executable followed by a set of optional parameters as described below.

Consult your documentation for a description of the other parameters.

This invocation method is typically used when the Dynamic Data Exchange interface is required.

The following is an example of invocation using *CreateProcess*:

```
PROCESS_INFORMATION pi;  
STARTUPINFO sui;  
char * pCmd;
```

```

memset( &sui, 0, sizeof(STARTUPINFO) );
sui.cb = sizeof(STARTUPINFO);

pCmd = "C:\\ARS\\ARSGUI32 "
       "/T Special OnDemand "
       "/I 3,C:\\MYAPP\\BITMAP.DLL,81 "
       "/W 5,10,90,80";

CreateProcess( NULL,
              pCmd,
              NULL,
              NULL,
              FALSE,
              CREATE_NEW_CONSOLE,
              NULL,
              NULL,
              &sui,
              &pi );

```

Content Manager OnDemand invocation and DDEML initialization

Content Manager OnDemand must be an active Windows application before a DDE conversation between it and a client application can be established. The client application typically invokes Content Manager OnDemand as described in “Invoking the OnDemand 32-bit from another Windows application” on page 129, specifying (at a minimum) the Enable DDE Interface parameter. This parameter causes Content Manager OnDemand to enable its DDE interface; if it is not specified, Content Manager OnDemand ignores all DDE communication.

Other parameters can be used to position the window, provide a custom title, logon a user, open a folder, and so on. Many of these actions can also be performed by DDE functions. The command line parameters should be used to establish the initial appearance of the Content Manager OnDemand window. Refer to “Parameters” on page 123 for a complete description of the command line parameters.

After Content Manager OnDemand has been made active, the client application must identify itself to the DDEML. This is done with the DdeInitialize DDEML function.

Finally, the client application must establish a DDE conversation with Content Manager OnDemand. This is done with the DdeConnect DDEML function. The service name of the Content Manager OnDemand server application is ARS. The topic name for the conversation is also ARS.

The following is a typical example for establishing a DDE conversation with Content Manager OnDemand for Windows.

```

/* Global Variables */
DWORD DdeInstance;
HCONV hDdeConv;
.
.
.
/* Local Variables */
FARPROC pfnDdeCallBack;
HSZ hDdeString1, hDdeString2;
UINT rc;
char cmdline[500], buffer[500];
.
.

```

```

    /* Invoke Content Manager OnDemand */
    PROCESS_INFORMATION pi;
    STARTUPINFO sui;

    memset( &sui, 0, sizeof(STARTUPINFO) );
    sui.cb = sizeof(STARTUPINFO);

    rc = CreateProcess( NULL,
                        cmdline,
                        NULL,
                        NULL,
                        FALSE,
                        CREATE_NEW_CONSOLE,
                        NULL,
                        NULL,
                        &sui,
                        &pi );

    if ( !rc )
    {
        sprintf( buffer,
                 "CreateProcess of '%s' failed with error %ld",
                 cmdline,
                 (long)GetLastError( ) );
        MESSAGE( buffer );
        return;
    }

    /* Initialize DDEML */
    pfnDdeCallBack = MakeProcInstance( (FARPROC)DdeCallBack, hInst );
    DdeInstance = 0;
    DdeInitialize( DdeInstance,
                  (PFNCALLBACK)pfnDdeCallBack,
                  APPCLASS_STANDARD | APPCMD_CLIENTONLY,
                  0L );

    if ( DdeInstance == 0 )
    {
        MESSAGE( "DdeInitialize failed" );
        return;
    }

    /* Connect to Content Manager OnDemand DDE Interface */
    hDdeString1 = DdeCreateStringHandle( DdeInstance, "ARS", 0 );
    hDdeString2 = DdeCreateStringHandle( DdeInstance, "ARS", 0 );
    for ( j = 0; j < 100; j++ )
    {
        hDdeConv = DdeConnect( DdeInstance, hDdeString1, hDdeString2, NULL );
        if ( hDdeConv != NULL )
            break;
    }
    DdeFreeStringHandle( DdeInstance, hDdeString1 );
    DdeFreeStringHandle( DdeInstance, hDdeString2 );
    if ( hDdeConv == NULL )
    {
        MESSAGE( "Unable to connect to Content Manager OnDemand" );
        return;
    }

    .
    .
    .

```

DDEML termination

When the client application wants to terminate the DDE conversation, it simply uses the `DdeUninitialize` DDEML function. This informs Content Manager OnDemand that its DDE client has ended the conversation, but it does not cause Content Manager OnDemand to terminate. If the client wants Content Manager OnDemand to terminate, it should tell Content Manager OnDemand to EXIT (refer to “EXIT” on page 145) before ending the conversation.

The client must also be prepared to have the DDE conversation ended by Content Manager OnDemand. If Content Manager OnDemand is terminated, either independently or as a result of a DDE EXIT, the client receives a `XTYP_DISCONNECT` transaction from the DDEML.

DDEML transactions

All Content Manager OnDemand commands are DDEML `XTYP_REQUEST` transactions. Some cause an operation to be performed; some return additional data; and all provide a return code.

The DDEML `DdeClientTransaction` function is used to initiate the transaction. The DDEML item name string contains the command and concatenated parameters. The syntax is the same as the command line (refer to “Parameter Syntax” on page 123) with the DDE command replacing the executable name.

The DDEML `DdeClientTransaction` function returns a data handle containing a return code, and optionally, additional data. The return code is a set of ASCII digits representing one of the values described for the individual commands and summarized in Chapter 8, “Return Codes,” on page 171. These ASCII digits are always followed by a single space character. If additional data is present, it is a null-terminated character string beginning at the character immediately following the space character.

The following example describes a C function that executes Content Manager OnDemand DDE commands and receives return information. All the examples for the individual Content Manager OnDemand DDE commands are based on this C function example.

```
/* Global Variables */
DWORD DdeInstance;
HCONV hDdeConv;

#define ERROR_MAP struct _ErrorMap
ERROR_MAP
{
    int    code;
    char * pMsg;
};

static ERROR_MAP Errors[] =
{
    { ARS_DDE_RC_UNKNOWN_COMMAND,      "Unknown command." },
    { ARS_DDE_RC_PARM_NOT_SPECIFIED,    "Parameter not specified." },
    { ARS_DDE_RC_INVALID_PARM_VALUE,    "Invalid parameter value." },
    { ARS_DDE_RC_SERVER_ERROR,          "Server error." },
    { ARS_DDE_RC_FILE_ERROR,            "File error." },
    { ARS_DDE_RC_NOT_LOGGED_ON,         "Not logged on." },
    { ARS_DDE_RC_MAX_FOLDERS_OPEN,      "Maximum folders open." },
    { ARS_DDE_RC_FOLDER_NOT_OPEN,       "Folder not open." },
    { ARS_DDE_RC_NO_DOC,                "No document exists." },
    { ARS_DDE_RC_OPEN_DOCS,             "Documents are open." }
};
```

```

#define NUM_ERRORS ( sizeof(Errors) / sizeof(ERROR_MAP) )

.
.
.

BOOL DoDdeCommand( char * pCmd,      /* -> Command string      */
                  char * pParms,    /* -> Command parameters  */
                  char * pData )    /* -> Buffer for returned data */
{
    HSZ hDdeString;
    HDEADATA hDdeResult;
    DWORD data_len;
    char * pString;
    int j, rc;

    /* Add parameters to command line. */
    if ( pParms == NULL )
        pParms = "";
    pString = malloc( strlen( pCmd ) + strlen( pParms ) + 2 );
    strcpy( pString, pCmd );
    strcat( pString, " " );
    strcat( pString, pParms );

    /* Perform Content Manager OnDemand DDE command. */
    hDdeString = DdeCreateStringHandle( DdeInstance, pCmd, 0 );
    hDdeResult = DdeClientTransaction( NULL,
                                      0,
                                      hDdeConv,
                                      hDdeString1,
                                      CF_TEXT,
                                      type,
                                      10000L,
                                      NULL );

    DdeFreeStringHandle( DdeInstance, hDdeString );
    free( pString );

    /* Process command result. */
    if ( hDdeResult == NULL )
    {
        /* This should never happen. */
        MESSAGE( "DDE Timeout." );
        return FALSE;
    }
    else
    {
        pString = (char*)DdeAccessData( hDdeResult, &data_len );
        rc = atoi( pString );
        if ( rc == ARS_DDE_RC_NO_ERROR )
        {
            if ( pData != NULL )
                strcpy( pData, strchr( pString, ' ' ) + 1 );
        }
        else
        {
            if ( pData != NULL )
                pData[0] = '\0';
            for ( j = 0; j < NUM_ERRORS; j++ )
                if ( Errors[j].code == rc )
                    break;
            MESSAGE( j < NUM_ERRORS ? Errors[j].pMsg : "Error - invalid return code." );
        }
        DdeUnaccessData( hDdeResult );
        return rc == ARS_DDE_RC_NO_ERROR;
    }
}

```

Chapter 7. Content Manager OnDemand DDE commands

A header file, ARSDDEEX.H, is distributed with Content Manager OnDemand. The header file contains symbolic definitions for many of the values used with the DDE commands. The header file can be included in C/C++ implementations or used as a reference for other languages.

This header file is installed into the INC subdirectory of the Content Manager OnDemand installation directory. This subdirectory can be added to the include file path or the file can be copied to another directory.

ACTIVATE_DOC

Command	Parameters
ACTIVATE_DOC	/D <i>doc id</i>

Parameters:

D

Specifies the document identifier, returned by the OPEN_DOC command, of the document to be activated. This parameter is required.

Action:

Content Manager OnDemand makes the document the active document by bringing its document window to the top and giving it the input focus.

Return Code:

0	ARS_DDE_RC_NO_ERROR
2	ARS_DDE_RC_PARM_NOT_SPECIFIED
3	ARS_DDE_RC_INVALID_PARM_VALUE
6	ARS_DDE_RC_NOT_LOGGED_ON
11	ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data:

None.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
DoDdeCommand( "ACTIVATE_DOC", "/D 1234", NULL );
```

ACTIVATE_FOLDER

Command	Parameters
ACTIVATE_FOLDER	/F <i>folder name</i>

Parameters:

F

Specifies the folder name of a currently open folder. This parameter is required.

Action:

Content Manager OnDemand makes the specified folder the active folder.

Return Code:

0	ARS_DDE_RC_NO_ERROR
2	ARS_DDE_RC_PARM_NOT_SPECIFIED
3	ARS_DDE_RC_INVALID_PARM_VALUE
6	ARS_DDE_RC_NOT_LOGGED_ON
11	ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data:

None.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
DoDdeCommand( "ACTIVATE_FOLDER", "Mary's Folder", NULL );
```

ANNOTATE_DOC

Command	Parameters
ANNOTATE_DOC	/N <i>doc number</i> /P <i>annotation path</i> /G <i>page number</i> /U /C

Parameters:**N**

Specifies the zero-based relative document number within the document list of the active folder. The number of documents in the list can be determined by using the GET_NUM_DOCS_IN_LIST command (refer to “GET_NUM_DOCS_IN_LIST” on page 148). The values associated with a particular document number can be retrieved by using the GET_DOC_VALUES command (refer to “GET_DOC_VALUES” on page 146).

This parameter is required.

P

Specifies the fully-qualified path of a file containing the text of the annotation. If the file contains more than 32,700 bytes, the text is truncated.

This parameter is required.

G

Specifies the document page number associated with the annotation.

This parameter is optional.

U

Indicates that the annotation is public rather than private.

This parameter is optional.

C

Indicates that the annotation may be copied to other servers.

This parameter is optional.

Action:

Content Manager OnDemand adds the annotation to the specified document.

Return Code:

```

0      ARS_DDE_RC_NO_ERROR
2      ARS_DDE_RC_PARM_NOT_SPECIFIED
4      ARS_DDE_RC_SERVER_ERROR
5      ARS_DDE_RC_FILE_ERROR
8      ARS_DDE_RC_FOLDER_NOT_OPEN
9      ARS_DDE_RC_NO_DOC
11     ARS_DDE_RC_USER_ACTION_IN_PROGRESS
12     ARS_DDE_RC_UNAUTHORIZED_OPERATION

```

Return Data:
None.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```

char parms[200];

sprintf( parms,
        "/N %s /L %s /C %s /R /K",
        "22",
        "HP LaserJet on LPT1.AES:",
        "2" );

DoDdeCommand( "PRINT_DOC", parms, NULL );

```

ARRANGE_DOCS

Command	Parameters
ARRANGE_DOCS	/C /H /V /Z /R

Parameters:

C

Indicates that the document windows are to be cascaded.

H

Indicates that the document windows are to be tiled horizontally.

V

Indicates that the document windows are to be tiled vertically.

Z

Indicates that the document windows are to be zoomed (maximized).

R

Indicates that the document windows are to be restored. This is the default parameter.

Action:

Content Manager OnDemand arranges the document windows as indicated by the parameter. If no parameter is specified, **R** is assumed. If multiple parameters are specified, the results are unpredictable.

Return Code:

```

0      ARS_DDE_RC_NO_ERROR
2      ARS_DDE_RC_PARM_NOT_SPECIFIED
6      ARS_DDE_RC_NOT_LOGGED_ON
11     ARS_DDE_RC_USER_ACTION_IN_PROGRESS

```

Return Data:

None.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
DoDdeCommand( "ARRANGE_DOCS", "/V", NULL );
```

CHANGE_PASSWORD

Command	Parameters
CHANGE_PASSWORD	/C <i>Current Password</i> /N <i>New Password</i> /V <i>New Password</i>

Parameters:

C

Specifies the users current password.

N

Specifies users new password.

V

Specifies the users new password again; this is for verification.

Action:

OnDemand changes the logon password for the active user.

Return Code:

0	ARS_DDE_RC_NO_ERROR
2	ARS_DDE_RC_PARM_NOT_SPECIFIED
4	ARS_DDE_RC_SERVER_ERROR
6	ARS_DDE_RC_NOT_LOGGED_ON
22	ARS_DDE_RC_INCORRECT_CURRENT_PASSWORD
23	ARS_DDE_RC_PASSWORD_TOO_SHORT
24	ARS_DDE_RC_NEW_PASSWORD_MISMATCH

Return Data:

None.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
DoDdeCommand( "CHANGE_PASSWORD", "/C tt1sd /N sfd45r /V sfd45r", NULL );
```

CLEAR_FIELDS

Command	Parameters
CLEAR_FIELDS	None.

Parameters:

None.

Action:

Content Manager OnDemand clears the search criteria entry windows for the active folder.

Return Code:

0 ARS_DDE_RC_NO_ERROR
8 ARS_DDE_RC_FOLDER_NOT_OPEN
11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data:

None.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
DoDdeCommand( "CLEAR_FIELDS", "", NULL );
```

CLOSE_ALL_DOCS

Command	Parameters
CLOSE_ALL_DOCS	None.

Parameters:

None.

Action:

Content Manager OnDemand closes all open documents, destroys all document windows, and displays the active folder window.

Return Code:

0 ARS_DDE_RC_NO_ERROR
11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data:

None.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
DoDdeCommand( "CLOSE_ALL_DOCS", "", NULL );
```

CLOSE_ALL_FOLDERS

Command	Parameters
CLOSE_ALL_FOLDERS	None.

Parameters:

None.

Action:

Content Manager OnDemand closes all open folders.

Return Code:

0 ARS_DDE_RC_NO_ERROR
11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data:

None.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
DoDdeCommand( "CLOSE_ALL_FOLDERS", "", NULL );
```

CLOSE_DOC

Command	Parameters
CLOSE_DOC	/D <i>doc id</i>

Parameters:

D

Specifies the document identifier, returned by the OPEN_DOC command, of the document to be closed. This parameter is required.

Action:

Content Manager OnDemand closes the document by destroying the document window.

Return Code:

0	ARS_DDE_RC_NO_ERROR
2	ARS_DDE_RC_PARM_NOT_SPECIFIED
3	ARS_DDE_RC_INVALID_PARM_VALUE
6	ARS_DDE_RC_NOT_LOGGED_ON
11	ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data:

None.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
DoDdeCommand( "CLOSE_DOC", "/D 1234", NULL );
```

CLOSE_FOLDER

Command	Parameters
CLOSE_FOLDER	None.

Parameters:

None.

Action:

Content Manager OnDemand closes the active folder. If there are other open folders, one of them becomes the active folder.

Return Code:

0	ARS_DDE_RC_NO_ERROR
8	ARS_DDE_RC_FOLDER_NOT_OPEN
11	ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data:

None.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
DoDdeCommand( "CLOSE_FOLDER", "", NULL );
```

COPY_DOC_PAGES

Command	Parameters
COPY_DOC_PAGES	/P <i>path</i> /G <i>page number</i> C /A

Parameters:

P

Specifies the fully-qualified path of a file to which the pages are to be copied. If the file does not exist, it is created; if it does exist, the pages are appended to the existing file. This parameter is required.

G

Specifies the pages to be copied:

Specify the *page number* or *page numbers* for the pages you want to copy.

You can specify a single page or a range of pages. You can also specify a combination of single pages and ranges of pages separated by a comma. For example,

/G 3, 5-7, 9, 110-120

copies page 3, pages 5 through 7, page 9, and pages 110 to 120.

If the **G** parameter is not specified, all pages are copied.

When you specify the **G** parameter, you can specify the page or range of pages or the current page by specifying **C**, but you cannot specify pages and **C** at the same time.

The **G** parameter is optional.

C

Specify a **C** to copy the current page.

The **C** parameter is optional.

A

If this parameter is specified, the pages are copied “AS IS” in the data stream format; otherwise, the pages are copied as ASCII text (for AFP and line data only).

This parameter is optional.

Action:

Content Manager OnDemand copies the specified page(s) from the active document to the specified file. If a list of pages is specified, the pages are copied in the order you specify.

Return Code:

2	ARS_DDE_RC_PARM_NOT_SPECIFIED
3	ARS_DDE_RC_INVALID_PARM_VALUE
4	ARS_DDE_RC_SERVER_ERROR
5	ARS_DDE_RC_FILE_ERROR
6	ARS_DDE_RC_NOT_LOGGED_ON
12	ARS_DDE_RC_UNAUTHORIZED_OPERATION
13	ARS_DDE_RC_USER_CANCELLED_OPERATION
14	ARS_DDE_RC_NO_ACTIVE_DOC

Return Data:

None.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
char parms[200];

sprintf( parms,
        "/P %s /G 6, 3-4, 8, 112-118",
        "C:\\ASCII.TXT")

DoDdeCommand( "COPY_DOC_PAGES", parms, NULL );
```

DELETE_DOC

Command	Parameters
DELETE_DOC	/N <i>doc number</i>

Parameters:

N

Specifies the zero-based relative document number within the document list of the active folder. The number of documents in the list can be determined by using the GET_NUM_DOCS_IN_LIST command (refer to “GET_NUM_DOCS_IN_LIST” on page 148). The values associated with a particular document number can be retrieved by using the GET_DOC_VALUES command (refer to “GET_DOC_VALUES” on page 146).

This parameter is required.

The *doc number* may be specified as -1 to indicate that all selected documents are to be deleted.

Action:

Content Manager OnDemand deletes the specified document or all selected documents from the database. The deleted document(s) are removed from the Document List. Since the document numbers may have changed, information from a previous GET_DOC_VALUES command may no longer be valid.

Return Code:

0	ARS_DDE_RC_NO_ERROR
2	ARS_DDE_RC_PARM_NOT_SPECIFIED
3	ARS_DDE_RC_INVALID_PARM
4	ARS_DDE_RC_SERVER_ERROR
8	ARS_DDE_RC_FOLDER_NOT_OPEN
11	ARS_DDE_RC_USER_ACTION_IN_PROGRESS
12	ARS_DDE_RC_UNAUTHORIZED_OPERATION

Return Data:

Content Manager OnDemand returns the number of documents that were successfully deleted. The returned null-terminated string can be converted to a long integer.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
DoDdeCommand( "DELETE_DOC", "23", NULL );
```

DESELECT_DOC

Command	Parameters
DESELECT_DOC	/N <i>doc number</i>

Parameters:

N

Specifies the zero-based relative document number within the document list of the active folder. The number of documents in the list can be determined by using the GET_NUM_DOCS_IN_LIST command (refer to “GET_NUM_DOCS_IN_LIST” on page 148). The values associated with a particular document number can be retrieved by using the GET_DOC_VALUES command (refer to “GET_DOC_VALUES” on page 146).

This parameter is required.

The *doc number* may be specified as -1 to indicate that all documents are to be deselected.

Action:

Content Manager OnDemand deselects (removes highlight from) the Document List line that corresponds to the specific document number.

Return Code:

0	ARS_DDE_RC_NO_ERROR
2	ARS_DDE_RC_PARM_NOT_SPECIFIED
3	ARS_DDE_RC_INVALID_PARM_VALUE
8	ARS_DDE_RC_FOLDER_NOT_OPEN
11	ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data:

None.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
DoDdeCommand( "DESELECT_DOC", "15", NULL );
```

DISABLE_SWITCH

Command	Parameters
DISABLE_SWITCH	/ N <i>number</i>

Parameters:

N

Specifies the number of the menu item/toolbar button to be disabled. The number must be an integer between 1 and the number specified with the “Enable DDE Interface — /I number,path,resid” on page 125. Specifying 1 disables the first menu item/toolbar button; specifying 2 disables the second; and so forth.

This parameter is optional. The default is 1.

Action:

Content Manager OnDemand disables a toolbar icon and a menu item that allow the user to transfer window focus to the client application.

Return Code:

0 ARS_DDE_RC_NO_ERROR
3 ARS_DDE_RC_INVALID_PARM_VALUE
11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data:

None.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
DoDdeCommand( "DISABLE_SWITCH", "", NULL );
```

ENABLE_SWITCH

Command	Parameters
ENABLE_SWITCH	/H <i>client hWnd</i> /C <i>client name</i> /N <i>number</i>

Parameters:**H**

Specifies, as a set of ASCII digits, the window handle of the client application. This parameter is required. The example below indicates how to prepare this parameter.

C

Specifies the name of the client application. This parameter is optional. If not specified, the name is not changed.

N

Specifies the number of the menu item/toolbar button to be enabled. The number must be an integer between 1 and the number specified with the “Enable DDE Interface — /I number,path,resid” on page 125. Specifying 1 enables the first menu item/toolbar button; specifying 2 enables the second; and so forth.

This parameter is optional. The default is 1.

Action:

Content Manager OnDemand enables a toolbar icon and a menu item that allow the user to transfer window focus to the client application. The client name is set as the menu item text.

If the client application has established an Advise Loop (see Chapter 9, “DDEML Advise Loop,” on page 173 for more information), Content Manager OnDemand informs the client application when the user transfers focus.

Return Code:

0 ARS_DDE_RC_NO_ERROR
2 ARS_DDE_RC_PARM_NOT_SPECIFIED
3 ARS_DDE_RC_INVALID_PARM_VALUE
11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data:

None.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
char parms[200];

sprintf( parms,
        "/H %ld /C %s",
        (long)hWnd,
        "Fred's Windows Application" );
DoDdeCommand( "ENABLE_SWITCH", parms, NULL );
```

EXIT

Command	Parameters
EXIT	None.

Parameters:

None.

Action:

Content Manager OnDemand attempts to terminate when this command is received. If at least one document is open, and the Disable User Confirmation command line parameter has not been specified (refer to “Disable User Confirmation — /B” on page 124), a message box is displayed asking the user to confirm program exit.

If Content Manager OnDemand terminates, the client receives a XTYP_DISCONNECT transaction from the DDEML.

Return Code:

```
0      ARS_DDE_RC_NO_ERROR
11     ARS_DDE_RC_USER_ACTION_IN_PROGRESS
```

Return Data:

None.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
DoDdeCommand( "EXIT", "", NULL );
```

GET_DISPLAY_FIELDS

Command	Parameters
GET_DISPLAY_FIELDS	None.

Parameters:

None.

Action:

None.

Return Code:

```
0      ARS_DDE_RC_NO_ERROR
8      ARS_DDE_RC_FOLDER_NOT_OPEN
```

Return Data:

Content Manager OnDemand returns a list of the display fields for the active folder. The list is a null-terminated character string with the tab character separating each field name.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
char data[1000], fields[10][100], * pToken;
int j;

if ( DoDdeCommand( "GET_DISPLAY_FIELDS", "", data ) )
{
    for ( pToken = strtok( data, "\t" ), j = 0;
          pToken != NULL && j < 10;
          pToken = strtok( NULL, "\t" ), j++ )
    {
        strcpy( fields[j], pToken );
    }
}
```

GET_DOC_VALUES

Command	Parameters
GET_DOC_VALUES	/N <i>doc number</i> /S

Parameters:

N

Specifies the zero-based relative document number within the document list of the active folder. The number of documents in the list can be determined by using the GET_NUM_DOCS_IN_LIST command (refer to “GET_NUM_DOCS_IN_LIST” on page 148).

This parameter is required.

S

Indicates that values are to be returned only if the specified list item is selected.

This parameter is optional.

Action:

None.

Return Code:

0	ARS_DDE_RC_NO_ERROR
2	ARS_DDE_RC_PARM_NOT_SPECIFIED
3	ARS_DDE_RC_INVALID_PARM
8	ARS_DDE_RC_FOLDER_NOT_OPEN
11	ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data:

Content Manager OnDemand returns a list of the document values for the specified document within the document list of the active folder. The list is a null-terminated character string with the tab character separating each field value. The values are in the same sequence as the display field names

returned by the GET_DISPLAY_FIELDS command (refer to “GET_DISPLAY_FIELDS” on page 145).

If the document number does not exist in the list or if the S parameter has been specified and the document number is not selected, no values are returned.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
char data[1000], values[10][100], * pToken;
int j;

if ( DoDdeCommand( "GET_DOC_VALUES", "8", data ) )
{
    for ( pToken = strtok( data, "\t" ), j = 0;
          pToken != NULL && j < 10;
          pToken = strtok( NULL, "\t" ), j++ )
    {
        strcpy( values[j], pToken );
    }
}
```

GET_FOLDER_FIELDS

Command	Parameters
GET_FOLDER_FIELDS	None.

Parameters:

None.

Action:

None.

Return Code:

0	ARS_DDE_RC_NO_ERROR
8	ARS_DDE_RC_FOLDER_NOT_OPEN
11	ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data:

Content Manager OnDemand returns a list of the folder fields for the active folder. The list is a null-terminated character string with the tab character separating each field name.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
char data[1000], fields[10][100], * pToken;
int j;

if ( DoDdeCommand( "GET_FOLDER_FIELDS", "", data ) )
{
    for ( pToken = strtok( data, "\t" ), j = 0;
          pToken != NULL && j < 10;
          pToken = strtok( NULL, "\t" ), j++ )
    {
        strcpy( fields[j], pToken );
    }
}
```

GET_FOLDERS

Command	Parameters
GET_FOLDERS	None.

Parameters:

None.

Action:

None.

Return Code:

0 ARS_DDE_RC_NO_ERROR
6 ARS_DDE_RC_NOT_LOGGED_ON
11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data:

Content Manager OnDemand returns a list of the available folders. The list is a null-terminated character string with the tab character separating each folder name.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
char data[1000], folders[10][31], * pToken;  
int j;  
  
if ( DoDdeCommand( "GET_FOLDERS", "", data ) )  
{  
    for ( pToken = strtok( data, "\t" ), j = 0;  
          pToken != NULL && j < 10;  
          pToken = strtok( NULL, "\t" ), j++ )  
    {  
        strcpy( folders[j], pToken );  
    }  
}
```

GET_NUM_DOCS_IN_LIST

Command	Parameters
GET_NUM_DOCS_IN_LIST	None.

Parameters:

None.

Action:

None.

Return Code:

0 ARS_DDE_RC_NO_ERROR
8 ARS_DDE_RC_FOLDER_NOT_OPEN
11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data:

Content Manager OnDemand returns the number of documents currently in the document list of the active folder. The returned null-terminated string can be converted to a long integer.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
char data[100];
long num_docs;

if ( DoDdeCommand( "GET_NUM_DOCS_IN_LIST", "", data ) )
    num_docs = atol( data );
```

GET_NUM_DOC_PAGES

Command	Parameters
GET_NUM_DOC_PAGES	None.

Parameters:

None.

Action:

None.

Return Code:

```
0      ARS_DDE_RC_NO_ERROR
6      ARS_DDE_RC_NOT_LOGGED_ON
14     ARS_DDE_RC_NO_ACTIVE_DOC
```

Return Data:

Content Manager OnDemand returns the number of pages in the active document.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
char data[50];
long num_pages;

if ( DoDdeCommand( "GET_NUM_DOC_PAGES.", "", data ) )
    num_pages = atol( data );
```

GET_PRINTERS

Command	Parameters
GET_PRINTERS	/L /S

Parameters:

L

Indicates that a list of Local printers is to be returned. This parameter is optional, but either this or the **S** parameter must be specified.

S

Indicates that a list of Server printers is to be returned. This parameter is optional, but either this or the **L** parameter must be specified.

Action:

None.

Return Code:

```
0      ARS_DDE_RC_NO_ERROR
```

```

2      ARS_DDE_RC_PARM_NOT_SPECIFIED
6      ARS_DDE_RC_NOT_LOGGED_ON
11     ARS_DDE_RC_USER_ACTION_IN_PROGRESS

```

Return Data:

Content Manager OnDemand returns a list of the available printers, either Local or Server printers as indicated by the parameter. If both parameters are specified, the results are unpredictable. The list is a null-terminated character string with the tab character separating each printer name.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```

char data[1000], local_printers[10][100], * pToken;
int j;

if ( DoDdeCommand( "GET_PRINTERS", "/L", data ) )
{
    for ( pToken = strtok( data, "\t" ), j = 0;
          pToken != NULL && j < 10;
          pToken = strtok( NULL, "\t" ), j++ )
    {
        strcpy( local_printers[j], pToken );
    }
}

```

GET_QUERY_FIELDS

Command	Parameters
GET_QUERY_FIELDS	None.

Parameters:

None.

Action:

None.

Return Code:

```

0      ARS_DDE_RC_NO_ERROR
8      ARS_DDE_RC_FOLDER_NOT_OPEN
11     ARS_DDE_RC_USER_ACTION_IN_PROGRESS

```

Return Data:

Content Manager OnDemand returns a list of the query fields for the active folder. The list is a null-terminated character string with the tab character separating each field name.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```

char data[1000], fields[10][100], * pToken;
int j;

if ( DoDdeCommand( "GET_QUERY_FIELDS", "", data ) )
{
    for ( pToken = strtok( data, "\t" ), j = 0;
          pToken != NULL && j < 10;
          pToken = strtok( NULL, "\t" ), j++ )
    {

```

```

        {
            strcpy( fields[j], pToken );
        }
    }
}

```

GET_SERVERS

Command	Parameters
GET_SERVERS	None.

Parameters:

None.

Action:

None.

Return Code:

0 ARS_DDE_RC_NO_ERROR
 11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data:

Content Manager OnDemand returns a list of the available servers. The list is a null-terminated character string with the tab character separating each server name.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```

char data[1000], servers[10][100], * pToken;
int j;

if ( DoDdeCommand( "GET_SERVERS", "", data ) )
{
    for ( pToken = strtok( data, "\t" ), j = 0;
          pToken != NULL && j < 10;
          pToken = strtok( NULL, "\t" ), j++ )
    {
        strcpy( servers[j], pToken );
    }
}

```

LOGOFF

Command	Parameters
LOGOFF	None.

Parameters:

None.

Action:

Content Manager OnDemand performs a logoff.

Return Code:

0 ARS_DDE_RC_NO_ERROR
 11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data:

None.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
DoDdeCommand( "LOGOFF", "", NULL );
```

LOGON

Command	Parameters
LOGON	/S <i>server name</i> /U <i>user id</i> /P <i>password</i> /R

Parameters:**S**

Specifies the logon server name. This parameter is required.

U

Specifies the logon user ID. This parameter is required.

P

Specifies the logon password. This parameter is required.

R

Indicates that if the logon fails, control is to be returned. This parameter is optional. If this parameter is not specified and the logon fails, the OnDemand Logon dialog box is displayed.

Action:

Content Manager OnDemand attempts a logon with the specified server name, user ID, and password. A list of the available server names can be retrieved by using the GET_SERVERS command (refer to “GET_SERVERS” on page 151).

If a user is already logged on, a logoff is performed prior to the logon. If the logon fails, the action is determined by the /R parameter.

Return Code:

0	ARS_DDE_RC_NO_ERROR
2	ARS_DDE_RC_PARM_NOT_SPECIFIED
11	ARS_DDE_RC_USER_ACTION_IN_PROGRESS
13	ARS_DDE_RC_USER_CANCELLED_OPERATION
25	ARS_DDE_RC_INVALID_USER_PASS_SERVER
26	ARS_DDE_RC_PASSWORD_EXPIRED

Return Data:

If the logon is successful, Content Manager OnDemand returns the user ID and password that were used to logon. The null-terminated character string contains the user ID, followed by a tab character, followed by the password.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
char parms[200];

sprintf( parms,
        "/S %s /U %s /P %s",
        "server26",
        "horatio",
```

```

"vrxtwc" );

DoDdeCommand( "LOGON", parms, NULL );

```

OPEN_DOC

Command	Parameters
OPEN_DOC	<i>/N doc number /P path</i>

Parameters:

N

Specifies the zero-based relative document number within the document list of the active folder. The number of documents in the list can be determined by using the GET_NUM_DOCS_IN_LIST command (refer to "GET_NUM_DOCS_IN_LIST" on page 148). The values associated with a particular document number can be retrieved by using the GET_DOC_VALUES command (refer to "GET_DOC_VALUES" on page 146).

This parameter is required. **P**

Specifies the fully-qualified path of a file containing the document data.

If this parameter is specified, Content Manager OnDemand does not access the database to retrieve a document, but it will retrieve a resource group if one is required.

This parameter is optional.

Action:

Content Manager OnDemand opens the document by displaying the first page in a document window.

Return Code:

0	ARS_DDE_RC_NO_ERROR
2	ARS_DDE_RC_PARM_NOT_SPECIFIED
3	ARS_DDE_RC_INVALID_PARM_VALUE
8	ARS_DDE_RC_FOLDER_NOT_OPEN
11	ARS_DDE_RC_USER_ACTION_IN_PROGRESS
12	ARS_DDE_RC_UNAUTHORIZED_OPERATION
13	ARS_DDE_RC_USER_CANCELLED_OPERATION

Return Data:

If the document is successfully opened, Content Manager OnDemand returns a *doc id*. This string contains a maximum of 20 characters.

This *doc id* is required as a parameter to other commands such as ACTIVATE_DOC and CLOSE_DOC.

Example:

Refer to "DDEML transactions" on page 132 for a description of the DoDdeCommand function.

```

char doc_id[21];

DoDdeCommand( "OPEN_DOC", "/N 23", doc_id );

```

OPEN_FOLDER

Command	Parameters
OPEN_FOLDER	<i>/F folder name /S /C /R /D /V /P /T /A /N /L /0 button text /1 button text /2 button text /3 button text /4 button text /5 button text /6 button text /7 button text /8 button text /9 button text</i>

Parameters:

F

Specifies the name of one of the folders available to the user on the server. This parameter is required.

S

Indicates that the Search button is **not** to appear on the Search Criteria of the Search Criteria and Document List dialog box. This parameter is optional.

C

Indicates that the Clear All Fields button is **not** to appear on the Search Criteria of the Search Criteria and Document List dialog box. This parameter is optional.

R

Indicates that the Restore Defaults button is **not** to appear on the Search Criteria of the Search Criteria and Document List dialog box. This parameter is optional.

D

Indicates that the AND/OR buttons are **not** to appear on the Search Criteria of the Search Criteria and Document List dialog box. This parameter is optional.

V

Indicates that the View All Selected button is **not** to appear on the Document List of the Search Criteria and Document List dialog box. This parameter is optional.

P

Indicates that the Print All Selected button is **not** to appear on the Document List of the Search Criteria and Document List dialog box. This parameter is optional.

T

Indicates that the Sort List ... button is **not** to appear on the Search Criteria of the Search Criteria and Document List dialog box. This parameter is optional.

A

Indicates that the Audit buttons are **not** to appear on the Search Criteria of the Search Criteria and Document List dialog box. This parameter is optional.

N

Indicates that the Append button is **not** to appear on the Document List of the Search Criteria and Document List dialog box. This parameter is optional.

L

Indicates that the AutoScroll button is **not** to appear on the Document List of the Search Criteria and Document List dialog box. This parameter is optional.

0 *button text*

Specifies the text for a button which is to appear on the Search Criteria of the Search Criteria and Document List dialog box. This parameter is optional.

If the text includes an &, the following character becomes an accelerator key. When the user clicks on the button, Content Manager OnDemand informs the client application through an Advise Loop established for the application. For more information on Advise Loops, see Chapter 9, "DDEML Advise Loop," on page 173.

1 *button text*

Specifies the text for a button which is to appear on the Search Criteria of the Search Criteria and Document List dialog box. This parameter is optional.

If the text includes an &, the following character becomes an accelerator key. When the user clicks on the button, Content Manager OnDemand informs the client application through an Advise Loop established for the application. For more information on Advise Loops, see Chapter 9, "DDEML Advise Loop," on page 173.

2 *button text*

Specifies the text for a button which is to appear on the Search Criteria of the Search Criteria and Document List dialog box. This is an optional parameter.

If the text includes an &, the following character becomes an accelerator key. When the user clicks on the button, Content Manager OnDemand informs the client application through an Advise Loop established for the application. For more information on Advise Loops, see Chapter 9, "DDEML Advise Loop," on page 173.

3 *button text*

Specifies the text for a button which is to appear on the Search Criteria of the Search Criteria and Document List dialog box. This is an optional parameter.

If the text includes an &, the following character becomes an accelerator key. When the user clicks on the button, Content Manager OnDemand informs the client application through an Advise Loop established for the application. For more information on Advise Loops, see Chapter 9, "DDEML Advise Loop," on page 173.

4 *button text*

Specifies the text for a button which is to appear on the Search Criteria of the Search Criteria and Document List dialog box. This is an optional parameter.

If the text includes an &, the following character becomes an accelerator key. When the user clicks on the button, Content Manager OnDemand informs the client application through an Advise Loop established for the application. For more information on Advise Loops, Chapter 9, "DDEML Advise Loop," on page 173.

5 button text

Specifies the text for a button which is to appear on the Document List of the Search Criteria and Document List dialog box. This is an optional parameter.

If the text includes an &, the following character becomes an accelerator key. When the user clicks on the button, Content Manager OnDemand informs the client application through an Advise Loop established for the application. For more information on Advise Loops, Chapter 9, "DDEML Advise Loop," on page 173.

6 button text

Specifies the text for a button which is to appear on the Document List of the Search Criteria and Document List dialog box. This is an optional parameter.

If the text includes an &, the following character becomes an accelerator key. When the user clicks on the button, Content Manager OnDemand informs the client application through an Advise Loop established for the application. For more information on Advise Loops, see Chapter 9, "DDEML Advise Loop," on page 173.

7 button text

Specifies the text for a button which is to appear on the Document List of the Search Criteria and Document List dialog box. This is an optional parameter.

If the text includes an &, the following character becomes an accelerator key. When the user clicks on the button, Content Manager OnDemand informs the client application through an Advise Loop established for the application. For more information on Advise Loops, see Chapter 9, "DDEML Advise Loop," on page 173.

8 button text

Specifies the text for a button which is to appear on the Document List of the Search Criteria and Document List dialog box. This is an optional parameter.

If the text includes an &, the following character becomes an accelerator key. When the user clicks on the button, Content Manager OnDemand informs the client application through an Advise Loop established for the application. For more information on Advise Loops, see Chapter 9, "DDEML Advise Loop," on page 173.

9 button text

Specifies the text for a button which is to appear on the Document List of the Search Criteria and Document List dialog box. This is an optional parameter.

If the text includes an &, the following character becomes an accelerator key. When the user clicks on the button, Content Manager OnDemand

informs the client application through an Advise Loop established for the application. For more information on Advise Loops, see Chapter 9, “DDEML Advise Loop,” on page 173.

Action:

Content Manager OnDemand attempts to open the specified folder. A list of the available folder names can be retrieved by using the GET_FOLDERS command (refer to “GET_FOLDERS” on page 148). The Open a Folder dialog box is not displayed unless the open fails. If the folder is successfully opened, it becomes the active folder.

Multiple folders can be open concurrently. One of them is the active folder.

Return Code:

0	ARS_DDE_RC_NO_ERROR
2	ARS_DDE_RC_PARM_NOT_SPECIFIED
3	ARS_DDE_RC_INVALID_PARM_VALUE
6	ARS_DDE_RC_NOT_LOGGED_ON
7	ARS_DDE_RC_MAX_FOLDERS_OPEN
11	ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data:

None.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
DoDdeCommand( "OPEN_FOLDER", "/F Mary's Folder", NULL );
```

PRINT_DOC

Command	Parameters
PRINT_DOC	/N <i>doc number</i> /L <i>printer name</i> /S <i>printer name</i> /C <i>copies</i> /R <i>orientation</i> /K <i>margins</i> /M

Parameters:

N *doc number*

Specifies the zero-based relative document number within the document list of the active folder. The number of documents in the list can be determined by using the GET_NUM_DOCS_IN_LIST command (refer to “GET_NUM_DOCS_IN_LIST” on page 148). The values associated with a particular document number can be retrieved by using the GET_DOC_VALUES command (refer to “GET_DOC_VALUES” on page 146).

This parameter is optional. If it is not specified, the active document is printed.

L *printer name*

Specifies a Local printer name. The names of the available local printers can be determined by using the GET_PRINTERS command (refer to “GET_PRINTERS” on page 149).

This parameter is optional, but either this or the **S** parameter must be specified.

S *printer*

Specifies a Server printer name. The names of the available server printers can be determined by using the GET_PRINTERS command (refer to "GET_PRINTERS" on page 149).

This parameter is optional, but either this or the **L** parameter must be specified.

C *copies*

Specifies the number of copies of the document to be printed. The value must be a number between 1 and 100.

This parameter is optional. If not specified, one copy is printed.

R *orientation*

Specifies the document orientation for printing. For *orientation*, specify one of the following:

- B** rotates the document to best fit the paper.

- P** prints the document in the portrait orientation.

- L** prints the document in the landscape orientation.

- A** prints the document as specified in the printer.

If *orientation* is not specified, **B** is assumed.

This parameter is optional. If this parameter is not specified, **/R A** is assumed.

If a Server printer is specified, the **R** parameter is ignored.

K *margins*

Specifies the page margins to be used. For *margins*, specify *t,b,l,r* where:

- t* is the top margin.

- b* is the bottom margin.

- l* is the left margin.

- r* is the right margin.

Each margin value must be a non-negative decimal number. If no margin values are given, the current margin values are used.

This parameter is optional. If this parameter is not specified, margins of 0 (zero) are used. (A zero margin may cause data truncation on some printers.)

If a Server printer is specified, the **K** parameter is ignored.

M

Indicates that the margins specified with the **K** parameter are given in millimeters rather than inches.

This parameter is optional. If the **M** parameter is not specified, the margins are assumed to be in inches.

Action:

Content Manager OnDemand prints the page(s) of the document on the specified printer.

Return Code:

0	ARS_DDE_RC_NO_ERROR
2	ARS_DDE_RC_PARM_NOT_SPECIFIED
3	ARS_DDE_RC_INVALID_PARM_VALUE
6	ARS_DDE_RC_NOT_LOGGED_ON
8	ARS_DDE_RC_FOLDER_NOT_OPEN

9 ARS_DDE_RC_NO_DOC
 11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS
 12 ARS_DDE_RC_UNAUTHORIZED_OPERATION

Return Data:
 None.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
char parms[200];

sprintf( parms,
        "/N %s /L %s /C %s /R B /K 0.5,1.2,1,1",
        "17",
        "HP LaserJet on LPT1.AES:",
        "2" );

DoDdeCommand( "PRINT_DOC", parms, NULL );
```

RESTORE_DEFAULTS

Command	Parameters
RESTORE_DEFAULTS	None.

Parameters:
 None.

Action:

Content Manager OnDemand sets the search criteria entry windows for the active folder to their default values.

Return Code:

0 ARS_DDE_RC_NO_ERROR
 8 ARS_DDE_RC_FOLDER_NOT_OPEN
 11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data:
 None.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
DoDdeCommand( "RESTORE_DEFAULTS", "", NULL );
```

RETRIEVE_DOC

Command	Parameters
RETRIEVE_DOC	/N <i>doc number</i> /P <i>doc path</i> /R <i>resgrp path</i> /C <i>combined path</i> /A

Parameters:
 N

Specifies the zero-based relative document number within the document list of the active folder. The number of documents in the list can be determined by using the GET_NUM_DOCS_IN_LIST command (refer to “GET_NUM_DOCS_IN_LIST” on page 148). The values associated with a

particular document number can be retrieved by using the GET_DOC_VALUES command (refer to "GET_DOC_VALUES" on page 146).

This parameter is required.

P

Specifies the fully-qualified path of a file into which the document data is to be placed. This parameter is optional.

R

Specifies the fully-qualified path of a file into which the resource group data is to be placed. This parameter is optional.

C

Specifies the fully-qualified path of a file into which the combined resource group and data is to be placed. This parameter is optional.

A

Indicates that the data is to be appended to an existing file rather than replacing the file. This parameter is optional.

Action:

Content Manager OnDemand retrieves the resource group and/or document data and copies or appends it to the specified files. Any combination of the parameters may be specified, except that N is required.

Return Code:

0	ARS_DDE_RC_NO_ERROR
2	ARS_DDE_RC_PARM_NOT_SPECIFIED
3	ARS_DDE_RC_INVALID_PARM_VALUE
4	ARS_DDE_RC_SERVER_ERROR
5	ARS_DDE_RC_FILE_ERROR
8	ARS_DDE_RC_FOLDER_NOT_OPEN
9	ARS_DDE_RC_NO_DOC
11	ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data:

None.

Example:

Refer to "DDEML transactions" on page 132 for a description of the DoDdeCommand function.

```
char parms[200];

sprintf( parms,
        "/N %d /C %s",
        26,
        "C:\\\\DATA\\COMBINED.FIL");

DoDdeCommand( "RETRIEVE_DOC", parms, NULL );
```

SEARCH_FOLDER

Command	Parameters
SEARCH_FOLDER	/A /O

Parameters:**A**

Indicates whether the documents resulting from the search are to be appended to the existing list. This parameter is optional. If not specified, the documents replace the previous list.

O

Indicates that the search criteria are to be ORed. This parameter is optional. If not specified, the search criteria are ANDed.

Action:

Content Manager OnDemand searches the active folder using the current search criteria.

Return Code:

0 ARS_DDE_RC_NO_ERROR
 8 ARS_DDE_RC_FOLDER_NOT_OPEN
 11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data:

None.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
DoDdeCommand( "SEARCH_FOLDER", "", NULL );
```

SELECT_DOC

Command	Parameters
SELECT_DOC	/N <i>doc number</i>

Parameters:**N**

Specifies the zero-based relative document number within the document list of the active folder. The number of documents in the list can be determined by using the GET_NUM_DOCS_IN_LIST command (refer to “GET_NUM_DOCS_IN_LIST” on page 148). The values associated with a particular document number can be retrieved by using the GET_DOC_VALUES command (refer to “GET_DOC_VALUES” on page 146).

This parameter is required.

The *doc number* may be specified as -1 to indicate that all documents are to be selected.

Action:

Content Manager OnDemand selects (highlights) the Document List line that corresponds to the specific document number.

Return Code:

0 ARS_DDE_RC_NO_ERROR
 2 ARS_DDE_RC_PARM_NOT_SPECIFIED
 3 ARS_DDE_RC_INVALID_PARM
 8 ARS_DDE_RC_FOLDER_NOT_OPEN
 11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data:

None.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
DoDdeCommand( "SELECT_DOC", "-1", NULL );
```

SET_FIELD_DATA

Command	Parameters
SET_FIELD_DATA	/F <i>field name</i> /O <i>operator</i> /1 <i>value1</i> /2 <i>value2</i>

Parameters:**F**

Specifies the name of a search field in the active folder. This parameter is required.

A list of the field names can be retrieved by using the GET_QUERY_FIELDS command (refer to “GET_QUERY_FIELDS” on page 150).

O

Specifies the search operator to be used for the field. It must be one of the following:

EQ	for Equal
NE	for Not Equal
LT	for Less Than
LE	for Less Than or Equal
GT	for Greater Than
GE	for Greater Than or Equal
BW	for Between
NB	for Not Between
IN	for In
NI	for Not In
LK	for Like
NL	for Not Like

The operator must be one of those permitted for the field.

This parameter is optional. If not specified, the operator remains unchanged.

1

Specifies the value to be used for the first, and perhaps only, entry window for the field. This parameter is optional. If not specified, the value remains unchanged.

2

Specifies the value to be used for the second entry window for the field. This value is ignored if the search operator for the field is other than Between or Not Between. This parameter is optional. If not specified, the value remains unchanged.

Action:

Content Manager OnDemand updates the search operator, the first entry window, and/or the second entry window for the specified search field in the active folder.

Return Code:

0	ARS_DDE_RC_NO_ERROR
2	ARS_DDE_RC_PARM_NOT_SPECIFIED
3	ARS_DDE_RC_INVALID_PARM_VALUE
8	ARS_DDE_RC_FOLDER_NOT_OPEN
11	ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data:

None.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
char parms[200];

sprintf( parms,
        "/F %s /O %s /1 %s /2 %s",
        "Account",
        "BW",
        "123456",
        "987654" );

DoDdeCommand( "SET_FIELD_DATA", parms, NULL );
```

SET_FOCUS

Command	Parameters
SET_FOCUS	None.

Parameters:

None.

Action:

Content Manager OnDemand becomes the active window.

Return Code:

0	ARS_DDE_RC_NO_ERROR
11	ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data:

None.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
DoDdeCommand( "SET_FOCUS", "", NULL );
```

SET_HELP_PATH

Command	Parameters
SET_HELP_PATH	/P <i>path</i>

Parameters:

P

Specifies the fully-qualified path for the Windows help file.

This parameter is required.

Action:

Content Manager OnDemand invokes the specified help file when the user requests help for one of the buttons or menu items associated with the client application.

The help file is invoked with one of the following context IDs:

Search Criteria and Document List Dialog Box:

0x50800

ARS_DDE_HELP_ID_CRITERIA_BUTTON_1

0x50801

ARS_DDE_HELP_ID_CRITERIA_BUTTON_2

0x50802

ARS_DDE_HELP_ID_CRITERIA_BUTTON_3

0x50803

ARS_DDE_HELP_ID_CRITERIA_BUTTON_4

0x50804

ARS_DDE_HELP_ID_CRITERIA_BUTTON_5

0x50805

ARS_DDE_HELP_ID_DOCLIST_BUTTON_1

0x50806

ARS_DDE_HELP_ID_DOCLIST_BUTTON_2

0x50807

ARS_DDE_HELP_ID_DOCLIST_BUTTON_3

0x50808

ARS_DDE_HELP_ID_DOCLIST_BUTTON_4

0x50809

ARS_DDE_HELP_ID_DOCLIST_BUTTON_5

Toolbar and Menu Items:

0x5080A

ARS_DDE_HELP_ID_SWITCH_FOCUS

Return Code:

0 ARS_DDE_RC_NO_ERROR

2 ARS_DDE_RC_PARM_NOT_SPECIFIED

11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data:

None.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
DoDdeCommand( "SET_HELP_PARM", "C:\DDEAPPL\DDEAPPL.HLP", NULL);
```

SET_USER_MSG_MODE

Command	Parameters
SET_USER_MSG_MODE	/M <i>mode</i>

Parameters:**M**

Specifies the user message mode.

For *mode*, specify one of the following:

- 1 Content Manager OnDemand always displays any user messages resulting from a DDE command.
- 2 Content Manager OnDemand displays the user messages resulting from a DDE command only if the Content Manager OnDemand window is visible and not minimized.
- 3 Content Manager OnDemand suppresses any user messages resulting from a DDE command.

This parameter is required.

Action:

Content Manager OnDemand displays or suppresses subsequent user messages as specified by the parameters.

Return Code:

- | | |
|---|-------------------------------|
| 0 | ARS_DDE_RC_NO_ERROR |
| 2 | ARS_DDE_RC_PARM_NOT_SPECIFIED |
| 3 | ARS_DDE_RC_INVALID_PARM_VALUE |

Return Data:

None.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
DoDdeCommand( "SET_USER_MSG_MODE", "/M 2", NULL);
```

SHOW_WINDOW

Command	Parameters
SHOW_WINDOW	/W <i>placement</i>

Parameters:**W**

Specifies the show status and position of the Content Manager OnDemand window. This parameter takes the same values as the Window Placement command line parameter. Refer to “Window Placement — /W placement” on page 127 for a description of these values.

If you specify this parameter without a value, the window is displayed at its most recent position with the most recent dimensions.

This parameter is required.

Action:

Content Manager OnDemand shows and/or positions its main window as specified by the parameter value.

Return Code:

0 ARS_DDE_RC_NO_ERROR
2 ARS_DDE_RC_PARM_NOT_SPECIFIED
11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data:

None.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
DoDdeCommand( "SHOW_WINDOW", "/W 25,0,75,100", NULL );
```

STORE_DOC

Command	Parameters
STORE_DOC	/P <i>doc path</i> /G <i>appl group name</i> /A <i>appl name</i> /n <i>value</i> /n <i>value</i> ...

Parameters:

P

Specifies the fully-qualified path of a file containing the document data to be stored in the Content Manager OnDemand database. This parameter is required.

G

Specifies the name of an application group within the active folder. It is the responsibility of the caller to know the application group names associated with the active folder. This parameter is required.

A

Specifies the name of an application within the specified application group. It is the responsibility of the caller to know the application names associated with the specified application group. This parameter is required.

n

Specifies the value associated with a folder field. n is X'01' for the first field of the folder; X'02' for the second; and so forth. The associated *value* is a character string which can be converted to data of the field type (i.e., integer, date, etc.).

The number and order of folder fields can be determined by using the GET_FOLDER_FIELDS command. For more information on this command, see “GET_FOLDER_FIELDS” on page 147.

Any folder fields not specified are given an empty string for string fields or zero for numeric fields. If extraneous fields are specified, they are ignored.

Date fields must be provided in the format required for the field (for example, specifying 02/03/96 is invalid when February 3, 1996 is required).

If a slash character is to be included in a value, such as in a date, two consecutive slashes must be specified. For more information, see "Parameter Syntax" on page 123.

Action:

Content Manager OnDemand converts the folder field values to application group fields and stores the data from the specified file in the database as a document associated with the specified application group and application.

Return Code:

0	ARS_DDE_RC_NO_ERROR
2	ARS_DDE_RC_PARM_NOT_SPECIFIED
4	ARS_DDE_RC_SERVER_ERROR
5	ARS_DDE_RC_FILE_ERROR
8	ARS_DDE_RC_FOLDER_NOT_OPEN
11	ARS_DDE_RC_USER_ACTION_IN_PROGRESS
12	ARS_DDE_RC_UNAUTHORIZED_OPERATION
15	ARS_DDE_RC_INVALID_APPL_GROUP_NAME
16	ARS_DDE_RC_INVALID_APPL_NAME
17	ARS_DDE_RC_INVALID_INTEGER_FIELD
18	ARS_DDE_RC_INVALID_DECIMAL_FIELD
19	ARS_DDE_RC_INVALID_DATE_FIELD
20	ARS_DDE_RC_INVALID_APPLGRP_FIELD_TYPE
27	ARS_DDE_RC_TOO_MANY_VALUE_CHARS

Return Data:

If the return code is one of the following:

ARS_DDE_RC_INVALID_INTEGER_FIELD
ARS_DDE_RC_INVALID_DECIMAL_FIELD
ARS_DDE_RC_INVALID_DATE_FIELD
ARS_DDE_RC_INVALID_APPLGRP_FIELD_TYPE
ARS_DDE_RC_TOO_MANY_VALUE_CHARS

then Content Manager OnDemand returns the relative folder field number of the invalid field. For example, if the first folder field is invalid, Content Manager OnDemand returns a 0 (zero); if the second folder field is invalid, Content Manager OnDemand returns a 1 (one); and so forth.

Example:

Refer to "DDEML transactions" on page 132 for a description of the DoDdeCommand function.

C/C + + Example

```
char parms[200];

sprintf( parms,
        "/D %s /G %s /A %s /\x'01' %s /\x'02' %s /\x'03' %s",
        "D:\DATA\DOCDATA.AFP",
        "Student Data",
        "Grades",
        "Coed, Mary",
        "05//23//95",
        "3.15" );

DoDdeCommand( "STORE_DOC", parms, NULL );
```

Visual Basic Example

```

Dim cmdline As String
cmdline = "STORE_DOC /P D:\Data\DocData.AFP "
cmdline = cmdline + "/G Student Data "
cmdline = cmdline + "/A Grades "
cmdline = cmdline + "/" Chr(1) + " Coed.Mary "
cmdline = cmdline + "/" Chr(2) + " 05//23//95 "
cmdline = cmdline + "/" Chr(3) + " 3.15"

Call fncDDElink ( arstopic, cmdline, linktype, 3000 )

```

Note: For a definition of fncDDElink, see Appendix A, "Microsoft Visual Basic DDE sample program," on page 223.

UPDATE_DOC

Command	Parameters
UPDATE_DOC	/N <i>doc number</i> /F <i>field name</i> /V <i>field value</i>

Parameters:

N

Specifies zero-based relative document number within the document list of the active folder. This parameter is required.

The doc number may be specified as -1 to indicate that all selected documents are to be updated.

F

Specifies the name of a folder field. This parameter is required.

V

Specifies the value to be stored in the specified folder field. This value will be converted to data of the field type (that is, integer, date, and so on).

Date fields must be provided in the format required for the field (for example, specifying 02/03/96 is invalid when February 3, 1996 is required).

If a slash (/) character is to be included in a value (for example, in a date), two consecutive slashes must be specified. For more information, see "Parameter Syntax" on page 123.

This parameter is required.

Action:

Content Manager OnDemand converts the folder field value to an application group field and updates the data from the specified value in the database.

Return Code:

0	ARS_DDE_RC_NO_ERROR
3	ARS_DDE_RC_INVALID_PARM_VALUE
4	ARS_DDE_RC_SERVER_ERROR
8	ARS_DDE_RC_FOLDER_NOT_OPEN
12	ARS_DDE_RC_UNAUTHORIZED_OPERATION
20	ARS_DDE_RC_INVALID_APPLGRP_FIELD_TYPE

Return Data:

Content Manager OnDemand returns the number of documents that were successfully updated. The returned null-terminated string can be converted to a long integer.

Example:

Refer to “DDEML transactions” on page 132 for a description of the DoDdeCommand function.

```
DoDdeCommand( "UPDATE_DOC", "/N 1 /F Balance /V 123.45", NULL );
```

Chapter 8. Return Codes

The following return codes are possible from DDE commands.

- 0 ARS_DDE_RC_NO_ERROR - Indicates that the command was executed without error.
- 1 ARS_DDE_RC_UNKNOWN_COMMAND - Indicates that the command was not a valid Content Manager OnDemand DDE command.
- 2 ARS_DDE_RC_PARM_NOT_SPECIFIED - Indicates that a required parameter was not specified.
- 3 ARS_DDE_RC_INVALID_PARM_VALUE - Indicates that a parameter specified an invalid value.
- 4 ARS_DDE_RC_SERVER_ERROR - Indicates that a server error occurred while accessing the database.
- 5 ARS_DDE_RC_FILE_ERROR - Indicates that an error occurred during an input/output operation.
- 6 ARS_DDE_RC_NOT_LOGGED_ON - Indicates that a user must be logged on for the command to be executed.
- 7 ARS_DDE_RC_MAX_FOLDERS_OPEN - Indicates that the maximum number of folders are already open.
- 8 ARS_DDE_RC_FOLDER_NOT_OPEN - Indicates that a folder must be open and active for the command to be executed.
- 9 ARS_DDE_RC_NO_DOC - Indicates that there is no database data associated with a document in the document list.
- 11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS - Indicates that Content Manager OnDemand is busy performing a user-initiated action. A modal dialog box or message box is probably being displayed.
- 12 ARS_DDE_RC_UNAUTHORIZED_OPERATION - Indicates that the user currently logged on is not authorized to perform the requested operation.
- 13 ARS_DDE_RC_USER_CANCELLED_OPERATION - Indicates that the user cancelled an operation requested through DDE.
- 14 ARS_DDE_RC_NO_ACTIVE_DOC - Indicates that there is no currently active document.
- 15 ARS_DDE_RC_INVALID_APPL_GROUP_NAME - Indicates that the application group name provided is not valid for the folder.
- 16 ARS_DDE_RC_INVALID_APPL_NAME - Indicates that the application name provided is not valid for the application group.
- 17 ARS_DDE_RC_INVALID_INTEGER_FIELD - Indicates that the value provided for a folder field is not a valid integer.
- 18 ARS_DDE_RC_INVALID_DECIMAL_FIELD - Indicates that the value provided for a folder field is not a valid decimal.
- 19 ARS_DDE_RC_INVALID_DATE_FIELD - Indicates that the value provided for a folder field is not a valid date.

- 20 ARS_DDE_RC_INVALID_APPLGRP_FIELD_TYPE - Indicates that the field type within the application group is not valid.
- 21 ARS_DDE_RC_DOC_NOT_VIEWABLE_OR_PRINTABLE - Indicates that the specified document cannot be displayed or printed using the DDE interface.
- 22 ARS_DDE_RC_INCORRECT_CURRENT_PASSWORD - Indicates that the current password specified for the user is incorrect.
- 23 ARS_DDE_RC_PASSWORD_TOO_SHORT - Indicates that the new password specified is not long enough.
- 24 ARS_DDE_RC_NEW_PASSWORD_MISMATCH - Indicates that the two passwords specified for the new password do not match.
- 25 ARS_DDE_RC_INVALID_USER_PASS_SERVER - Indicates that the userid, password, or server was not valid.
- 26 ARS_DDE_RC_PASSWORD_EXPIRED - Indicates that the password has expired.

Chapter 9. DDEML Advise Loop

The client application may create a DDEML Advise Loop in order to be informed when one of the following events occur:

Code	Event
------	-------

S	User switched focus to the client application using the first menu item/toolbar button.
S2	User switched focus to the client application using the second menu item/toolbar button.
S3	User switched focus to the client application using the third menu item/toolbar button.
S4	User switched focus to the client application using the fourth menu item/toolbar button.
S5	User switched focus to the client application using the fifth menu item/toolbar button.
0	User clicked Search Criteria button 1 on the Search Criteria and Document List dialog box.
1	User clicked Search Criteria button 2 on the Search Criteria and Document List dialog box.
2	User clicked Search Criteria button 3 on the Search Criteria and Document List dialog box.
3	User clicked Search Criteria button 4 on the Search Criteria and Document List dialog box.
4	User clicked Search Criteria button 5 on the Search Criteria and Document List dialog box.
5	User clicked Document List button 5 on the Search Criteria and Document List dialog box.
6	User clicked Document List button 1 on the Search Criteria and Document List dialog box.
7	User clicked Document List button 2 on the Search Criteria and Document List dialog box.
8	User clicked Document List button 3 on the Search Criteria and Document List dialog box.
9	User clicked Document List button 4 on the Search Criteria and Document List dialog box.

The DDEML DdeClientTransaction function is used, with ADV_START and ADV_STOP transactions, to start and stop an Advise Loop. The DDEML item name string must contain "1". The client application may initiate an Advise Loop after establishing connection to Content Manager OnDemand and maintain it for the duration of the conversation or start and stop the loop at its discretion. Notification of events takes place only when a loop is active.

The client application receives notification of an event through an XTYP_ADVDATA transaction at its DDEML callback function. The data returned is a null-terminated character string containing the **Code** for the **Event**.

Chapter 10. External Applications and Dynamic Link Libraries

For the Windows client, Content Manager OnDemand provides menu and toolbar extensions that allow an end user to invoke another Windows application or execute a function in a Dynamic Link Library (DLL). This facility is activated by placing information in the Windows system registry. If during initialization, Content Manager OnDemand detects this information, it adds menu items and toolbar buttons. When the user chooses one of these menu items or clicks one of these toolbar buttons, Content Manager OnDemand invokes the associated application or calls the associated entry point in a DLL.

Content Manager OnDemand reads the HKEY_CURRENT_USER\Software\IBM\OnDemand32\Client\ExternalApps (or HKEY_LOCAL_MACHINE\Software\IBM\OnDemand32\Client\ExternalApps) key in the Windows system registry for the following keys and string values:

Table 1. External Applications and Dynamic Link Library Keys in the Registry

Key	Value Name	Value Data
ExternalApps	Apps	<i>a1</i> [<i>a2</i>][<i>a3</i>][<i>a4</i>][<i>a5</i>]
<i>a1</i>	Path	application path
	MenuText	menu item text
	BitmapDLL	toolbar button bitmap DLL path
	BitmapResid	toolbar button bitmap resource identifier
	Folders	folder name[\\folder name]...
	ExcludeFolders	folder name[\\folder name]...
	Doc	0 1 2 3
	CopyDoc	ASIS ASCII
	Parameter	parameter data
ExternalDlls	Dlls	<i>d1</i> [<i>d2</i>][<i>d3</i>][<i>d4</i>][<i>d5</i>]
<i>d1</i>	Path	DLL path
	Function	function name
	MenuText	menu item text
	BitmapDll	toolbar button bitmap DLL path
	BitmapResid	toolbar button bitmap resource identifier
	Folders	folder name[\\folder name]...
	ExcludeFolders	folder name[\\folder name]...
	Doc	0 1 2 3
	CopyDoc	ASIS ASCII
	Parameter	parameter data

A maximum of five applications and five DLLs may be specified. They may be divided between the HKEY_CURRENT_USER and HKEY_LOCAL_MACHINE keys. The key names may be any string. The values, which must be string values, are interpreted as follows:

- Path specifies the path for the application or DLL. This value must be provided and should be fully-qualified or in the execution path.
- Function specifies the name of an entry point in the DLL specified with Path. This value must be provided for a DLL. It is not relevant for an application.
- MenuText specifies the text to appear on the associated menu item under the Content Manager OnDemand Window menu. This value is optional. If not provided, the menu item is blank.
- BitmapDLL specifies the path for a DLL containing a bitmap resource to be used for a toolbar button to be associated with the application or DLL. This DLL may be the same as or different from any DLL specified for a Path DLL or another BitmapDLL DLL. The bitmap should be 16 pels wide and 16 pels high. This value is optional. If not provided, a toolbar button will not be created.
- BitmapResid specifies the resource id of the bitmap within the DLL specified for BitmapDLL. This value is ignored if BitmapDLL is not specified and optional if it is. If not provided, a value of 0 is assumed.
- Folders specifies one or more names of Content Manager OnDemand folders. If multiple names are provided, they must be separated by a backslash ('\') character. An asterisk (*) may be used as a wildcard character in the last position of the name. This is equivalent to listing all folder names beginning with the characters preceding the asterisk.

The associated menu item, and corresponding toolbar button, is enabled whenever: 1) a document is being viewed and the folder associated with the active document is one of the specified folders, or 2) a document is not being viewed and the current folder is one of the specified folders.

This value is optional. If the ExcludeFolders value is provided, this value is ignored. If neither is provided, a folder name test is not performed before enabling the menu item and toolbar button.

- ExcludeFolders specifies one or more names of Content Manager OnDemand folders. The syntax is the same as the Folders value.

The associated menu item, and corresponding toolbar button, is enabled whenever: 1) a document is being viewed and the folder associated with the active document is **not** one of the specified folders, or 2) a document is not being viewed and the current folder is **not** one of the specified folders.

This value is optional. If not provided, enablement is controlled by the Folders value.

- Doc may be specified as one of the following values:
 - 0 indicates that enablement of the associated menu item and corresponding toolbar button is limited only by the Folders and ExcludeFolders values.
 - 1 indicates that the associated menu item and corresponding toolbar button is enabled only when a document is being viewed.
 - 2 indicates that the associated menu item and corresponding toolbar button is enabled only when a document list is being viewed and at least one document is selected.
 - 3 indicates that the associated menu item and corresponding toolbar button is enabled only when a document is being viewed or a document list is being viewed and at least one document is selected.

This value is optional. If not provided, a value of 3 is assumed.

- CopyDoc indicates that a copy of one or more documents is to be provided to the external application or DLL and specifies the type of data to be provided. If the value is ASIS, the document data is in its native format. If the value is ASCII, the document data is converted to an ASCII file.

If the end user chooses the associated menu item or corresponding toolbar button when a document is being viewed, the data provided is for the active document. If the folder document list is being displayed, the data is a concatenation of all documents selected in the document list.

This value is optional. If not provided, no document is provided to the external application or DLL.

- Parameter specifies a maximum of 255 characters to be passed as parameter data to the external application or DLL. This value is optional. If not provided, no parameter data is passed to the external application or DLL.

The following three screens show an example of the required registry entries for External Applications.

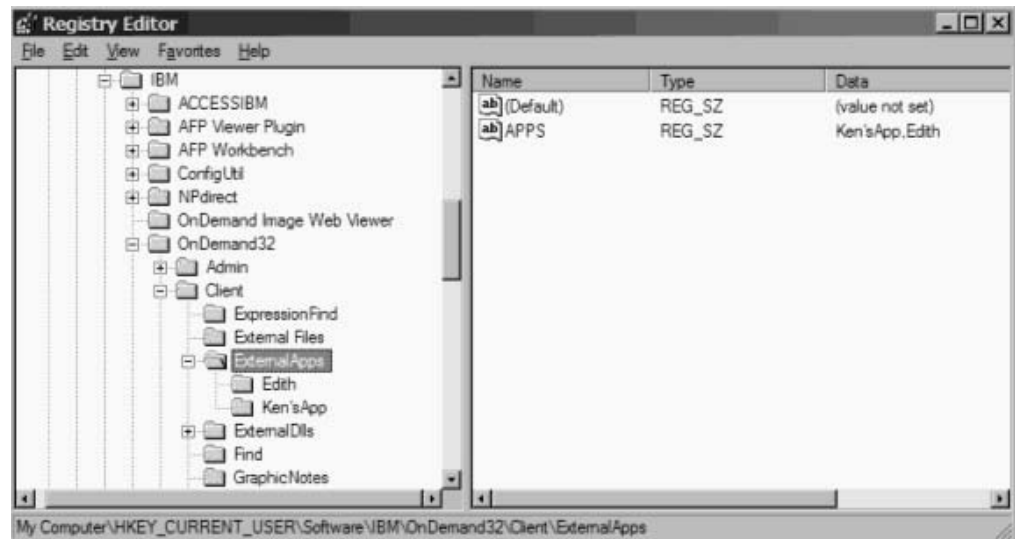


Figure 1. Example of External Applications in the Registry Part 1 of 3

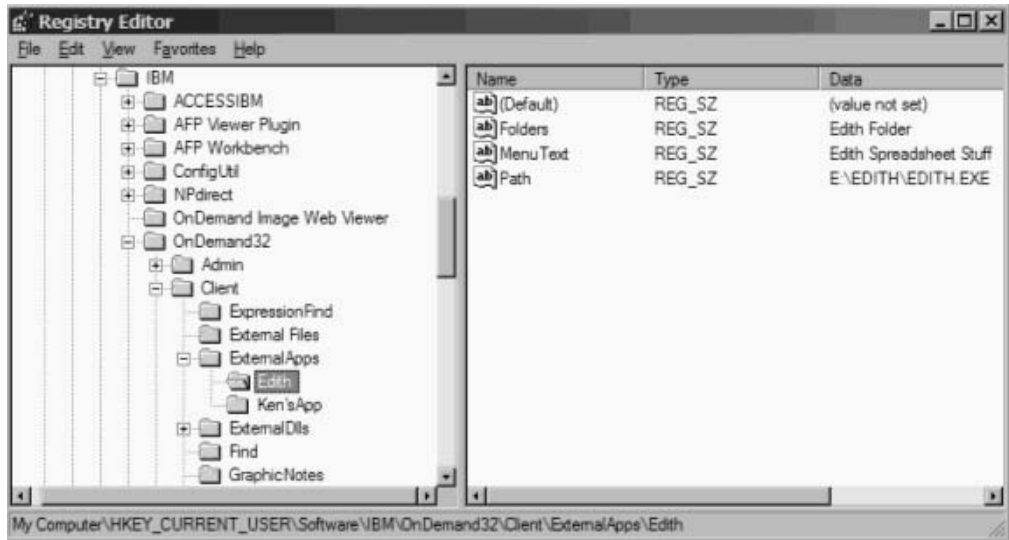


Figure 2. Example of External Applications in the Registry Part 2 of 3

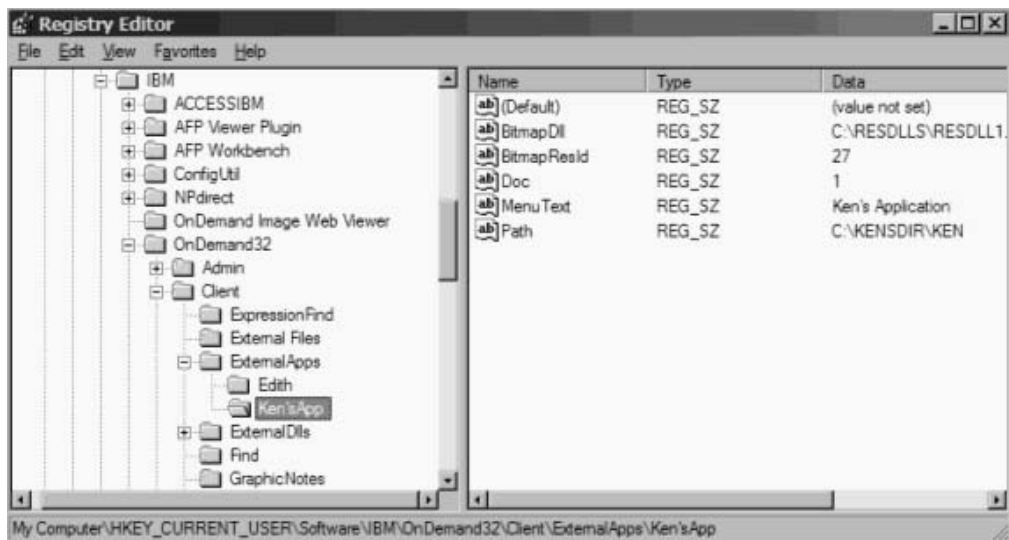


Figure 3. Example of External Applications in the Registry Part 3 of 3

The following two screens show an example of the required registry entries for External DLLs.

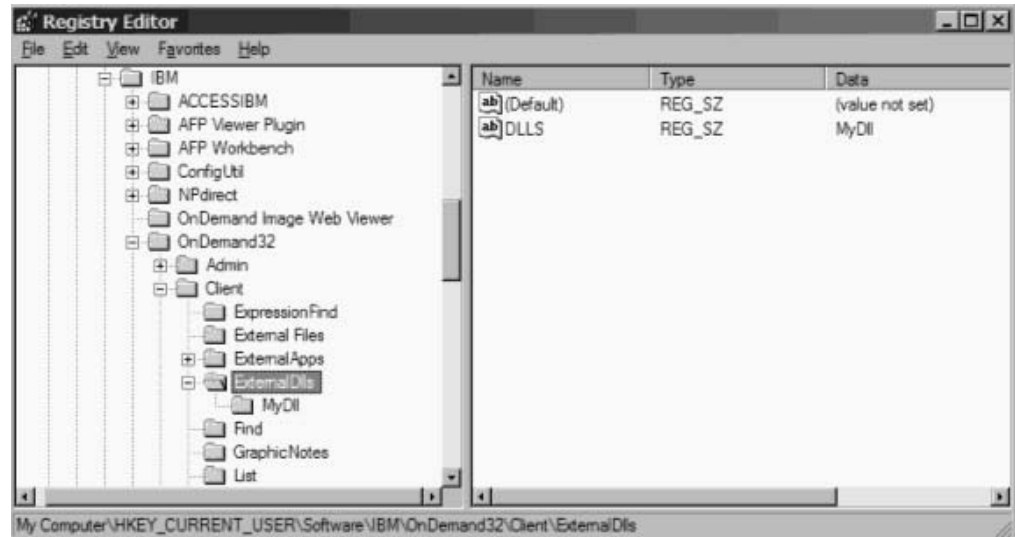


Figure 4. Example of External DLLs in the Registry Part 1 of 2

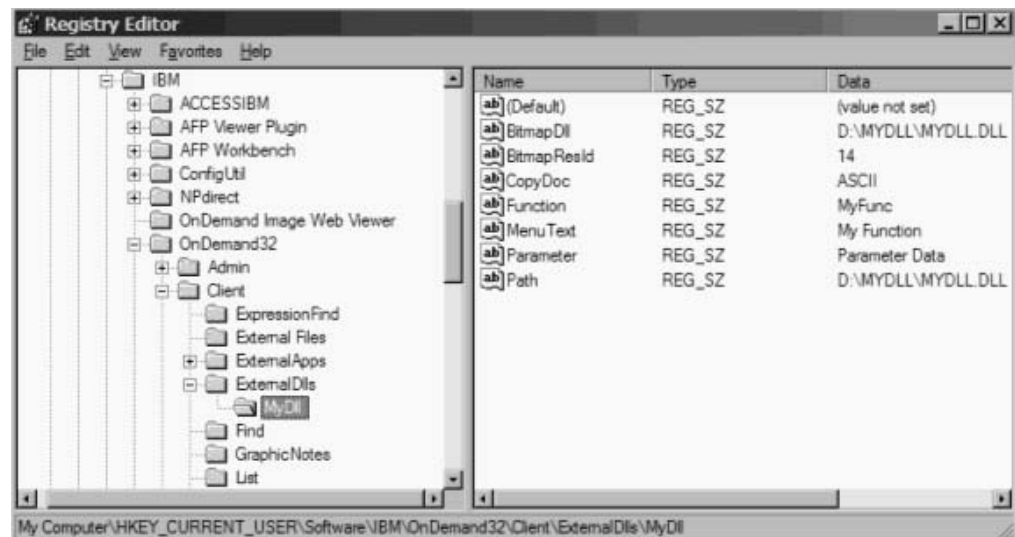


Figure 5. Example of External DLLs in the Registry Part 2 of 2

If the user chooses a menu item or clicks a toolbar button associated with an application, Content Manager OnDemand invokes the application with the command line:

application path /F folder /P page /T text /A attrvalue /D fields /N filename /R data

Where:

- application path is the path specified with the Path value in the registry.
- folder is the name of the active folder or, if a document is being displayed, the folder associated with that document. If no folder is open, the /F parameter is not provided.
- page is the current page of the document being viewed. If no document is being viewed or if the active folder window is being displayed, the /P parameter is not provided.
- text is the currently selected text in the document being viewed or the document list of the active folder. The format of the text is the same as that obtained by

copying it to the clipboard. It may contain tab and newline characters. If no such text exists, the /T parameter is not provided.

- attrvalue is the attribute/value pair associated with the current page of the document being viewed. The attribute and value are separated by a tab character. If no such attribute/value pair exists, the /A parameter is not provided.
- fields is a set of folder field name/value pairs associated with the document being viewed. Each pair contains the name of a folder field and, separated by a tab character, the value of that field for the document. The pairs are separated by a newline character. If no such text exists, the /T parameter is not provided.
- filename is the fully-qualified path and filename of document data as described for the CopyDoc registry entry. If the CopyDoc entry is not specified or no data is available, the /N parameter is not provided.
- data is the parameter data specified for the Parameter registry entry. If the Parameter entry is not specified, the /R parameter is not provided.

If the user chooses a menu item or clicks a toolbar button associated with a DLL, Content Manager OnDemand calls the entry point (function) specified with the Function value in the registry. This function must be of the following type:

```
typedef void ( WINAPI * ArsExternalDllFunction )  
            ( long    page_number,  
              char *  pFolderName,  
              char *  pSelectedText,  
              char *  pAttributeAndValue,  
              char *  pFieldsAndValues,  
              char *  pFilename,  
              char *  pParameterData );
```

Where:

- page_number is the current page of the document being viewed. If no document is being viewed or if the active folder window is being displayed, this value is 0.
- pFolderName is a pointer to a null-terminated character string containing the name of the active folder or, if a document is being displayed, the folder associated with that document. If no folder is open, this value is NULL.
- pSelectedText is a pointer to a null-terminated character string containing the currently selected text in the document being viewed or the document list of the active folder. The format of the text is the same as that obtained by copying it to the clipboard. It may contain tab and newline characters. If no such text exists, this value is NULL.
- pAttributeAndValue is a pointer to a null-terminated character string containing the attribute/value pair associated with the current page of the document being viewed. The attribute and value are separated by a tab character. If no such attribute/value pair exists, this value is NULL.
- pFieldsAndValues is a pointer to a null-terminated character string containing a set of folder field name/value pairs associated with the document being viewed. Each pair contains the name of a folder field and, separated by a tab character, the value of that field for the document. The pairs are separated by a newline character. If no such text exists, this value is NULL.
- pFilename is a pointer to a null-terminated character string containing the fully-qualified path and filename of document data as described for the CopyDoc registry entry. If the CopyDoc entry is not specified or no data is available, this value is NULL.

- `pParameterData` is a pointer to a null-terminated character string containing the parameter data specified for the Parameter registry entry. If the Parameter entry is not specified, this value may be `NULL` or point to an empty string.

Chapter 11. Related documents

For the Windows client, Content Manager OnDemand provides menu and toolbar extensions that allow an end user to retrieve and view a document related to the document currently being viewed. This facility is activated by placing information in the Windows system registry. If during initialization, Content Manager OnDemand detects this information, it adds menu items and toolbar buttons. When the user chooses one of these menu items or clicks one of these toolbar buttons, Content Manager OnDemand retrieves the related document and displays it along with the original document.

A typical application for Related Documents is a credit card folder containing monthly statements. The end user views a customer's statement which has a line for each credit card transaction, selects a transaction number on one of these lines, and clicks a toolbar button. Content Manager OnDemand searches an associated transaction folder for the customer's account number and selected transaction number, then displays the transaction document alongside the statement.

Content Manager OnDemand reads the HKEY_CURRENT_USER\Software\IBM\OnDemand32\Client (or HKEY_LOCAL_MACHINE\Software\IBM\OnDemand32\Client) key in the Windows system registry for the following keys and string values:

Table 2. Related Documents Keys in the Registry

Key	Value Name	Value Data
RelatedDocs	Related	<i>r1</i> [, <i>r2</i>][, <i>r3</i>][, <i>r4</i>][, <i>r5</i>][, <i>r6</i>][, <i>r7</i>][, <i>r8</i>][, <i>r9</i>][, <i>r10</i>]
<i>r1</i>	MenuText	menu item text
	BitmapDLL	toolbar button bitmap DLL path
	BitmapResid	toolbar button bitmap resource identifier
	Folders	folder name[\folder name]...
	ExcludeFolders	folder name[\folder name]...
	RelatedFolder	related folder name
	Fields	field information
	Arrange	V H M C U

The RelatedDocs key contains one string value. The string name is Related. The value data for the Related string value is a comma separated list of additional Client keys that represent the related documents. Each related document that is specified in the Related string value should be added as a key to the Client key. A maximum of ten related documents may be specified. They may be divided between the HKEY_CURRENT_USER and HKEY_LOCAL_MACHINE keys. The key names may be any string.

The values, which must be string values, are interpreted as follows:

- **MenuText** specifies the text to appear on the associated menu item under the Content Manager OnDemand Window menu. This value is optional. If not provided, the menu item is blank.
- **BitmapDLL** specifies the path for a DLL containing a bitmap resource to be used for a toolbar button to be associated with the Related Documents. The bitmap should be 16 pels wide and 16 pels high.

This value is optional. If not provided, a toolbar button will not be created.

- **BitmapResid** specifies the resource id of the bitmap within the DLL specified for **BitmapDLL**. This value is ignored if **BitmapDLL** is not specified and optional if it is. If not provided, a value of 0 is assumed.
- **Folders** specifies one or more names of Content Manager OnDemand folders. If multiple names are provided, they must be separated by a backslash ('\') character. An asterisk (*) may be used as a wildcard character in the last position of the name. This is equivalent to listing all folder names beginning with the characters preceding the asterisk.

The associated menu item, and corresponding toolbar button, is enabled whenever: 1) a document is being viewed and the folder associated with the active document is one of the specified folders, or 2) a document is not being viewed and the current folder is one of the specified folders.

This value is optional. If the **ExcludeFolders** value is provided, this value is ignored. If neither is provided, a folder name test is not performed before enabling the menu item and toolbar button.

- **ExcludeFolders** specifies one or more names of Content Manager OnDemand folders. The syntax is the same as the **Folders** value.

The associated menu item, and corresponding toolbar button, is enabled whenever: 1) a document is being viewed and the folder associated with the active document is **not** one of the specified folders, or 2) a document is not being viewed and the current folder is **not** one of the specified folders.

This value is optional. If not provided, enablement is controlled by the **Folders** value.

- **RelatedFolder** specifies the name of an Content Manager OnDemand folder which contains a document related to the document being viewed. This folder may be the same as or different from the folder specified by **Folders**.
- **Fields** specifies information describing the document to be retrieved from the related folder. This information is used to perform a search of the folder. The folder fields are first initialized to their default operator and values. The fields specified here are then set to the operator and values requested. The first document in the document list resulting from the search is considered to be the related document.

The information has the following format:

```
fieldname=operator\value1[\value2]; ... ;fieldname=operator\value1[\value2]
```

Where:

- **fieldname** is the name of a field defined for the related folder.
- **operator** is the search operator to be used for the field. It must be one of the following values:

EQ	for Equal
NE	for Not Equal
GT	for Greater Than
GE	for Greater Than Or Equal
LT	for Less Than
LE	for Less Than Or Equal
BW	for Between

- NB for Not Between
 - IN for In
 - NI for Not In
 - LK for Like
 - NL for Not Like
- value1 is the first or only value to be used for the field. It may be any string, including an empty string, and may contain the following substitution characters.
 - %v If the original folder contains a field of the same name, the value of that field for the original document is substituted in place of these characters; otherwise, an empty string is substituted.
 - %s These characters are replaced by any text which the user has currently selected in the original document.
 - %% These characters are replaced by a single %.
 - value2 is the second value to be used for the field. It is ignored unless the operator is Between or Not Between. The format is the same as value1.
 - Arrange specifies the arrangement of the document windows after the related document is brought into view. The value must be one of the following:
 - V for tiled vertically.
 - H for tiled horizontally.
 - M for maximized with the related document overlaying the original.
 - C for cascaded.
 - U for left to the user's specification via the Maintain Document Arrangement menu item.

The following three screen captures show an example of registry entries required to support two related documents.

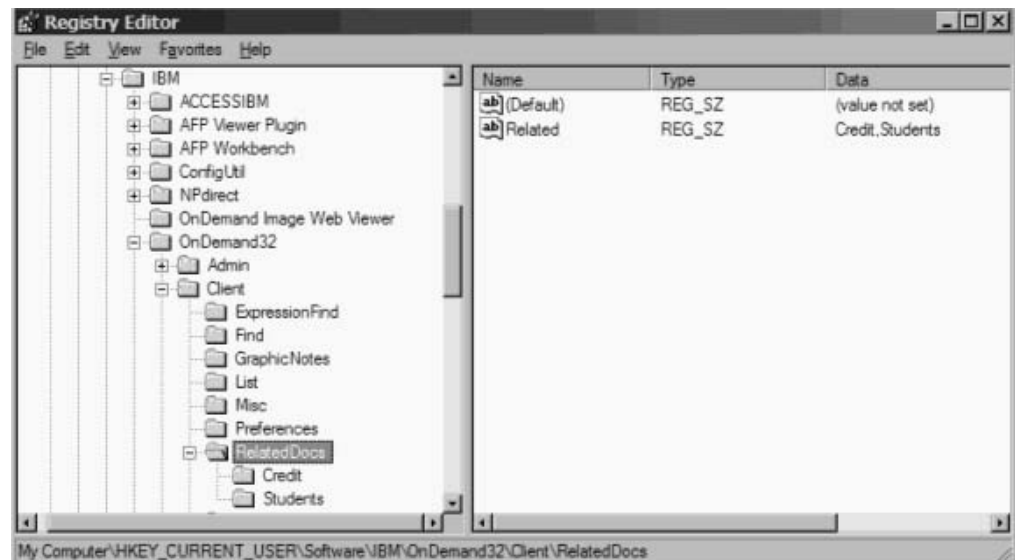


Figure 6. RelatedDocs key showing values for two related documents

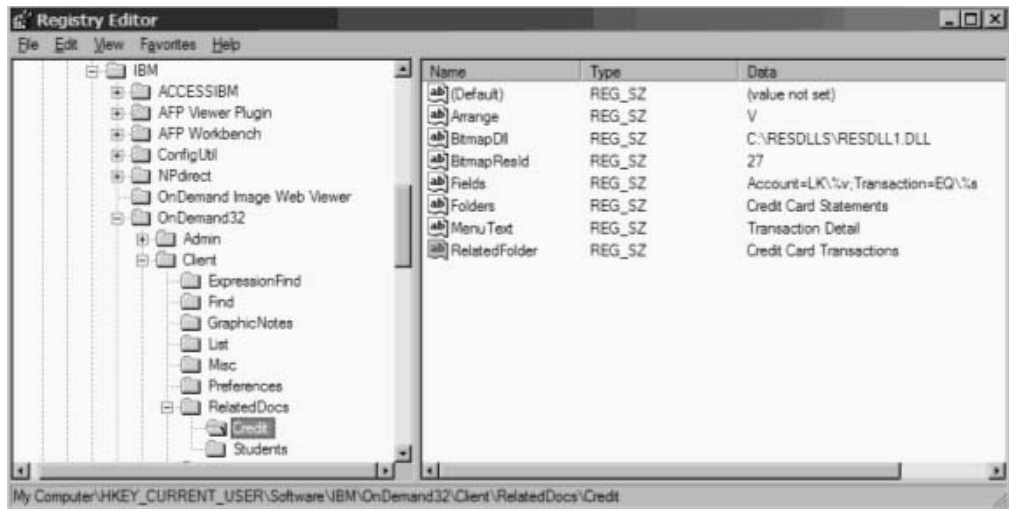


Figure 7. Credit key showing values for the first related document

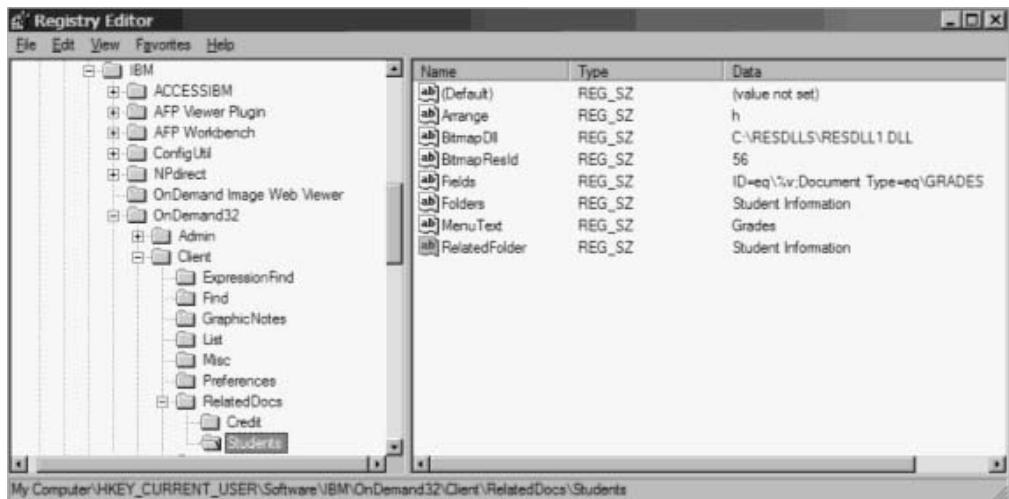


Figure 8. Students key showing values for the second related document

Chapter 12. Program Information File

A Program Information File (PIF) can be created to customize the Content Manager OnDemand application title and the appearance of the “About” dialog box that appears during initialization and when the user selects the About item in the Help menu.

A PIF has the name PRODUCT.INF and is found in the same directory from which Content Manager OnDemand is executed. The PIF has the same format and syntax as a standard Windows INI file. It contains a single section named PRODUCT. The section contains the following entries:

- NAME specifies the title to be used at the top of the main Content Manager OnDemand window and on a number of menu items.
- LOGO_FILE specifies a fully-qualified path of a Device Independent Bitmap (DIB) file to be used as the logo in the About dialog box.
- ABOUT_TITLE specifies a title to be used for the About dialog box.
- ABOUT_LINE*n* specifies each line of the About dialog box text. ABOUT_LINE1 gives the first line; ABOUT_LINE2 gives the second; and so forth through ABOUT_LINE8.

Chapter 13. Document Audit Facility

Overview

The Document Audit Facility (DAF) can be used to audit documents stored in Content Manager OnDemand. To use the DAF, you must first create a control file and define the reports to be audited to OnDemand. After you load the reports into the system, you can use the Windows client to audit the documents. When you retrieve a document from a folder defined in the DAF control file, OnDemand displays two additional command buttons on the client viewing window. One button is used to mark documents that pass the audit. The other button is used to mark documents that fail the audit.

Note: Users that need to audit documents must be given permission to update documents. See “Controlling access to the DAF” on page 191 for more information.

The following topics contain additional information:

- Creating the DAF control file
- Defining the report
- Controlling access to the DAF
- Using the DAF

Creating the DAF control file

The DAF is controlled by a file named ARSGUI.CFG, which you must create and store in the Windows client program directory (\Program Files\IBM\OnDemand32 by default). The DAF file has the same format and syntax as a standard Windows INI file. The DAF file contains a section named AUDIT, which identifies one or more folder sections. Each folder section identifies a folder that can be audited. Figure 9 shows a sample DAF file.

```
[AUDIT]
FOLDERS=LDR,Student Information

[LDR]
FOLDER=Loan Delinquency Report
AUDIT_FIELD=Document Audit
PASS_TEXT=Pass
FAIL_TEXT=Fail
PASS_VALUE=P
FAIL_VALUE=F

[Student Information]
FOLDER=Student Information
AUDIT_FIELD=Audit Status
PASS_VALUE=P
FAIL_VALUE=F
```

Figure 9. Sample DAF (ARSGUI.CFG) File

The AUDIT section

The AUDIT section contains one record, the FOLDERS record. The FOLDERS record contains a comma-separated list of folder section names. You must create an

additional section in the DAF file for each folder section named in the FOLDERS record. The total number of characters in the FOLDERS record must not exceed 255.

The folder section

Each folder section contains the following records:

- FOLDER specifies the name of the folder, exactly as it appears in OnDemand. The FOLDER record is required.
- AUDIT_FIELD specifies the name of the folder field used to audit documents, exactly as it appears in OnDemand. See “Defining the Folder” on page 191 for more information. The AUDIT_FIELD record is required.
- PASS_TEXT is the caption that appears on the command button used to mark a document that passes an audit. The total number of characters in the PASS_TEXT record must not exceed 50. The PASS_TEXT record is optional. The default caption is Pass.
- FAIL_TEXT is the caption that appears on the command button used to mark a document that fails an audit. The total number of characters in the FAIL_TEXT record must not exceed 50. The FAIL_TEXT record is optional. The default caption is Fail.
- PASS_VALUE is the value that is stored in the database for documents that pass an audit. The value is stored in the application group field. See “Defining the application group” for more information. The total number of characters in the PASS_VALUE record must not exceed 254. The PASS_VALUE record is required.
- FAIL_VALUE is the value that is stored in the database for documents that fail an audit. The value is stored in the application group field. See “Defining the application group” for more information. The total number of characters in the FAIL_VALUE record must not exceed 254. The FAIL_VALUE record is required.

Defining the report

The following topics provide information that you need to specify when you define a report that uses the DAF. The information provided is in addition to all of the other attributes you need to specify when you define a report to OnDemand.

- Defining the application group
- Defining the application
- Defining the folder

Defining the application group

Add the audit field to the application group on the Field Definition page. The field type must be STRING.

Define the attributes of the audit field on the Field Information page.

- In the String area, set the Case to Upper, Type to Fixed, and Length to 1 (one).
- In the Mapping area, add the Database and Displayed Values for the audit field. Database values are stored in the database and Displayed Values appear in search fields and the document list. Enter a Database Value and its corresponding Displayed Value in the spaces provided. Click Add to add each pair of values to the application group.

You must add one set of values for documents that pass an audit (PASS), one set of values for documents that fail an audit (FAIL), and one set of default values:

- The PASS Database Value must match the value of the PASS_VALUE record in the folder section of the DAF file. We recommend that the PASS Displayed Value match the value of the PASS_TEXT record in the folder section of the DAF file.
- The FAIL Database Value must match the value of the FAIL_VALUE record in the folder section of the DAF file. We recommend that the FAIL Displayed Value match the value of the FAIL_TEXT record in the folder section of the DAF file.
- The default values are used to set the status of all documents when a report is loaded into the application group. You typically set the default values to N, for None or Not Audited.

Table 3 shows an example.

Table 3. Database and Displayed Values

Database Value	DAF File	Displayed Value	DAF File
F	FAIL_VALUE=F	Fail	FAIL_TEXT=Fail
P	PASS_VALUE=P	Pass	PASS_TEXT=Pass
N		Not Audited	

Note: The N Database Value and the Not Audited Displayed Value are not stored in the DAF file.

Defining the application

Add the default value for the audit field on the Load Information page. This is the value that OnDemand stores in the database for all documents when a report is loaded into the application group. We recommend that you set the default value to N (for None or Not Audited).

The default value must match the default Database Value for the audit field. You defined the default Database Value for the audit field on the application group Field Information page. See “Defining the application group” on page 190 for more information.

Defining the Folder

Add the audit field to the folder on the Field Definition page. The field type must be STRING. The name of the field must match the value of the AUDIT_FIELD record in the folder section of the DAF file. See “The folder section” on page 190 for more information.

Map the folder audit field to the application group audit field on the Field Mapping page.

Controlling access to the DAF

There are several ways you can manage access to documents that need to be audited. However, to simplify administration of the system, we strongly encourage you to use groups. For example, some customers will define two groups:

- Viewers. At a minimum, users in the Viewers group are not permitted to audit documents. In addition, some customers create a default logical view so that the audit field does not appear when users in the Viewers group open a document. Other customers define a query restriction so that users in the Viewers group see only documents that have passed an audit.

- **Auditors.** The users in the Auditors group are permitted to audit documents. Users in the Auditors group must be given permission to update documents in the application groups that contain documents to be audited.

If the requirements of your system are not met by these two groups, you may need to define additional groups or configure the system differently. Contact the IBM support center if you have questions about users, groups, or other aspects of administering the system.

Using the DAF

After a report is loaded into OnDemand, authorized users can use the DAF to audit the documents. To audit a document, open one of the folders defined in the DAF control file. Search the folder for documents that need to be audited. For example, search for documents that contain the value `Not Audited` in the audit field. Select and view one or more documents from the document list. In the document viewing window, click the **Pass** button to mark a document that passes the audit; click the **Fail** button to mark a document that fails the audit. OnDemand updates the database with the pass/fail value specified in the DAF control file.

Chapter 14. Modifying client behavior through the Registry

Note: The following procedure requires that you edit the registry on the computer. You should not edit the registry unless it is absolutely necessary. If there is an error in the registry, the computer may not function properly. Before you proceed, you should make a backup copy of the registry and you should be familiar with how to restore the registry to the same version you were using when you last successfully started the computer. For instructions, see your Windows information.

The following registry values are available to modify the behavior of the Windows client. The values must be placed in this registry key:

HKEY_CURRENT_USER\Software\IBM\OnDemand32\Client\Preferences

FORCE_DEF_CHARSET	If set to 1, this value forces the use of the default character set when selecting the document list font. Default value: 0
GREEN_BAR_COLOR	May be used to override the color of the “green” bands of the Green Bar background color. It must be specified as an RGB value. For example, to use gray bars instead of green, specify 192,192,192. Default value: 128,255,128
LINEDATA_FONT_FACE_1 LINEDATA_FONT_FACE_2 LINEDATA_FONT_FACE_3	May be used to override the initial font used for line data documents. The first time that a line data document is opened after OnDemand is initially installed, the client selects a font. If any of these values are specified, OnDemand will search first for the font face name specified by LINEDATA_FONT_FACE_1, then for LINEDATA_FONT_FACE_2, and finally LINEDATA_FONT_FACE_3 before trying other names. For example, for LINEDATA_FONT_FACE_1, specify FixedSys
MAXIMIZE_WINDOW	If set to 1, maximizes the client window. Default value: 0
MULTIFIND	If set to 1, when performing a text find operation in an AFP document, only the first occurrence of the string on a page is highlighted. For an AFP document, you can modify the value of MULTIFIND in the registry key HKEY_CURRENT_USER\Software\IBM\OnDemand32\Client\Misc instead of HKEY_CURRENT_USER\Software\IBM\OnDemand32\Client\Preferences. Default value: 0
SCS_CODEPAGE	May be used to override the code page used for

	SCS data by the AFP Viewer. If this value is 0, the code page used is language dependent. Default value: 0
SUPPRESS_FILE_CLEANUP	If set to 1, prevents the deletion of obsolete files in data and resource directories during client initialization. Default value: 0
SUPPRESS_WIN_POS_SAVE	If set to 1, prevents the saving of window position and size during client termination. Default value: 0
TRACE	If set to 1, causes a trace file to be produced during AFP Viewer operations. The file is named VIEWER.LOG and is placed in the OnDemand project directory. Default value: 0
TTONLY	If set to 1, forces the AFP Viewer to use only True Type fonts. Default value: 0
UNDEFINED_CP	May be used to override the character displayed by the AFP Viewer in place of undefined code points. It must be specified as a hexadecimal character. For example, specify 2E for a period. Default value: 20

Chapter 15. Integration with Monarch Version 5

This section provides information about how to integrate Monarch Version 5 with the OnDemand Windows client. This function allows users to automatically load documents from the OnDemand client into Monarch. The user can then do complex data manipulation from Monarch, such as creating derived columns and generating charts and reports.

Note: Monarch is a software program that is available from Datawatch Corporation.

This section is of primary interest to administrators responsible for installing, configuring, and distributing software products. This section shows the steps that you typically need to take to integrate Monarch with OnDemand. This section describes how to do some of the tasks, but you will need other OnDemand information and your Monarch information to do others.

An administrator can use the OnDemand Windows 32-bit client installation program to distribute Monarch files with the OnDemand client and configure workstations to run Monarch from the OnDemand client. We recommend that an administrator configure a copy of the OnDemand client software on a reference workstation¹ and configure the installation program on a distribution server.² Each user planning to run Monarch from OnDemand must have the Monarch software installed on their workstation before they install or upgrade the OnDemand client with this function.

An administrator can store the Monarch files in the OnDemand Windows client installation directory tree on a distribution server. The installation program copies the Monarch files along with the standard OnDemand client files when the user installs the client from the server to the workstation. You can distribute several types of files, including:

- Registry files. The installation program imports these files into the Registry on the user's workstation. A Registry file named ODMonarch.Reg that contains information about the Monarch DLL for the OnDemand client must be present for the installation program to integrate OnDemand and Monarch on the user's workstation.

Important: It is easy to accidentally destroy important data and render a system completely unusable by importing Registry files. Plan to backup the Registry on the user's workstation before they run the Setup program.

- Monarch model files. The installation program copies these files to the Monarch\Models directory on the user's workstation.

The following topics provide additional information:

- Before you begin

1. A reference workstation contains an installed copy of the OnDemand client with the Monarch DLL defined to OnDemand. An administrator uses configuration information from the reference workstation to distribute software to other users.

2. A distribution server is a network file server that contains a copy of the OnDemand client installation software in a shared location. Other users on the network use the copy to run the OnDemand installation program and install the client from the distribution server to their workstations.

- Configuring the OnDemand client software
- Configuring the OnDemand Setup program
- Running the OnDemand Setup program
- Running Monarch from OnDemand
- Upgrading the client

Before you begin

Important: If your organization has integrated Monarch with OnDemand Version 2.2.1.2 or earlier, please see “Upgrading your OnDemand client” on page 204.

Before you continue, you should have already completed the following tasks:

- Installed Monarch Version 5 on the reference workstation. See your Monarch information for details.
- Obtained the latest PTF for the OnDemand Windows client from IBM service on the World Wide Web. To do so, point your Web browser to:

`ftp://www.service.software.ibm.com/software/ondemand/fixes`

Then follow the links to the latest PTF for the client. Click on the `odwin32.zip` file and save it to disk.

Configuring the client

This chapter describes how to configure the client on the reference workstation with the information needed to integrate Monarch with OnDemand. If you have not done so already, install a copy of Monarch Version 5 (see your Monarch information for details). Then install a copy of the OnDemand client software (see your other OnDemand information for details). Next, add the Registry key, values, and value data that define the properties of the Monarch DLL to the OnDemand client. Then verify that you have correctly integrated Monarch with OnDemand by starting the OnDemand client, opening a document, and invoking Monarch using the associated menu command or toolbar button. When you are satisfied that everything is working correctly, export the Registry key to a file. The following topics provide more information:

- Adding the Registry key
- Exporting the Registry key
- Using multiple Monarch model files

Adding the Registry key

Note: You must install the OnDemand Windows client on the reference workstation before you add the Registry key.

Placing information in the Registry activates Monarch integration with the OnDemand client. If during initialization, OnDemand detects this information, it adds menu items and toolbar buttons to the client workspace. When the user chooses one of the menu items or clicks one of the toolbar buttons, OnDemand calls the associated entry point in the Monarch DLL. OnDemand looks in the Registry under the **ExternalDlls** subkey to determine the action to take and other information. Figure 10 on page 197 shows an example of the ExternalDlls subkey.

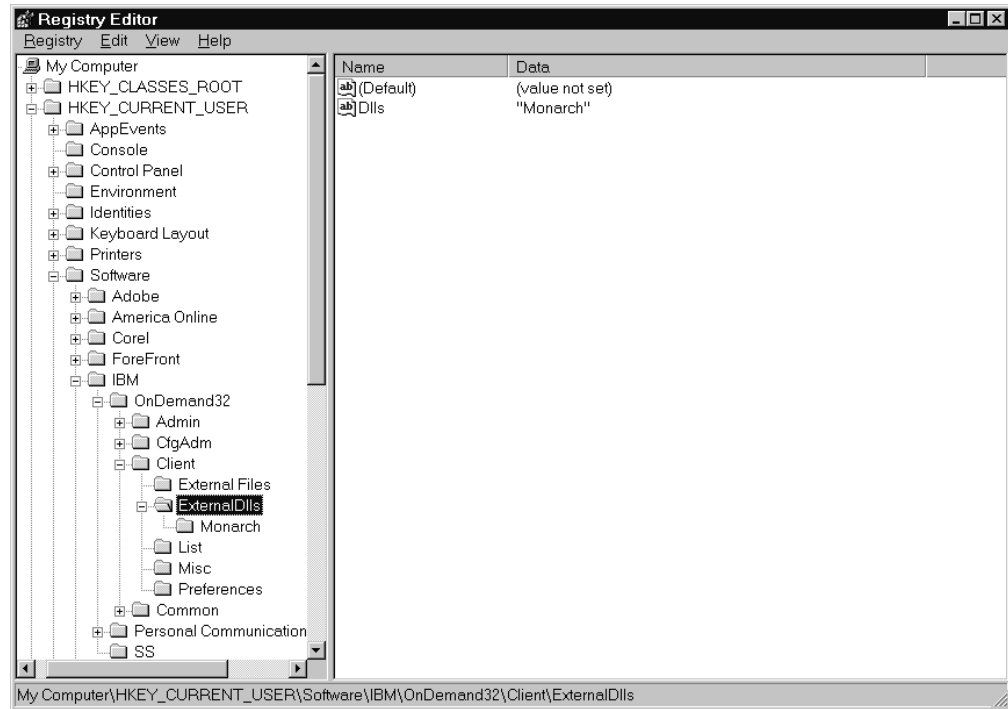


Figure 10. ExternalDlls subkey showing one value in Dlls

Subkeys under ExternalDlls contain the information used by OnDemand to call the DLLs integrated into the client. Figure 11 on page 198 shows an example of a subkey with which a user can invoke the new model Monarch DLL from the client. Since the subkey doesn't specify the name of a model file to run, OnDemand simply starts Monarch with the current document.

Note: When a user invokes Monarch from the client and no model file is specified in the Registry key, OnDemand starts Monarch with the current document. The user can then use Monarch functions to analyze the data and optionally create and save a model. To subsequently run a model that was created and saved from the client, the model must be identified in Dlls and a subkey for the model must be added under ExternalDlls. The subkey for the model must specify the values that are required to run the model from the client. For example, the Parameter value must contain the full path name of the model file and any Monarch parameters that are required to run the model.

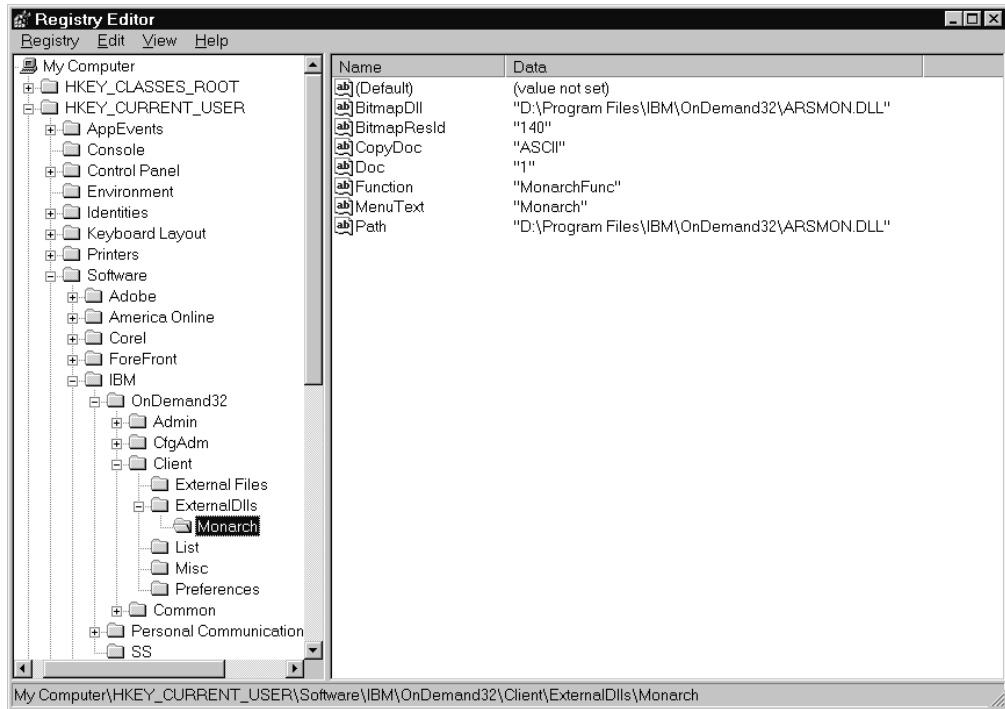


Figure 11. Subkey and values for the Monarch DLL

Table 4 provides information about the subkey values and value data shown in Figure 11. Table 5 on page 200 provides information about other values (that are not depicted in Figure 11) that you may specify.

Table 4. Monarch DLL subkey values

Value and Data	Description
BitmapDll C:\Program Files\IBM\OnDemand32\ARSMON.DLL	The full path name of the DLL file that contains the bitmap resource for the toolbar button used to invoke Monarch from the OnDemand client. The example uses the default provided by IBM. However, this DLL file may be different from the DLL file specified for the Path or you can use a different BitmapDll DLL file. The bitmap should be 16 picture elements (pel) wide and 16 pels high. This value is optional. If not provided, a toolbar bitmap will not be created.
BitmapResId 140	The resource identifier of the bitmap within the DLL file specified for BitmapDll. This value is ignored if BitmapDll is not specified and optional if it is. If not provided, a value of 0 (zero) is assumed.

Table 4. Monarch DLL subkey values (continued)

Value and Data	Description
CopyDoc ASCII	Indicates that a copy of one or more documents is to be provided to Monarch and specifies the type of data to be provided. If the value is ASCII, the document data is converted to an ASCII file. If the value is ASIS, the document data is in its native format. If the user chooses the menu item or toolbar button when a document is being viewed, the data provided is for the current document. If the document list is being displayed, the data is a concatenation of all documents selected in the document list. This value is optional. If not provided, no document is provided to Monarch.
Doc 1	May be specified as one of the following values: 0 (zero) indicates that enabling the associated menu item and corresponding toolbar button is limited only by the Folders and ExcludeFolders values. See for information about the Folders and ExcludeFolders values. 1 (one) indicates that the menu item and toolbar button are enabled only when a document is being viewed. 2 (two) indicates that the menu item and toolbar button are enabled only when the document list is being viewed and at least one document is selected. 3 (three) indicates that the menu item and toolbar button are enabled only when a document is being viewed or a document list is being viewed and at least one document is selected. This value is optional. If not provided, a value of 3 (three) is assumed.
Function MonarchFunc	The name of the entry point into the Monarch DLL file. This value must be provided.
MenuText Monarch	The text that appears on the menu item used to invoke Monarch from the OnDemand client. The menu item appears under the Window menu. The text also appears when the user passes the mouse pointer over the associated toolbar button. note. The text you specify can describe a specific application in Monarch, such as Loan Analysis. This value is optional. If not provided, the menu item is blank.
Path C:\Program Files\IBM\OnDemand32\ARSMON.DLL	The full path name of the Monarch DLL file on the user's workstation. The example shows the default installation drive and directory. This value must be provided.

Table 5. Other DLL subkey values

Value	Description
ExcludeFolders	Specified the names of one or more OnDemand folders. The syntax is the same as the Folders value. The menu item and toolbar button are enabled whenever: • A document is being viewed and the folder associated with the document is not one of the specified folders, or • A document is not being viewed and the current folder is not one of the specified folders. This value is optional. If not provided, enabling is controlled by the Folders value.
Folders	Specifies the names of one or more OnDemand folders. If you specify more than one name, you must separate the names with the backslash (\) character. An asterisk (*) character may be used as a wildcard character in the last position of the name. (This is equivalent to listing all of the folder names that begin with the characters preceding the asterisk.) The menu item and toolbar button are enabled whenever: • A document is being viewed and the folder associated with the document is one of the specified folders, or • A document is not being viewed and the current folder is one of the specified folders. This value is optional. If the ExcludeFolders value is provided, this value is ignored. If neither is provided, a folder name test is not performed before enabling the menu item and toolbar button.
Parameter	Specifies a maximum of 255 characters to be passed as parameter data to Monarch. This value is optional. If not provided, then no parameter data is passed to Monarch. See your Monarch information for a list of the parameters that you can specify. Note: To run a specific Monarch model, the Parameter value must contain the full path name of the model file and any Monarch parameters that are required to run the model. If you do not specify the Parameter value, then OnDemand simply starts Monarch with the current document. The user can then use Monarch functions to analyze the data and optionally create and save a model.

To add the Registry key:

- The ExternalDlls subkey contains a list of the DLLs that are integrated with the OnDemand client. It also contains subkeys that define the properties of each DLL. Add the ExternalDlls subkey to the following subkey:
HKEY_CURRENT_USER\Software\IBM\OnDemand32\Client
- Add the value Dlls to the ExternalDlls subkey. Dlls must have a type of String (REG_SZ). The data value of Dlls is a comma-separated list of identifiers for the DLLs that are integrated with the OnDemand client. Each identifier in the list must have an associated subkey under ExternalDlls. In , the identifier is Monarch.
- Define each DLL listed in Dlls by adding a subkey under the ExternalDlls subkey. (Each subkey that you add must be listed in Dlls.) In , we added the Monarch subkey.

- Define the properties of a DLL by adding string values to the subkey that you added. In the example (see Figure 12 on page 202), we added seven string values. To run a specific Monarch model, the Parameter subkey value must contain the full path name of the model file and any Monarch parameters that are required to run the model.

Exporting the Registry key

After you have added the Registry key that defines the properties of the Monarch DLL to the OnDemand Windows 32-bit client (and tested that you can invoke Monarch from the client), you can export it to a file. Later, you will store the file in the CUSTOM subdirectory tree under the OnDemand Windows 32-bit client installation directory tree on a distribution server. Once you have placed the file there, users installing the OnDemand client will automatically have the Registry key imported into the Registry on their workstation when they run Setup from the distribution server. If you have questions about the Registry key, see “Adding the Registry key” on page 196.

Important: If you have other applications integrated with OnDemand (that is, you have DLLs from applications other than Monarch defined under the ExternalDlls subkey), contact the IBM support center before you proceed.

To export the Registry key, do the following:

1. Start the REGEDIT program.
2. Move to the HKEY_CURRENT_USER\IBM\OnDemand32\Client\ExternalDlls key.
3. From the Registry menu, select Export Registry File.
4. Select a directory to hold the file.
5. In the File Name field, enter: ODMonarch.Reg
6. Under Export Range, select Selected Branch.
7. Click Save.

Using multiple Monarch model files

“Adding the Registry key” on page 196 describes how to configure the client on the reference workstation to run a single instance of Monarch. You can configure the client to run multiple instances of Monarch, with each instance using a different model file. To configure the client, add an identifier for each instance of Monarch to the Dlls string value and add a subkey under the ExternalDlls subkey for each identifier. Each subkey should represent a different model file that you want to let your users run from the client. This section provides information to help you configure the client with multiple model files.

For each model file that you want to run from the client:

- Add an identifier to the Dlls string value under the ExternalDlls subkey. Add a unique identifier for each model file. Figure 12 on page 202 shows an example of the ExternalDlls subkey with multiple identifiers. (We added the identifiers Model1, Model2, and Model3.)

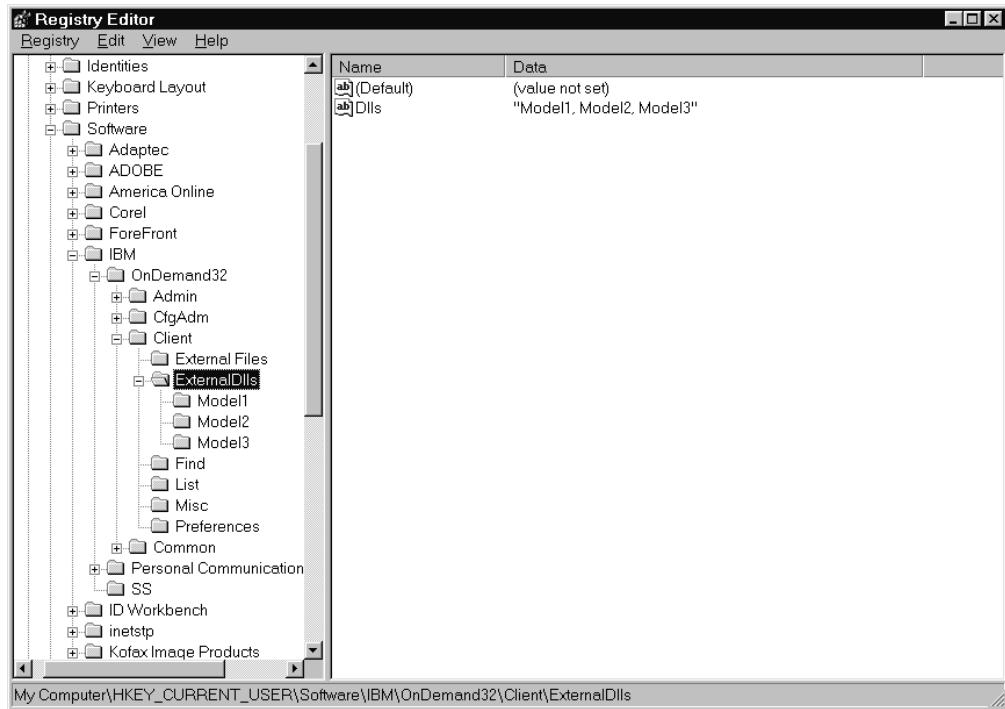


Figure 12. ExternalDlls subkey showing multiple values in Dlls

- Define the DLL to the client by adding a subkey under the ExternalDlls subkey. Each subkey that you add must be listed in Dlls. Define the properties of the DLL by adding string values to the subkey that you added:
 - Use the Parameter value to specify the full path name of the model file. To run a specific Monarch model, the Parameter value must contain the full path name of the model file and any Monarch parameters that are required to run the model. See your Monarch information for a list of the parameters that you can specify.
 - You can use the MenuText value to provide specific menu text and hover help for each model file
 - You can use the BitMapDll value to specify a different toolbar button bitmap for each model file

See Table 4 on page 198 and Table 5 on page 200 for help with the properties that you can specify.

When you have finished defining the model files to the client, export the Registry key. Use the process described in “Exporting the Registry key” on page 201.

Configuring Setup

This topic describes how to configure the client installation program to copy your Monarch model files to the user's workstation and import the Registry key into the Registry on the user's workstation. To begin, extract the latest version of the OnDemand Windows 32-bit client software from the ZIP file to a directory on the distribution server. Then add the custom directories to the OnDemand Windows 32-bit client installation directory tree on the distribution server. Next, copy the Monarch files to the custom directories. Then share the installation directory tree to make it available to your users. The following topics provide more information:

- Copying client software to the distribution server

- Adding custom directories
- Copying the Monarch files to the distribution server
- Sharing the installation directory

Copying client software

Note: See “Before you begin” on page 196 for information about obtaining the latest OnDemand Windows client software.

To copy the client software to the distribution server:

1. Log on to the server with a userid that has administrator permissions.
2. Extract the contents of the OnDemand Windows 32-bit client ZIP file (odwin32.zip) to a directory on the server. For example, extract the files to the \OD2217 directory.

Important: Use a file extraction method that preserves the ZIP contents directory and file structure on the server.

When complete, the target directory (\OD2217 in the example) should contain the Setup program and files and the ARS directory tree.

Adding subdirectories

You must store the Monarch files in the CUSTOM subdirectory tree under the OnDemand Windows client directory tree on the distribution server. By default, the directory tree is ARS32. (If you followed the example in “Copying client software,” the client directory tree on the distribution server will be \OD2217\ARS32.) To add subdirectories to hold the Monarch files:

1. Create a CUSTOM directory under the Windows client directory. For example:

```
mkdir \od2217\ars32\custom
```
2. Add the \od2217\ars32\custom\registry directory.

This directory will hold the Registry file (ODMonarch.Reg) that you created in “Exporting the Registry key” on page 201. When the user runs the Setup program from the distribution server, it imports this file into the Registry on the workstation.

3. Add the \od2217\ars32\custom\monarch directory.

This directory will hold the Monarch model files that you need to distribute to your users. The Setup program copies the Monarch model files to the Monarch\Models directory on the user's workstation.

Copying Monarch files

After copying the OnDemand client software to the distribution server and creating the CUSTOM directories (see “Adding subdirectories”), copy the Monarch files to the subdirectories. For example:

- Copy the ODMonarch.Reg file that you created in “Exporting the Registry key” on page 201 to the \OD2217\ARS32\CUSTOM\REGISTRY directory
- Copy your Monarch model files to the \OD2217\ARS32\CUSTOM\MONARCH directory

Sharing the installation directory

After you copy the OnDemand client software to the distribution server, create the CUSTOM directories, and copy the Monarch files to the server, you must make the installation directories on the server available to your users. The procedure differs

for each network and operating system. However, you generally need to provide users with read-only access to the directories. If applicable for your network, share the folders by giving the folder location on the distribution server a network name (share name). For example, to permit users to run the Setup program from the \OD2217 directory on the server and access all of the files in the \OD2217\ARS32 directory tree, you could assign the share name ODMONAR to the \OD2217 directory.

Running Setup

Note: Before your users run the Setup program from the distribution server to install the OnDemand client and integrate Monarch with the client, they must have a copy of Monarch Version 5 installed on their workstations.

After you have configured the Setup program on the distribution server with the Registry key and your Monarch model files, you can test the installation by having a user run the Setup program from the distribution server.

Note: It is easy to accidentally destroy important data and render a system completely unusable by importing Registry files. Plan to backup the Registry on the user's workstation before they run the Setup program.

When the user selects the Typical option during install, the Setup program automatically copies the Monarch files to the workstation.

When the user selects the Custom option during install, the Setup program copies the Monarch files to the workstation if the user selects to install one of the clients. If the user does not select one of the clients, then the Setup program does not copy the Monarch files to the workstation. For example, if the user selects the Custom option and chooses to install only the Sonoran Fonts or the administrative client, the Setup program does not copy the Monarch files to the workstation; if the user selects the Custom option and chooses to install the US English client, then the Setup program copies the Monarch files to the workstation.

Running Monarch from OnDemand

After your users install the client from the distribution server, they should test that everything works correctly. Have them start the OnDemand client, open a document, and then start Monarch using the associated menu command or toolbar button. Monarch should start and load the document into a separate window.

Upgrading your OnDemand client

If your organization has integrated Monarch with OnDemand Version 2.2.1.2 or earlier, follow these important instructions before your users upgrade the client to OnDemand Version 2.2.1.3 or later:

- Copy your old Registry file (for example, ODMonarch.Reg) to the CUSTOM\REGISTRY directory on the distribution server.

If the old Registry file does not exist, follow the steps in to create one. Then copy the Registry file to the CUSTOM\REGISTRY directory on the distribution server.

- Copy your Monarch model files to the CUSTOM\MONARCH directory on the distribution server.

Chapter 16. Installing client software on a network

This section provides information about installing the client software to be shared by multiple users over a network. This section is of primary interest to administrators responsible for installing and configuring software products.

Sharing OnDemand clients among multiple users

There are several ways that you can install OnDemand client software:

Standard or Custom Install

Use to install the OnDemand client to a hard disk on the user's workstation. These users do not have to rely on a network file server to run the software. To run a standard install, start the Setup program and select the Typical option. A custom install is required if you need to install the administrative client on a workstation, if you need to install the OnDemand client in a language other than the default language set on the workstation. To run a custom install, start the Setup program and select the Custom option. Then select the components to install. The *User's Guide* describes the standard and custom installation procedures.

Citrix Server install

The OnDemand client can be installed on a Citrix XenApp server once, and shared among users from their remote workstations. This way there is no need to install and maintain the OnDemand client on each user's workstation. For instructions on configuring the OnDemand client for the Citrix environment, refer to the following technote: http://www-1.ibm.com/support/docview.wss?rs=129&context=SSEPCD&context=SSEPC6&context=SS3U2Y&q1=1306090&uid=swg21306090&loc=en_US&cs=utf-8?=en.

Distribution install

Use to copy the OnDemand client software to a hard disk on a network file server. Users can then use the copy on the server to perform subsequent standard, custom and distribution installs. A distribution install is also useful if you need to add user-defined files to the client installation. "Distribution install" on page 206 describes the distribution installation procedure.

Installation directories

The OnDemand client CD-ROM is organized into directories for each of the clients. The Windows client comes with one program for the standard, custom, multiple user, and node installs:

`\client\windows\setup.exe`

When you install a client, the installation program copies the client software to the directories listed in Table 6 on page 206.

Table 6. Windows client installation directories

Directory	Purpose
\Program Files\IBM\OnDemand32	Program directory. For standard, custom, and multiple user installs, identifies a hard disk on the workstation. For node installs, identifies a network drive.
\Program Files\IBM\OnDemand32	Local directory. Identifies a hard disk on the workstation.
- On Windows XP: \Users\<i>admin</i>\AppData\Roaming\IBM\OnDemand32\DATA - On Windows 7: %APPDATA%\IBM\OnDemand32\DATA	Temporary data directory of the user's own profiles. Identifies a hard disk on the workstation. Where <i>admin</i> is the name of the administrator of the workstation on Windows XP.
\Program Files\IBM\OnDemand32	AFP font file directory. For standard, custom, multiple user installs, identifies a hard disk on the workstation. For node installs, identifies a network drive.
- On Windows XP: \Users\<i>admin</i>\AppData\Roaming\IBM\OnDemand32\PRINT - On Windows 7: %APPDATA%\IBM\OnDemand32\PRINT	Temporary print directory of the user's own profiles. Identifies a hard disk on the workstation. Where <i>admin</i> is the name of the administrator of the workstation on Windows XP.
- On Windows XP: \Users\<i>admin</i>\AppData\Roaming\IBM\OnDemand32\RES - On Windows 7: %APPDATA%\IBM\OnDemand32\RES	Temporary resource directory of the user's own profiles. Identifies a hard disk on the workstation. Where <i>admin</i> is the name of the administrator of the workstation on Windows XP.

Distribution install

Overview

A distribution install copies the contents of the OnDemand Windows client directory tree from the OnDemand client CD-ROM to a shared location on a network file server. You can then use the copy on the network file server to perform subsequent standard, custom and distribution installs. The distribution install is also useful if you need to add user-defined files to the client installation.

Note: You can also distribute OnDemand client software by sharing the CD-ROM from the CD-ROM drive on a network file server.

Distributing Adobe software

If you need to distribute Adobe Acrobat viewing software to your OnDemand users, you can copy the Adobe software to a shared location on a network file server. Other users can then use the copy to install the Adobe software on their workstations. See the *User's Guide* for information about obtaining and installing Adobe software.

Copying OnDemand software to the server

To copy the OnDemand client software to a network file server, follow these steps:

1. Log on to the server with a userid that has administrator permissions.
2. Insert the OnDemand client CD-ROM in the drive.
3. Copy the entire contents of the \client\windows client directory tree to a drive on the network file server.

Note: Use a copy method that preserves the directory and file structure of the CD-ROM on the network file server.

After you copy the OnDemand client software to the network file server, make the server directories (or folders) available to network users. The procedure differs for each network and operating system. However, you generally need to give users read-only access to directories. If applicable for your network, share the folders by giving the folder location on the network file server a network name (share name).

After you copy the software to the server and share the directories, users can run a standard or custom install from the server to install a client on their workstation.

Distributing user-defined files

An administrator can store user-defined files in the OnDemand client installation directory tree on a distribution server. Any user-defined files stored there get copied to the workstation when a user runs the Setup program from the server. User-defined files can be distributed to Windows clients. To read more about support for user-defined files, see Chapter 17, “Distributing user-defined files,” on page 211.

Configuring the client to connect to a z/OS Version 2.1 server

Important: The OnDemand for OS/390® Version 2.1 server is no longer supported.

The IBM Content Manager OnDemand for z/OS Version 2.1 server must support the OnDemand Windows client level that is installed on your machine. Before you install the client, you must ensure that your server maintenance level supports the client you plan to install. Later releases of the server display supported clients during server initialization.

To enable the client to access an OnDemand for z/OS Version 2.1 server, you need to configure the client and define some settings. These settings can be defined during setup, with the setup.ini file, or later by using a manual process.

To configure the OnDemand client to access a z/OS Version 2.1 server:

1. **Optional but recommended:** Specify settings in the setup.ini file that are necessary for the OnDemand for z/OS Version 2.1 server.
2. Add the OnDemand for z/OS Version 2.1 required command line parameters. You can add the parameters in the setup.ini file, or manually follow installation steps. After you add the parameters, ensure that the shortcut properties target field is updated with these parameters.
3. **Required for Object Linking and Embedding (OLE):** Use the SetFolderCloseMemoryRelease method.
4. **Optional but recommended:** Configure the client to run in short sockets mode for better performance. This configuration can be automated through the setup.ini file, or you can do this process manually.

5. In the Update Servers window, clear the **Attempt Unified Logon** check box, unless you need this option for a specific purpose.

Step 1: Recommended: Specifying settings in the setup.ini file

You use the setup.ini file when you run Setup to install the OnDemand client with certain options. You can add an [S390] section to automatically install the client to connect by using short sockets mode, and add to the OnDemand client shortcut the parameters required by OnDemand for z/OS Version 2.1. For example:

```
[S390]
Sockets=User
Parameters=/Q /O 1 /9
```

The short sockets mode is recommended for better performance, and the command-line parameters are required when you connect to an OnDemand for z/OS Version 2.1 server.

Sockets=User

Places the Disconnect=1 short sockets parameter in the Windows registry HKEY_CURRENT_USER registry key.

Sockets=Local

Places the Disconnect=1 short sockets parameter in the Windows registry HKEY_LOCAL_MACHINE registry key. For OnDemand on Windows systems where multiple users need to log on to the operating system with separate IDs, this option is recommended.

Parameters

In addition to the required command line parameters listed in the above example, you can add other command line parameters to the [S390] section. See “Step 2: Required: Adding the required command line parameters” for more information. For example:

```
Parameters=/Q /O 1 /9 /S servername
```

Step 2: Required: Adding the required command line parameters

You must add several command-line parameters for the OnDemand Windows client to work properly when accessing an OnDemand for z/OS Version 2.1 server. The command-line parameters can be added automatically when you set up the Setup.ini file, or they can be added manually to the OnDemand shortcut. An example of the target for the OnDemand shortcut is:

```
C:\Program Files\IBM\OnDemand32\ARSGUI32.EXE /1 C:\Program Files\IBM\OnDemand32\LOCALE\ENU /Q /O 1 /9
```

where

/Q – Frees memory when a folder is closed

Frees all memory associated with a folder when it is closed, and refreshes the folder information from the server when it is reopened.

/O 1 – Maximum open folders

Limits the number of open folders to one.

/9 – Additional OnDemand for z/OS parameters

If you add this parameter, the Public Note check box is checked and disabled. All notes in OnDemand for z/OS Version 2.1 are public. The **Note can be copied to another server** check box is checked and disabled. This allows notes to be saved with the ad-hoc CD-ROM feature. Also, from

the Document Hitlist, server print is enabled, and local and LAN print are disabled. When you view the document, local, LAN, and server print are all enabled.

Step 3: Required for OLE: Using the SetFolderCloseMemoryRelease method

If you want to implement Object Linking and Embedding (OLE) Controls, use the SetFolderCloseMemoryRelease method when a folder is closed for OnDemand for z/OS Version 2.1. This is the equivalent of the required /Q command line parameter described in "Step 2: Required: Adding the required command line parameters" on page 208

Step 4: Recommended: Configuring the client to run in short sockets mode

When the OnDemand client is installed, it is configured to start the server as a long running task (long sockets). For maximum performance, change the default setting from long running task (long sockets) to short running task (short sockets).

DISCONNECT = 1 is the parameter for short sockets, and DISCONNECT = 0 is the parameter for long sockets (default). You can configure short sockets in one of the following ways:

- Use the Setup.ini file to configure short sockets automatically
- Manually configure short sockets

For more information about how to use the Setup.ini file to configure short sockets automatically, see "Step 1: Recommended: Specifying settings in the setup.ini file".

To manually configure short sockets, you must edit the Windows System Registry (REGEDT32):

1. Ensure that the OnDemand client is installed on your workstation.
2. Select **Start > Run**, and open **regedit**.
3. In regedit, select **HKEY_CURRENT_USER** or **HKEY_LOCAL_MACHINE > Software > IBM > OnDemand32 > Common > Servers**.
4. Click **Servers**. You should see entries for Default and SERVERS.
5. Right-click the right side of the screen.
6. Select **New > String Value**.
7. Enter **Disconnect**.
8. Double-click **Disconnect**. Then, enter 1 for short sockets or 0 for long sockets.
9. Close regedit.

Step 5: Clearing the Attempt Unified Logon check box in the Update Servers window to access a z/OS server

After you start the OnDemand client, in the Update Servers window, ensure that the **Attempt Unified Logon** check box is not selected. A unified logon is valid only when the OnDemand server is in a Windows environment. If you are connecting to OnDemand on z/OS, you should not select **Attempt Unified Logon**.

Chapter 17. Distributing user-defined files

This section provides information about how to configure the OnDemand Windows client installation program to distribute user-defined files to your users. This section is of primary interest to administrators responsible for installing and configuring software products.

Overview

User-defined files can be stored in the OnDemand Windows client directory tree on a distribution server.³ Any files (and subdirectories of files) that you store there are copied along with the standard OnDemand client files when a user installs the client from the server to a workstation. You can distribute the following types of user-defined files:

- Registry files. These files are imported into the Registry on the user's workstation.

Note: It is easy to accidentally destroy important data and render a system completely unusable by importing Registry files. Be careful when using this function.

- Windows files. These files are copied to the base Windows directory tree on the user's workstation.
- Miscellaneous client files. These files are copied to the \Program Files\IBM\OnDemand32 directory tree on the user's workstation. You can use this feature to replace (overwrite) files supplied by IBM. When a user runs the Setup program from the server, it first copies the files supplied by IBM to the workstation. The Setup program then copies any user-defined files to the workstation. If a user-defined file has the same name as a file supplied by IBM, the IBM-supplied file will be overwritten by the user-defined file. For example, suppose you need to modify the AFP character set definition file (CSDEF.FNT) to map fonts your documents were created with to fonts that can be displayed on the workstation. You can automatically distribute an updated version of the file to your users by storing the modified CSDEF.FNT file in the CUSTOM\FILES\FONT directory on the distribution server. When a user installs the client from the server, the Setup program first copies the CSDEF.FNT file supplied by IBM to the workstation. The Setup program then copies the CSDEF.FNT file you modified to the workstation.

The following topics contain more information about distributing user-defined files:

- Copy OnDemand client software to a distribution server
- Add subdirectories to hold user-defined files
- Store user-defined files in subdirectories
- Install the OnDemand client on a user's workstation

3. A distribution server is a network file server that contains a copy of the OnDemand client software in a shared location. Other users on the network use the copy to run standard, custom and distribution installs. See Chapter 16, "Installing client software on a network," on page 205 for more information about setting up a distribution server to share OnDemand software on a network.

Copying OnDemand client software to the server

See Chapter 16, “Installing client software on a network,” on page 205 for instructions about copying the OnDemand client software to a distribution server.

Adding subdirectories

All user-defined files must be stored in the CUSTOM subdirectory tree under the OnDemand Windows client directory tree on the distribution server. By default, the Windows client directory tree is ARS32. (If you followed the instructions in Chapter 16, “Installing client software on a network,” on page 205, then the OnDemand Windows client directory tree on the distribution server is \client\windows\ars32.)

To configure the server to install user-defined files:

1. Create a CUSTOM directory under the Windows 32-bit client directory. For example:

```
mkdir \client\windows\ars32\custom
```
2. Add one or more of the following subdirectories to the CUSTOM directory. The subdirectories that you add will depend on the type of user-defined files you want to distribute to your users. For example:

\client\windows\ars32\custom\registry

To hold Registry files (file type REG). These files will be imported into Registry on the user's workstation. Registry files typically comprise a selected branch of the Registry exported using a Registry editor.

Note: It is easy to accidentally destroy important data and render a system completely unusable by importing Registry files. Be careful when using this function.

\client\windows\ars32\custom\windows

To hold Windows files. The Setup program copies these files and subdirectories to the base Windows directory on the user's workstation. (The Setup program automatically determines the name of the base Windows directory on a user's workstation.) If necessary, add subdirectories. For example, suppose you plan to distribute Windows 32-bit system files. You would add a SYSTEM32 subdirectory (\CLIENT\WINDOWS\ARS32\CUSTOM\WINDOWS\SYSTEM32). The Setup program copies files in this directory to the \WINDOWS\SYSTEM32 directory on the user's workstation.

\client\windows\ars32\custom\files

To hold miscellaneous OnDemand client files. The Setup program copies these files and subdirectories to the \Program Files\IBM\OnDemand32 directory tree on the user's workstation. If necessary, add subdirectories. For example, you plan to distribute AFP font files. Add a FONT subdirectory (\CLIENT\WINDOWS\ARS32\CUSTOM\FILES\FONT). The Setup program copies files in this directory to the \Program Files\IBM\OnDemand32\FONT directory on the user's workstation.

Storing user-defined files on the server

After you copy the OnDemand client software to the distribution server and create the \custom directories, you can store files in individual subdirectories. For example, store DLL files that you want OnDemand to run in the \CLIENT\WINDOWS\ARS32\CUSTOM\FILES directory.

Installing the OnDemand client

After you set up the CUSTOM directory tree on the distribution server, users can begin installing the client and the user-defined files. The next time that a user runs the Setup program from the server, the Setup program installs the OnDemand client on the workstation and copies all of the user-defined files that you stored on the server to the user's workstation.

When a user selects the Typical option during install, the Setup program automatically copies the user-defined files to the workstation.

When a user selects the Custom option during install, the Setup program can copy the user-defined files to the workstation. If the user chooses to install one of the clients, then the Setup program copies the user-defined files to the workstation. Otherwise, the Setup program does not copy the user-defined files to the workstation. For example, if the user selects the Custom option and chooses to install only the Sonoran Fonts, then the Setup program does not copy the user-defined files to the workstation. However, if the user selects the Custom option and chooses to install the administrative client, then the Setup program copies the user-defined files to the workstation.

Chapter 18. Using response files

This section describes how to use response files to automate the installation of the client software. This section is of primary interest to administrators responsible for installing and configuring software products.

Introduction

This section provides information that can be used to create and use response files to install the Windows client software on workstations connected to the network. You typically use the Setup program to create a response file. You then install the software on other workstations by running the Setup program and specifying the name of the response file.

A response file is an ASCII file that supplies the client-specific configuration information required during redirected installation of a product on a workstation. The response file contains predefined answers to the configuration questions that users are normally asked during a product installation, such as the installation drive and directory and the components to install. A system administrator can use a response file to automate the installation and configuration of the Windows 32-bit client software over a network of workstations. The response file makes it unnecessary for the system administrator (or other user) to sit at each workstation and manually enter an answer to each question that is displayed during installation.

Format of a response file

The format of a response file is similar to that of an .INI file. A response file contains pairs of keywords and values organized into sections. The keywords and values are interpreted during software installation.

Creating a response file

Response files commonly have an extension of .ISS and are found in the Windows directory.

You can create a response file by running the Setup program with the **-r** command line option. For example, the command:

```
\client\windows\setup -r
```

Causes the Setup program to record all of your answers to the product installation questions in the SETUP.ISS response file. You can direct the Setup program to place the response file in a different directory and name a response file by specifying the **-f1** command line option. For example, the command:

```
\client\windows\setup -r -f1n:\client\windows\ars32in.iss
```

Causes the Setup program to create the ARS32IN.ISS file in the \CLIENT\WINDOWS directory on the N drive.

Installing software using a response file

A response file is not invoked directly. Instead, a response file is specified as a parameter value for the installation program. You can run the Setup program and specify a response file with the **-s** command line option. For example, the command:

```
\client\windows\setup -s
```

Causes the Setup program to install the software using the instructions found in the SETUP.ISS response file in the \CLIENT\WINDOWS directory on the current drive. By default, the response file must be located in the directory from which you run the Setup program. Use the **-f1** option to identify the location and name of the response file. For example, the command:

```
\client\windows\setup -s -f1n:\client\windows\ars32in.iss
```

Causes the Setup program to install the software using the ARS32IN.ISS response file located in the \CLIENT\WINDOWS directory on the N drive.

The response file directs the processing of the installation for the Windows 32-bit client software. When you run the Setup program with the **-s** option, no messages or dialog boxes are displayed. Instead, messages are written to a log file. By default, the log file (SETUP.LOG) is written to the directory from that contains the Setup program. You can direct the Setup program to place the log file in a different directory and name the log file by specifying the **-f2** command line option. For example, the command:

```
\client\windows\setup -s -f1n:\client\windows\ars32in.iss -f2c:\temp  
\ars32in.log
```

Causes the Setup program to write the log file ARS32IN.LOG in the TEMP directory on the C drive.

Verifying software installation

To verify the installation of a product that you installed using a response file, open the log file and locate the ResponseResult section. Examine the value of the ResultCode parameter. The return code should be zero (0).

Using a response file to install OnDemand software

In general, you would complete the following steps to prepare the OnDemand Windows client software for installation using a response file and then install the software on other workstations connected to the network.

1. Install the software on a workstation. Run the Setup program with the **-r** option to create the response file and the **-f1** option to identify the location and name of the response file. When prompted by the Setup program, select the Typical option.
2. Test the installation process and the response file by installing the software on a user's workstation. Specify the name of the response file you created in step .
3. After testing and validating the response file, install the software on other workstations. Run the Setup program with the **-s** option to read the response file you created in step , the **-f1** option to identify the response file, and the **-f2** option to identify the directory where the Setup program writes the log file.
4. Examine the log files to verify the installation of the software.

Chapter 19. Full report browse and reprint

After a you log on to a server, open a folder, and generate a hit list, the **View Full Report** button displays. This additional button is displayed if the server is at version 7.1.2.3 or later and the Content Manager for OnDemand Administration Client has been used to specify that the full report browse feature is permitted for the specific folder.

Selecting **View Full Report** causes the entire document load associated with a hit to be viewed as a single document. The hits that constitute this composite document are:

- The load ID is extracted from the document handle associated with a selected hit.
- The file containing the complete set of hits for that load ID is retrieved from the server.
- The hits are extracted from that file and a hitlist created in the original load sequence.
- A document query is done against the server asking for all hits associated with the load ID. This creates a second hit list, in unpredictable sequence, of all those hits accessible to the current user.
- Any hit in the first hit list that does not also appear in the second hit list is removed. The hit list that results consists of all hits from the load that are accessible to the current user and is in the original load sequence.

The document that initially displays is the first piece of the load. This piece is either the entire first hit for a non-large object document or the first segment of the first hit for a large object document. When first displayed, the number of pages in the composite document (that is, the entire load) is not known.

Differences from normal document viewing are:

- The status bar displays "Page 1 of ?".
- The vertical scroll bar tab cannot be dragged to change to a different page.
- In the GoTo dialog box, the maximum page number is shown as "?"
- The Thumbnails Bar does not display.

The following menu items or toolbar icons are disabled:

- **Send**
- All items under the Notes menu
- **View Previous Hit in Document List**
- **View Next Hit in Document List**
- **New Window**

Operations on the composite document proceed in a manner similar to large objects. If an action, such as a page change or a find string, requires the retrieval of a new piece, the user is prompted to confirm that retrieval before proceeding, unless the **Auto Continue Through All Segments** option is selected. If an action causes an attempt to retrieve a page which is greater than any page in the load, the action terminates when the last page of the load has been reached. As soon as an

| action results in the retrieval of the final piece of the document, the status bar,
| vertical scroll bar, and GoTo dialog box revert to normal display and behavior.

| If a full report document is being viewed and the entire document is printed to a
| server printer, the entire report will be printed as a single job.

| The full report browse feature can be used for any viewer type other than User
| Defined.

Chapter 20. ICU converters

You can create and maintain custom ICU converters to use with the OnDemand Client. The codepage or character set conversion engine in the OnDemand Client was upgraded from ICONV to ICU 3.0 in version 7.1.2.1. The AFP Workbench Viewer that the OnDemand Client invokes to render and display AFP documents was also subsequently updated to ICU 3.0 in V7.1.2.5. The OnDemand Client V8.4.0.0 and later uses ICU 3.6.

Note: For the 7.1.2 level of the OnDemand Client, only versions 7.1.2.0 through 7.1.2.9 were released. There are no versions beyond 7.1.2.9.

To create custom ICU 3.6 converters:

1. To create a new custom code page mapping, please follow these steps to create a new ICU converter: Download the Windows Binary ICU 3.6 package from the ICU download page:<http://www.icu-project.org/download/3.6.html>
2. Add the bin directory from the extracted ICU Binary package in your PATH environment variable.
3. Create the custom ICU 3.6 data directory. From the OnDemand Client locale directory, for example, C:\Program Files\IBM\OnDemand32\Locale, create a subdirectory with the same name as the ICU 3.6 data file (for example, icudt36l).
4. Download the latest character mapping table (ucm) files from ICU from: <http://source.icu-project.org/repos/icu/data/trunk/charset/data/ucm/>
5. If you need to modify an existing ICU 3.6 converter that was shipped with the OnDemand Client V8.4.0.0 or later, and it is not available from ICU, download the `afpucm36.zip` from <ftp://service.software.ibm.com/software/ondemand/utills/cp2ucm/> or <ftp://service.software.ibm.com/software/ondemand/utills/cp2ucm/>. This compressed archive file contains all the character mapping tables used by the AFP Viewer for display of AFP Printer fonts that are not available from ICU.
6. Select an existing ucm file that closely matches your desired new converter. This ucm can be used as the starting point to change existing code point values. When modifying an existing ucm, we recommend removing the year specific version number from the filename. For example, if the ucm filename is `ibm-2065_p100-2005.ucm`, it should be renamed to `ibm-2065.ucm`. Ensure to also change the `<code_set_name>` value inside the ucm file for consistency.
7. If you are creating a new ICU converter, be sure to use one of CDRA defined private use CCSIDs in the range of 0xE000 to 0xEFFF in hexadecimal, or 57344 to 61439 in decimal.
8. Run the `makeconv` ICU command to create the `.cnv` file. For example, if the new converter name is `ibm-57344`, the syntax would be `makeconv ibm-57344.ucm`
9. Add the `ibm-57344.cnv` file in the ICU data directory that you created in step 3.
10. Test the code page mappings by restarting the OnDemand Client and rendering the report.

If you have custom ICU converters created for OnDemand Client 7.1.2.5 through 7.1.2.9 and want to migrate them to OnDemand Client 8.4.0.0 and later, perform the following steps:

1. If a modified converter has the year specific version number, like `ibm-380_p100-2006.cnv`, remove the year specific version number for ICU to recognize this converter. In this case, rename the custom converter to `ibm-380.cnv`.
2. The format of the `.cnv` files can change between ICU versions. We recommend recompiling your `ucm` files using the `makeconv` utility from the Windows Binary ICU 3.6 package (<http://www.icu-project.org/download/3.6.html>). If a converter has been renamed in step 1, be sure to change the `<code_set_name>` value inside the `ucm` file before running `makeconv`.

Chapter 21. Troubleshooting

StoreDoc() API returns error code 2

Symptom

Your StoreDoc() API returns an error code of 2.

Possible causes

For the StoreDoc() API to successfully load a document, two attributes that must be set in the following way:

- The Application Group's database organization must have the attribute Multiple Loads per Table in order for StoreDoc() to succeed. This attribute is set in the General/Advanced tab in the Properties window of the application group.
- The Application Group's Expiration Type in Storage Management must be set to "Expiration Type: Segment" in order for StoreDoc() to succeed. This attribute is set in the Storage Management tab in the Properties window of the application group.

Action

Make sure that both attributes of the application group is correctly set.

Hints and tips

How to allow a user to print AFP files without opening them with the workbench or navigation plug-in

To allow a user to print AFP files without opening them with the workbench or navigation plug-in, you cannot use npoafp32.dll, instead, you should use the Content Manager OnDemand OLE control.

Appendix A. Microsoft Visual Basic DDE sample program

This program sample is provided on an as-is basis. A licensee of the OnDemand product is free to copy, revise, modify, and make derivative works of this program sample as they see fit.

Global variables used by the sample program

This section defines the global variables used by the sample program.

```
Option Explicit
Global Const apptitle = "OnDemand VB Demo"
Global Const yes = 1
Global Const no = 0
Global Const leave = 100000#

Global Const arstopic = "ARS|ARS"      'Default DDE application at topic

Global Const defini = "arsvblan.ini"   'Default ini file name
Global Const defstanza = "VBDEMO"     'Default stanza
Global Const arsgui = "ARSGUI32.EXE"  'Default Client exe

'Default Client parms
'D Default Client startup parms
' /I  = enable DDE interface
' /B  = disable user confirmation
' /W N = invisible window (don't use during debugging)
' /K  = disable Exit (don't use during debugging)
' /V  = disable anticipation
Global Const arsguiopts = " /I /B /V /W N /K"

Global ininame As String               'Ini file name
Global server As String                'Server name
Global userid As String                'userid
Global pass As String                  'password
Global folder As String                'folder

Global guipath As String               'Client exe path
Global Const linktype = 2              'DDE linkage = manual
Global Const linktime = 3000           'DDE wait time

Global doc_ids(0 To 2) As String       'Doc ids returned on OPEN_DOC

Global txtStack(1 To 10) As String

'Define the Windows APIs used by the program
Declare Function GetModuleHandle Lib "Kernel" (ByVal lpModuleName As String) As Integer
Declare Function GetPrivateProfileString Lib "Kernel" (ByVal sname$, ByVal Kname$,
    ByVal Def$, ByVal ret$,ByVal Size%, ByVal FName$) As Integer
Declare Function SetFocusAPI Lib "User" Alias "SetFocus" (ByVal hWnd As Integer) As Integer
```

Entry point for the sample program

The sample program will drive the OnDemand Windows client by using the DDE interface. Some important things to note:

- The commands sent to the OnDemand client are not DDE EXECUTE's, they are all DDE REQUEST's. This is important because sending DDE EXECUTE's to the OnDemand client will result in an error.

- It is important to start the OnDemand client with at least the /I option. This enables the DDE interface of the OnDemand client. (See the globals section to see what options were used for the VBDEMO.)

The sample program was written using Visual Basic 5.0 (32-bit). The txtDemo control (which is hidden) is used as the DDE client in the DDE conversation with the OnDemand client window. In fncDDElink(), you will see where we setup this control to perform the DDE request and that the data comes back to this control. Therefore any data returned by the OnDemand client window will have to be parsed from out of this control.

```
Sub Main()
    Dim rc As Integer

    'Initialize globals and read in data from ini file(s).
    Call fncInit

    frmStatusDlg.lblStatus.Caption = "Starting client..."
    frmStatusDlg.Show 0

    'Start OnDemand client.
    Shell (guipath + arsgui + arsguiopts)

    'Logon to the server (logon information was gathered from ini
    'file during fncInit. User cannot do anything else while this
    'is going on. Try and use SetFocusAPI() to restore focus to our
    'status message while this is going on.
    frmStatusDlg.lblStatus.Caption = "Logging on to Server..."
    Call frmCreditV1.fncLogon
    rc = SetFocusAPI(frmStatusDlg.hWnd)

    'Open the "Baxter Bay Credit" folder
    frmStatusDlg.lblStatus.Caption = "Opening folder..."
    Call frmCreditV1.fncOpenFolder

    'Don't need the status message box any more.
    frmStatusDlg.Hide

    'Only after we have logged on and opened the folder do we
    'display the VBDEMO form.
    frmCreditV1.Show 1
End Sub

'Send DDE REQUEST of CLOSE_ALL_DOCS to the client window:

Private Sub fncCloseDoc()
    Dim cmdline, qrc As String

    Call fncDispStatus("Close all open docs...")
    cmdline = "CLOSE_ALL_DOCS"
    Call fncDDElink(arstopic, cmdline, linktype, 3000)
    qrc = fncGetrc(txtDemo)
    If qrc <> "0" Then
        Call quit(cmdline, qrc)
    End If

    Call fncDispStatus("All open docs closed.")
End Sub

'This procedure handles the link to the OnDemand client.
'Topic should come is as ARS|ARS, this is the app name and topic name.

Private Sub fncDDElink(ByVal topic As String, ByVal cmnd As String,
    ByVal mode As Integer, ByVal waittime As Integer)
    'Setup local variables
```

```

    Dim sync, lntxtDemo, i, rc As Integer
    Dim workchar, workline, msg As String
'Set up error handler to show contact errors
    On Error GoTo HandleError
'Set up DDE link and pass required data:
    txtDemo = "-"
    txtDemo.LinkTimeout = waittime
    txtDemo.LinkTopic = topic
    txtDemo.LinkItem = cmdnd
    txtDemo.LinkMode = mode
    'Calling LinkRequest performs the request.
    txtDemo.LinkRequest
Exit Sub

HandleError:
'Handle DDE errors
    rc = Err
    Select Case rc
        Case 280 To 297
            Select Case rc
                Case 280
                    msg = "DDE channel not closed; awaiting response from foreign application"
                Case 281
                    msg = "No more DDE channels"
                Case 282
                    msg = "DDE requests are being refused"
                Case 283
                    msg = "Too many apps responded"
                Case 284
                    msg = "DDE channel locked"
                Case 285
                    msg = "App is not accepting DDE requests..."
            End Select
        Case Else
            msg = "Non-DDE error occurred " + Str(rc)
    End Select
    MsgBox msg
    Resume Next
End Sub

'Used to send DDE REQUEST command of ACTIVATE_DOC or OPEN_DOC to
' the OnDemand client.

Private Sub fncDispDoc(ByVal docnum As Integer)
    Dim cmdline, qrc As String

    'If the document the user is requesting to be displayed has
    'previously been opened, then use ACTIVATE_DOC to redisplay
    'it, otherwise we will need to OPEN_DOC and store away the
    'document id.
    'If the user closes the document view from the client interface
    'we need to be notified of this event so that we can update
    'our doc_id array. Currently the client does not support this.

    If doc_ids(docnum) <> "0" Then
        Call fncDispStatus("Activating the document...")
        cmdline = "ACTIVATE_DOC /D " + doc_ids(docnum)
        Call fncDDELink(arstopic, cmdline, linktype, 3000)
        qrc = fncGetrc(txtDemo)
        If qrc <> "0" Then
            'The user possibly closed the view from the client,
            'reset the document id to 0 and tell the user to try again.
            doc_ids(docnum) = "0"
            Call fncDispStatus("Activating the document...ERROR")
            MsgBox "Could not activate, try to view again!"
            Exit Sub
        End If
    End If

```

```

        Call fncDispStatus("Activating the document...done")
Else
    'Open the document
    Call fncDispStatus("Open the document...")
    cmdline = "OPEN_DOC /N " + Str(docnum)
    Call fncDDElink(arstopic, cmdline, linktype, 3000)
    qrc = fncGetrc(txtDemo)
    If qrc <> "0" Then
        Call quit(cmdline, qrc)
    End If
    doc_ids(docnum) = fncGetdochandle(txtDemo)
    Call fncDispStatus("Open the document...done.")
End If

'Make the display visible
Call fncDispStatus("Opening the display...")
cmdline = "SHOW_WINDOW /W"
Call fncDDElink(arstopic, cmdline, linktype, 3000)
qrc = fncGetrc(txtDemo)
If qrc <> "0" Then
    Call quit(cmdline, qrc)
End If
Call fncDispStatus("Opening the display...done.")

Call fncDispStatus("Document retrieval complete.")
End Sub

'Obtains the hitlist of documents from the OnDemand client.
Private Sub fncGetHitlist()
    Dim cmdline, qrc As String
    Dim num_docs As Integer

    'Get the number of documents which matched the search
    'criteria.
    cmdline = "GET_NUM_DOCS_IN_LIST"
    Call fncDDElink(arstopic, cmdline, linktype, 3000)
    num_docs = CInt(Mid(txtDemo, 3, 10))
    If num_docs = 0 Then
        MsgBox "No documents found matching search criteria!"
        Exit Sub
    End If

    Call fncDispStatus("Getting account information...")

    'Get the first document and parse its data to display
    cmdline = "GET_DOC_VALUES /N 0"
    Call fncDDElink(arstopic, cmdline, linktype, 3000)
    Call fncExtract(txtDemo.Text)
    'Display its data
    pnlPayData1.Caption = txtStack(1)
    Panel3D1.Caption = txtStack(4)
    'Add about 20 days or so to the statement date'
    Panel3D2.Caption = fncParseDate(txtStack(1))
    cmdViewStmt1.Enabled = True

    'If there are at lease two documents then get number 2
    If num_docs > 1 Then
        cmdline = "GET_DOC_VALUES /N 1"
        Call fncDDElink(arstopic, cmdline, linktype, 3000)
        Call fncExtract(txtDemo.Text)
        'Display its data
        pnlPayData2.Caption = txtStack(1)
        Panel3D3.Caption = txtStack(4)
        'Add about 20 days or so to the statement date'
        Panel3D4.Caption = fncParseDate(txtStack(1))
        cmdViewStmt2.Enabled = True
    End If

```

```

        'There was only 1 document so disable 2nd "View" button.
        cmdViewStmt2.Enabled = False
    End If

'If there are at lease three documents then get number 3
    If num_docs > 2 Then
        cmdline = "GET_DOC_VALUES /N 2"
        Call fncDDElink(arstopic, cmdline, linktype, 3000)
        Call fncExtract(txtDemo.Text)
        'Display its data
        pnlPayData3.Caption = txtStack(1)
        Panel3D5.Caption = txtStack(4)
        'Add about 20 days or so to the statement date'
        Panel3D6.Caption = fncParseDate(txtStack(1))
        cmdViewStmt3.Enabled = True

    Else
        'There were only 2 documents so disable 3rd "View" button.
        cmdViewStmt3.Enabled = False
    End If
    Call fncDispStatus("Getting account information...done.")
End Sub

'Procedure used to hide the OnDemand client window.
'Sends a DDE REQUEST message of SHOW_WINDOW to the client.
Private Sub fncHideWindow()
    Dim cmdline, qrc As String

    cmdline = "SHOW_WINDOW /W N"
    Call fncDDElink(arstopic, cmdline, linktype, 3000)
    qrc = fncGetrc(txtDemo)
    If qrc <> "0" Then
        Call quit(cmdline, qrc)
    End If
End Sub

'Logon to the OnDemand client.
Public Sub fncLogon()
    Dim cmdline, qrc As String

    Call fncDispStatus("Logon to Client...")
    cmdline = "LOGON /S " + server + " /U " + userid + " /P " + pass
    Call fncDDElink(arstopic, cmdline, linktype, 3000)
    qrc = fncGetrc(txtDemo)
    If qrc <> "0" Then
        'If we fail the logon the client will display his logon dialog.
        'We will not return from this DDE call until the user either
        'successfully log's onto a server or cancel's the process, in
        'which case we end up with an error code and inside of this If
        'statement. Close the client and then ourselves.
        'I am not sure if the above statement is valid if you started up
        ' the OnDemand client with the Disable anticipation (/V) parameter.
        Call fncDDElink(arstopic, "EXIT", linktype, 3000)
        Call fncDispStatus("Logon to client...failed.")
    End If
    End If
    Call fncDispStatus("Logon to Client...done.")
End Sub

'Open up an OnDemand folder.
Public Sub fncOpenFolder()
    Dim cmdline, qrc As String

    Call fncDispStatus("Open the folder...")
    cmdline = "OPEN_FOLDER /F " + folder
    Call fncDDElink(arstopic, cmdline, linktype, 3000)
    qrc = fncGetrc(txtDemo)

```

```

    If qrc <> "0" Then Call quit(cmdline, qrc)
    Call fncDispStatus("Open the folder...done.")

End Sub

'Search the OnDemand folder for documents.
Private Sub fncSearchDoc(ByVal AcctNum As String)
    Dim cmdline, qrc As String

    'Setup our search fields with the client.
    Call fncDispStatus("Setting up Search...")
    cmdline = "SET_FIELD_DATA /F Account /1 " + AcctNum
    Call fncDDElink(arstopic, cmdline, linktype, 3000)
    qrc = fncGetrc(txtDemo)
    If qrc <> "0" Then
        Call quit(cmdline, qrc)
    End If
    Call fncDispStatus("Setting up Search...done.")

    'Have the client perform the search.
    Call fncDispStatus("Performing the Search...")
    cmdline = "SEARCH_FOLDER"
    Call fncDDElink(arstopic, cmdline, linktype, 3000)
    qrc = fncGetrc(txtDemo)
    If qrc <> "0" Then
        Call quit(cmdline, qrc)
    End If
    Call fncDispStatus("Performing the Search...done.")
End Sub

'Performs three DDE steps for us:
' - Inform client to retrieve selected document.
' - Enable the switch back toolbar button on the clients toolbar so that the
'   user can get back easily to the VBDEMO.
' - Switch focus to the client.
Private Sub fncViewDoc(ByVal docnum As Integer)
    'Setup local variables
    Dim MyHandle As Integer

    'Display the document
    Call fncDispDoc(docnum)
    'Activate DDE and transfer Focus to OnDemand
    MyHandle = frmCreditV1.hWnd
    Call fncDDElink(arstopic, "ENABLE_SWITCH /H " + Str(MyHandle) + " /C " + apptitle,
    linktype, 3000)
    Call fncDDElink(arstopic, "SET_FOCUS", linktype, 3000)
End Sub

'Displays error code.
Private Sub quit(ByVal qinfo As String, ByVal qrc As String)
    Dim quitstring As String

    quitstring = "Error encountered: " + qinfo + " rc=" + qrc
    MsgBox quitstring
End Sub

End Sub

'GUI control used to display customer information.
'We do not obtain the customer information from out of OnDemand, it is
' not stored there. The normal way to obtain this information would be
' to get it out of your business database. After which you would look up
' the customer statements in OnDemand. We simply get this information from
' out of an ini file.
Private Sub cmdCustInfo_Click()
    Dim acct_num, ini_str As String
    Dim cmdline, qrc As String
    Dim rc As Integer

```

```

Dim first_num, second_num, third_num As Integer

'Zero out the Payment record fields before retrieving new customer
pnlPayData1.Caption = ""
Panel3D1.Caption = ""
Panel3D2.Caption = ""
pnlPayData2.Caption = ""
Panel3D3.Caption = ""
Panel3D4.Caption = ""
pnlPayData3.Caption = ""
Panel3D5.Caption = ""
Panel3D6.Caption = ""
'Zero out the Customer Information fields.
pnlNameData.Caption = ""
pnlSSNData.Caption = ""
pnlDOBData.Caption = ""
pnlMNameData.Caption = ""
pnlAddrData1.Caption = ""
pnlAddrData2.Caption = ""
pnlPhoneData.Caption = ""

'Disable "View" buttons
cmdViewStmt1.Enabled = False
cmdViewStmt2.Enabled = False
cmdViewStmt3.Enabled = False

'Hide client window
Call fncHideWindow

'Look up the account number, contained in the pnlAcctnumData text field
'in the arsvblan.ini file. If found, read the respective
'fields. If not found display error message.
acct_num = txtAcctnumData.Text

'Do at least a little validation.
If Len(acct_num) <> 11 Then
    MsgBox "Correct format for account # is 000-000-000"
    Exit Sub
End If

'If we have gotten to here we know that we have an account
'number of the format 000-000-000. If either of the first
'two sections of the number are non-zero or if the third
'section is not between 001-046 then default to the account
'number 000-000-001.
first_num = Int(Mid(acct_num, 1, 3))
second_num = Int(Mid(acct_num, 5, 3))
third_num = Int(Mid(acct_num, 9, 3))
If first_num <> 0 Or second_num <> 0 Or third_num > 46 Then
    acct_num = "000-000-001"
ElseIf third_num = 0 Then
    MsgBox "Invalid account number!"
    Exit Sub
End If

ini_str = fncParmGet(acct_num, "Name", ininame)
If Len(ini_str) = 0 Then
    MsgBox "'Name' field not found for acct#" + acct_num + "in " + ininame
    Exit Sub
End If
pnlNameData.Caption = " " + ini_str

ini_str = fncParmGet(acct_num, "SSN", ininame)
If Len(ini_str) = 0 Then
    MsgBox "'SSN' field not found for acct#" + acct_num + "in " + ininame
    Exit Sub
End If

```

```

pnlSSNData.Caption = " " + ini_str

ini_str = fncParmGet(acct_num, "DOB", ininame)
If Len(ini_str) = 0 Then
    MsgBox "'DOB' field not found for acct#" + acct_num + "in " + ininame
    Exit Sub
End If
pnlDOBData.Caption = " " + ini_str

ini_str = fncParmGet(acct_num, "MaidenName", ininame)
If Len(ini_str) = 0 Then
    MsgBox "'MaidenName' field not found for acct#" + acct_num + "in " + ininame
    Exit Sub
End If
pnlMNameData.Caption = " " + ini_str

ini_str = fncParmGet(acct_num, "StreetAddress", ininame)
If Len(ini_str) = 0 Then
    MsgBox "'StreetAddress' field not found for acct#" + acct_num + "in " + ininame
    Exit Sub
End If
pnlAddrData1.Caption = " " + ini_str

ini_str = fncParmGet(acct_num, "CityStateZip", ininame)
If Len(ini_str) = 0 Then
    MsgBox "'CityStateZip' field not found for acct#" + acct_num + "in " + ininame
    Exit Sub
End If
pnlAddrData2.Caption = " " + ini_str

ini_str = fncParmGet(acct_num, "PhoneNum", ininame)
If Len(ini_str) = 0 Then
    MsgBox "'PhoneNum' field not found for acct#" + acct_num + "in " + ininame
    Exit Sub
End If
pnlPhoneData.Caption = " " + ini_str

'We are changing customer accounts so before we get new customer
'information, close old customers open documents.
If doc_ids(0) <> "0" Or doc_ids(1) <> "0" Or doc_ids(2) <> "0" Then
    doc_ids(0) = "0"
    doc_ids(1) = "0"
    doc_ids(2) = "0"
    cmdline = "CLOSE_ALL_DOCS"
    Call fncDDElink(arstopic, cmdline, linktype, 3000)
    qrc = fncGetrc(txtDemo)
    If qrc <> "0" Then
        Call quit(cmdline, qrc)
    End If
End If

'Set up the search fields and perform search.
Call fncSearchDoc(acct_num)

'Get the 3 most recent statements.
Call fncGetHitlist

'Give ourselves back the focus
rc = SetFocusAPI(frmCreditV1.hWnd)
End Sub

'User has chosen to exit the VBDEMO. Before exiting close down the client.
Private Sub cmdExit_Click()
    Dim OnDemandHandle As Integer

    Call fncDispStatus("Program ending...")

```

```

'Determine if OnDemand is loaded
OnDemandHandle = GetModuleHandle(arsgui)
'If not loaded, then quit, else shutdown
If OnDemandHandle > 0 Then
    Call fncDispStatus("Shutting Client Down...")
    Call fncDDElink(arstopic, "EXIT", linktype, linktime)
End If

'Terminate the VBDEMO
End
End Sub

'View button number 1. Have the client retrieve the first document in
' the hitlist and display it.
Private Sub cmdViewStmt1_Click()

    Call fncViewDoc(0)

End Sub

'View button number 2. Have the client retrieve the second document in
' the hitlist and display it.
Private Sub cmdViewStmt2_Click()

    Call fncViewDoc(1)

End Sub

'View button number 3. Have the client retrieve the third document in
' the hitlist and display it.
Private Sub cmdViewStmt3_Click()

    Call fncViewDoc(2)

End Sub

'If the user is not using the Exit button to close down the demo
' this function will be called as a result of the form being unloaded
' so go ahead and shut down the client then exit.
Private Sub Form_Unload(Cancel As Integer)
    Dim OnDemandHandle As Integer

    Call fncDispStatus("Program ending...")

    'Determine if OnDemand is loaded
    OnDemandHandle = GetModuleHandle(arsgui)
    'If not loaded, then quit, else shutdown
    If OnDemandHandle > 0 Then
        Call fncDispStatus("Shutting Client Down...")
        Call fncDDElink(arstopic, "EXIT", linktype, linktime)
    End If

    'Terminate the VBDEMO
    End
End Sub

'Make sure that the data they are entering for the account number
' is valid.
Private Sub txtAcctnumData_KeyPress(KeyAscii As Integer)

    Dim pos As Integer

    pos = txtAcctnumData.SelStart

    Select Case KeyAscii
        Case 48 To 59

```

```

        'pos must be 0-2, 4-6, 8-10
    Select Case pos
        Case 0 To 2, 4 To 6, 8 To 10
            'OK
        Case Else
            Beep
            KeyAscii = 0
    End Select
Case 45
    ' the - character
    'pos must be 3 or 7
    If pos <> 3 And pos <> 7 Then
        Beep
        KeyAscii = 0
    End If
Case 8, 127
    'Just let these through.
Case Else
    Beep
    KeyAscii = 0
End Select

End Sub

'This procedure fills in the status line on the form and left adjusts.
Public Sub fncDispStatus(ByVal status As String)
    frmCreditV1.pnlStatus.Caption = status + Space$(255)
End Sub

'This procedure breaks out the words of the input
'string. Words must be delimited by a tab character.
'The words are stored in the global string array txtStack.
Sub fncExtract(ByVal workstring As String)
    Dim txtptr, lenstring, i As Integer
    Dim tabchar, workline, workchar As String

    txtptr = 0
    tabchar = Chr(9)
    workline = ""
    lenstring = Len(workstring)
    workstring = Mid$(workstring, 3, lenstring)

    'Extract chars to the first blank
    For i = 1 To lenstring
        workchar = Mid$(workstring, i, 1)
        'When a tab is found, store result, reset
        If workchar = tabchar Then
            txtptr = txtptr + 1
            txtStack(txtptr) = workline
            workline = ""
        'Otherwise, keep building the work string
        Else
            workline = workline + workchar
        End If
    Next

    If Len(workline) > 0 Then
        txtptr = txtptr + 1
        txtStack(txtptr) = workline
    End If
End Sub

'This function extracts out the document handle from the
'return string.
Function fncGetdochandle(ByVal workstring As String)
    Dim lenstring, first, i As Integer
    Dim rc, workline, workchar As String

```

```

'Set the return code for invalid function call
rc = "999"
first = yes
workstring = Trim$(workstring)
lenstring = Len(workstring)

'Extract chars to the first blank
If lenstring > 0 Then
    workline = ""
    For i = 1 To lenstring
        workchar = Mid$(workstring, i, 1)
        'When a second blank is found, stop
        If workchar = " " Then
            If first = yes Then
                first = no
                workline = ""
            Else
                rc = workline
                i = leave
            End If
        'Otherwise build up return code
        Else
            workline = workline + workchar
        End If
    Next
    'If the doc handle has been built, assign it
    If workline <> "" Then
        rc = workline
    End If
End If

'Set the function return value
fncGetdochandle = rc

```

End Function

```

'This function extracts out the return code from the
'return string.
Function fncGetrc(ByVal workstring As String)
    Dim lenstring, i As Long
    Dim rc, workline, workchar As String

    'Set the return code for invalid function call
    rc = "999"
    workstring = Trim$(workstring)
    lenstring = Len(workstring)

    'Extract chars to the first blank
    If lenstring > 0 Then
        workline = ""
        For i = 1 To lenstring
            workchar = Mid$(workstring, i, 1)
            'When a blank is found, stop
            If workchar = " " Then
                rc = workline
                i = leave
            'Otherwise build up return code
            Else
                workline = workline + workchar
            End If
        Next
        'If a return code has been built, assign it
        If workline <> "" Then
            rc = workline
        End If
    End If
End Function

```

```

    'Set the function return value
    fncGetrc = rc

End Function

'Perform global initialization
Sub fncInit()
    Dim ini_str As String

    'Set document ids for all three to 0
    doc_ids(0) = "0"
    doc_ids(1) = "0"
    doc_ids(2) = "0"

    'Disable "View" buttons
    frmCreditV1.cmdViewStmt1.Enabled = False
    frmCreditV1.cmdViewStmt2.Enabled = False
    frmCreditV1.cmdViewStmt3.Enabled = False

    'The VBDEMO keyword in the PATHS stanza of the ars.ini file
    'points to the .ini file where the other
    'demo settings can be picked up. If the
    'VBDEMO keyword cannot be found, the ini
    'file is set to arsvblan.ini.

    'Try to find vbdemo inifile name
    ininame = defini
    ini_str = fncParmGet("PATHS", "VBDEMO", "ars.ini")
    'If the ini name is found, then set
    If Len(ini_str) > 0 Then
        ininame = ini_str
    End If

    'Try to find arsgui execution path
    ini_str = fncParmGet(defstanza, "GUIPath", ininame)
    'If it can't be found, check for an env var
    If Len(ini_str) = 0 Then
        MsgBox "Cannot find GUIPath in " + ininame
    End If

    'If the path is found, then set
    If Len(ini_str) > 0 Then
        guipath = ini_str + "\"
    End If

    'Try to find the server in the ars ini file
    ini_str = fncParmGet(defstanza, "Server", ininame)
    'If it can't be found, check for an env var
    If Len(ini_str) = 0 Then
        MsgBox "Cannot find Server in " + ininame
    End If
    If Len(ini_str) > 0 Then
        server = ini_str
    End If

    'Try to find the userid in the ars ini file
    ini_str = fncParmGet(defstanza, "Userid", ininame)
    'If it can't be found, check for an env var
    If Len(ini_str) = 0 Then
        MsgBox "Cannot find Userid in " + ininame
    End If
    If Len(ini_str) > 0 Then
        userid = ini_str
    End If

    'Try to find the password in the ars ini file
    ini_str = fncParmGet(defstanza, "Password", ininame)

```

```

'If it can't be found, check for an env var
If Len(ini_str) = 0 Then
    MsgBox "Cannot find Password in " + ininame
End If
If Len(ini_str) > 0 Then
    pass = ini_str
End If
If pass = "<NULL>" Then
    pass = ""
End If

'Try to find the folder in the ars ini file
ini_str = fncParmGet(defstanza, "Folder", ininame)
folder = ini_str

End Sub
'This function returns information from the ini file.
Function fncParmGet(ByVal stanza As String, ByVal keyname As String, ByVal inifile As String)
    Dim Default, result As String
    Dim rc As Integer

    Default = ""
    result = Space$(255)

    rc = GetPrivateProfileString(stanza, keyname, Default, result, Len(result), inifile)
    If rc Then
        fncParmGet = Trim$(result)
        If Len(fncParmGet) > 1 Then
            fncParmGet = Left$(fncParmGet, Len(fncParmGet) - 1)
        End If
    Else
        fncParmGet = ""
    End If
End Function

'This function is only used to dummy up the date paid
'field of the form. The reason being is that for the
'demo, which uses the 'Baxter Bay Credit' folder,
'we cannot get this information from the database.
'This function adds approximately 20 days to the statement
'date field (which is passed in).
Public Function fncParseDate(ByVal stmtdate As String)
    Dim date_array(1 To 3) As String
    Dim searchch, workline, workchar As String
    Dim txtptr, lenstring, i As Integer
    Dim pay_day, pay_month, pay_year As Integer

    txtptr = 0
    searchch = Chr(47)
    workline = ""
    lenstring = Len(stmtdate)

'Extract chars to the first '/'
    For i = 1 To lenstring
        workchar = Mid$(stmtdate, i, 1)
        'When a '/' is found, store result, reset
        If workchar = searchch Then
            txtptr = txtptr + 1
            date_array(txtptr) = workline
            workline = ""
        'Otherwise, keep building the work string
        Else
            workline = workline + workchar
        End If
    Next

```

```

If Len(workline) > 0 Then
    txtptr = txtptr + 1
    date_array(txtptr) = workline
End If

'date_array contains three elements, the first is the month
'number, the second is the day of the month and third is
'the year. Simply check if the day of the month plus 20
'is greater than 28, if so the difference becomes the new
'day of the month and we increment the month number.
pay_day = Int(date_array(2)) + 20
pay_month = Int(date_array(1))
pay_year = Int(date_array(3))
If pay_day > 28 Then
    pay_day = pay_day - 28
    pay_month = pay_month + 1
    If pay_month > 12 Then
        pay_month = 1
        pay_year = pay_year + 1
    End If
End If

fncParseDate = LTrim(Str(pay_month)) + "/" + LTrim(Str(pay_day)) + "/" + LTrim(Str(pay_year))
End Function

```

Appendix B. Microsoft Visual C++ DDE sample program

This program sample is provided on an as-is basis. A licensee of the OnDemand product is free to copy, revise, modify, and make derivative works of this program sample as they see fit.

Global variables used by the sample program

This section defines the global variables used by the sample program.

```
#include "stdafx.h"
#include <ddeml.h>
#include <winpool.h>

#include "vcdde32.h"
#include "MainDlg.h"
#include "arsddeex.h"                                // Shipped with OnDemand

static CMainDlg * pMainDlg;
static char      RequestedData[10000];              // Returned data from DDE command
static HSZ      hsz1, hsz2;
static DWORD    DdeInstance;
static HCONV    hDdeConv;

extern CDdeTestApp * pApp;                          // Pointer to application instance

#define ERROR_MAP struct _ErrorMap
ERROR_MAP
{
    int    code;
    char * pMsg;
};

static ERROR_MAP Errors[] =
{
    { ARS_DDE_RC_UNKNOWN_COMMAND, "Unknown command." },
    { ARS_DDE_RC_PARM_NOT_SPECIFIED, "Parameter not specified." },
    { ARS_DDE_RC_INVALID_PARM_VALUE, "Invalid parameter value." },
    { ARS_DDE_RC_SERVER_ERROR, "Server error." },
    { ARS_DDE_RC_FILE_ERROR, "File error." },
    { ARS_DDE_RC_NOT_LOGGED_ON, "Not logged on." },
    { ARS_DDE_RC_MAX_FOLDERS_OPEN, "Maximum folders open." },
    { ARS_DDE_RC_FOLDER_NOT_OPEN, "Folder not open." },
    { ARS_DDE_RC_NO_DOC, "No document exists." },
    { ARS_DDE_RC_NO_ACTIVE_DOC, "No document is active." },
    { ARS_DDE_RC_USER_ACTION_IN_PROGRESS, "User action in process." },
    { ARS_DDE_RC_UNAUTHORIZED_OPERATION, "Unauthorized operation." },
    { ARS_DDE_RC_USER_CANCELLED_OPERATION, "User cancelled operation." },
    { ARS_DDE_RC_INVALID_APPL_GROUP_NAME, "Invalid Appl Group Name." },
    { ARS_DDE_RC_INVALID_APPL_NAME, "Invalid Appl Name." },
    { ARS_DDE_RC_INVALID_INTEGER_FIELD, "Invalid integer field." },
    { ARS_DDE_RC_INVALID_DECIMAL_FIELD, "Invalid decimal field." },
    { ARS_DDE_RC_INVALID_DATE_FIELD, "Invalid date field." },
    { ARS_DDE_RC_INVALID_APPLGRP_FIELD_TYPE, "Invalid Appl Group field type." }
};

#define NUM_ERRORS ( sizeof(Errors) / sizeof(ERROR_MAP) )

#define ADV_MAP struct _AdvMap
ADV_MAP
{
    char * pAdvData;
    char * pMsg;
};
```

```

static ADV_MAP Advises[] =
{ { ARS_DDE_EVENT_CRITERIA_BUTTON_1, "DDE Criteria 1 Button Clicked." },
  { ARS_DDE_EVENT_CRITERIA_BUTTON_2, "DDE Criteria 2 Button Clicked." },
  { ARS_DDE_EVENT_CRITERIA_BUTTON_3, "DDE Criteria 3 Button Clicked." },
  { ARS_DDE_EVENT_CRITERIA_BUTTON_4, "DDE Criteria 4 Button Clicked." },
  { ARS_DDE_EVENT_CRITERIA_BUTTON_5, "DDE Criteria 5 Button Clicked." },
  { ARS_DDE_EVENT_DOCLIST_BUTTON_1, "DDE Doclist 1 Button Clicked." },
  { ARS_DDE_EVENT_DOCLIST_BUTTON_2, "DDE Doclist 2 Button Clicked." },
  { ARS_DDE_EVENT_DOCLIST_BUTTON_3, "DDE Doclist 3 Button Clicked." },
  { ARS_DDE_EVENT_DOCLIST_BUTTON_4, "DDE Doclist 4 Button Clicked." },
  { ARS_DDE_EVENT_DOCLIST_BUTTON_5, "DDE Doclist 5 Button Clicked." },
  { ARS_DDE_EVENT_SWITCH_FOCUS, "Switch focus requested." },
  { ARS_DDE_EVENT_SWITCH_FOCUS_2, "Switch focus *** 2 *** requested." },
  { ARS_DDE_EVENT_SWITCH_FOCUS_3, "Switch focus *** 3 *** requested." },
  { ARS_DDE_EVENT_SWITCH_FOCUS_4, "Switch focus *** 4 *** requested." },
  { ARS_DDE_EVENT_SWITCH_FOCUS_5, "Switch focus *** 5 *** requested." } };

#define NUM_ADVISES ( sizeof(Advises) / sizeof(ADV_MAP) )

// DDE variables and functions

static HDEDDATA hDdeData, hDdeResult;

HDEDDATA FAR PASCAL DdeCallBack ( UINT      iType,
                                  UINT      iFmt,
                                  HCONV     hConv,
                                  HSZ       hsz1,
                                  HSZ       hsz2,
                                  HDEDDATA  hData,
                                  DWORD      dwData1,
                                  DWORD      dwData2 )
{
    int      j;
    char * pData;
    DWORD    data_len;

    switch ( iType )
    {
        case XTYP_DISCONNECT:
            hDdeConv = NULL;
            break;
        case XTYP_ADVDATA:
            if ( hData == NULL )
                AfxMessageBox( "hData is NULL in XTYP_ADVDATA" );
            else
            {
                pData = (char*)DdeAccessData( hData, &data_len );
                for ( j = 0; j < NUM_ADVISES; j++ )
                    if ( strcmp( Advises[j].pAdvData, pData ) == 0 )
                        break;
                AfxMessageBox( j < NUM_ADVISES
                               ? Advises[j].pMsg
                               : "Logic Error - invalid ADVDATA." );
                DdeUnaccessData( hData );
            }
            break;
    }
    return NULL;
}

static BOOL DoDdeCommand( char * pCommand, char * pParms )
{
    DWORD    data_len;
    char * pString1, * pData, * pFirstChar;
    int      j, rc;

```

```

if ( pParms == NULL )
    pParms = "";
pString1 = new char[ strlen( pCommand ) + strlen( pParms ) + 2 ];
strcpy( pString1, pCommand );
strcat( pString1, " " );
strcat( pString1, pParms );

hsz1 = DdeCreateStringHandle( DdeInstance, pString1, 0 );
hDdeResult = DdeClientTransaction( NULL,
                                   0,
                                   hDdeConv,
                                   hsz1,
                                   CF_TEXT,
                                   XTYP_REQUEST,
                                   120000L,
                                   NULL );
DdeFreeStringHandle( DdeInstance, hsz1 );

delete pString1;

RequestedData[0] = '\\0';
if ( hDdeResult == NULL )
{
    int    error;
    char * pErr;

    error = DdeGetLastError( DdeInstance );
    switch ( error )
    {
        case DMLERR_ADVACKTIMEOUT:
            pErr = "DdeClientTransaction failed with DMLERR_ADVACKTIMEOUT";
            break;
        case DMLERR_BUSY:
            pErr = "DdeClientTransaction failed with DMLERR_BUSY";
            break;
        case DMLERR_DATAACKTIMEOUT:
            pErr = "DdeClientTransaction failed with DMLERR_DATAACKTIMEOUT";
            break;
        case DMLERR_DLL_NOT_INITIALIZED:
            pErr = "DdeClientTransaction failed with DMLERR_DLL_NOT_INITIALIZED";
            break;
        case DMLERR_DLL_USAGE:
            pErr = "DdeClientTransaction failed with DMLERR_DLL_USAGE";
            break;
        case DMLERR_EXEACKTIMEOUT:
            pErr = "DdeClientTransaction failed with DMLERR_EXEACKTIMEOUT";
            break;
        case DMLERR_INVALIDPARAMETER:
            pErr = "DdeClientTransaction failed with DMLERR_INVALIDPARAMETER";
            break;
        case DMLERR_LOW_MEMORY:
            pErr = "DdeClientTransaction failed with DMLERR_LOW_MEMORY";
            break;
        case DMLERR_MEMORY_ERROR:
            pErr = "DdeClientTransaction failed with DMLERR_MEMORY_ERROR";
            break;
        case DMLERR_NO_CONV_ESTABLISHED:
            pErr = "DdeClientTransaction failed with DMLERR_NO_CONV_ESTABLISHED";
            break;
        case DMLERR_NOTPROCESSED:
            pErr = "DdeClientTransaction failed with DMLERR_NOTPROCESSED";
            break;
        case DMLERR_POKEACKTIMEOUT:
            pErr = "DdeClientTransaction failed with DMLERR_POKEACKTIMEOUT";
            break;
    }
}

```

```

        case DMLERR_POSTMSG_FAILED:
            pErr = "DdeClientTransaction failed with DMLERR_POSTMSG_FAILED";
            break;
        case DMLERR_REENTRANCY:
            pErr = "DdeClientTransaction failed with DMLERR_REENTRANCY";
            break;
        case DMLERR_SERVER_DIED:
            pErr = "DdeClientTransaction failed with DMLERR_SERVER_DIED";
            break;
        case DMLERR_SYS_ERROR:
            pErr = "DdeClientTransaction failed with DMLERR_SYS_ERROR";
            break;
        case DMLERR_UNADVACKTIMEOUT:
            pErr = "DdeClientTransaction failed with DMLERR_UNADVACKTIMEOUT";
            break;
        case DMLERR_UNFOUND_QUEUE_ID:
            pErr = "DdeClientTransaction failed with DMLERR_UNFOUND_QUEUE_ID";
            break;
    }
    AfxMessageBox( pErr );
    return FALSE;
}
else
{
    pData = (char*)DdeAccessData( hDdeResult, &data_len );
    rc = atoi( pData );
    if ( rc == ARS_DDE_RC_NO_ERROR )
    {
        pFirstChar = strchr( pData, ' ' );
        strcpy( RequestedData, &pFirstChar[1] );
    }
    else
    {
        for ( j = 0; j < NUM_ERRORS; j++ )
            if ( Errors[j].code == rc )
                break;
        AfxMessageBox( j < NUM_ERRORS
            ? Errors[j].pMsg
            : "Logic Error - invalid return code." );
    }
    DdeUnaccessData( hDdeResult );
    return rc == ARS_DDE_RC_NO_ERROR;
}
}

static BOOL DoAdviseLoop( char * pName, BOOL stop )
{
    hsz1 = DdeCreateStringHandle( DdeInstance, pName, 0 );
    hDdeResult = DdeClientTransaction( NULL,
        0,
        hDdeConv,
        hsz1,
        CF_TEXT,
        stop ? XTYP_ADVSTOP : XTYP_ADVSTART,
        120000L,
        NULL );
    DdeFreeStringHandle( DdeInstance, hsz1 );

    if ( hDdeResult == NULL )
    {
        int    error;
        char * pErr;

        error = DdeGetLastError( DdeInstance );
        switch ( error )
        {
            case DMLERR_ADVACKTIMEOUT:

```

```

        pErr = "DdeClientTransaction failed with DMLERR_ADVACKTIMEOUT";
        break;
    case DMLERR_BUSY:
        pErr = "DdeClientTransaction failed with DMLERR_BUSY";
        break;
    case DMLERR_DATAACKTIMEOUT:
        pErr = "DdeClientTransaction failed with DMLERR_DATAACKTIMEOUT";
        break;
    case DMLERR_DLL_NOT_INITIALIZED:
        pErr = "DdeClientTransaction failed with DMLERR_DLL_NOT_INITIALIZED";
        break;
    case DMLERR_DLL_USAGE:
        pErr = "DdeClientTransaction failed with DMLERR_DLL_USAGE";
        break;
    case DMLERR_EXECACKTIMEOUT:
        pErr = "DdeClientTransaction failed with DMLERR_EXECACKTIMEOUT";
        break;
    case DMLERR_INVALIDPARAMETER:
        pErr = "DdeClientTransaction failed with DMLERR_INVALIDPARAMETER";
        break;
    case DMLERR_LOW_MEMORY:
        pErr = "DdeClientTransaction failed with DMLERR_LOW_MEMORY";
        break;
    case DMLERR_MEMORY_ERROR:
        pErr = "DdeClientTransaction failed with DMLERR_MEMORY_ERROR";
        break;
    case DMLERR_NO_CONV_ESTABLISHED:
        pErr = "DdeClientTransaction failed with DMLERR_NO_CONV_ESTABLISHED";
        break;
    case DMLERR_NOTPROCESSED:
        pErr = "DdeClientTransaction failed with DMLERR_NOTPROCESSED";
        break;
    case DMLERR_POKEACKTIMEOUT:
        pErr = "DdeClientTransaction failed with DMLERR_POKEACKTIMEOUT";
        break;
    case DMLERR_POSTMSG_FAILED:
        pErr = "DdeClientTransaction failed with DMLERR_POSTMSG_FAILED";
        break;

    case DMLERR_REENTRANCY:
        pErr = "DdeClientTransaction failed with DMLERR_REENTRANCY";
        break;
    case DMLERR_SERVER_DIED:
        pErr = "DdeClientTransaction failed with DMLERR_SERVER_DIED";
        break;
    case DMLERR_SYS_ERROR:
        pErr = "DdeClientTransaction failed with DMLERR_SYS_ERROR";
        break;
    case DMLERR_UNADVACKTIMEOUT:
        pErr = "DdeClientTransaction failed with DMLERR_UNADVACKTIMEOUT";
        break;
    case DMLERR_UNFOUND_QUEUE_ID:
        pErr = "DdeClientTransaction failed with DMLERR_UNFOUND_QUEUE_ID";
        break;
    }
    AfxMessageBox( pErr );
    return FALSE;
}
else
    return TRUE;
}

////////////////////////////////////
// CMainDlg dialog

CMainDlg::CMainDlg(CWnd* pParent /*=NULL*/)
: CDialog(CMainDlg::IDD, pParent)

```

```

{
    //{AFX_DATA_INIT(CMainDlg)
    // NOTE: the ClassWizard will add member initialization here
    //}AFX_DATA_INIT
    // Note that LoadIcon does not require a subsequent DestroyIcon in Win32
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CMainDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{AFX_DATA_MAP(CMainDlg)
    // NOTE: the ClassWizard will add DDX and DDV calls here
    //}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CMainDlg, CDialog)
    //{AFX_MSG_MAP(CMainDlg)
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    ON_LBN_DBLCLK(IDC_DOCLIST, OnDblclkDoclist)
    ON_BN_CLICKED(IDC_PRINT, OnPrint)
    ON_BN_CLICKED(IDC_CLOSE, OnCloseDlg)
    //}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////
// CMainDlg message handlers

BOOL CMainDlg::OnInitDialog()
{
    CListBox * pList = (CListBox*)GetDlgItem( IDC_DOCLIST );
    CComboBox * pPrinterList = (CComboBox*)GetDlgItem( IDC_PRINTERS );
    long l, num_hits;
    char * pToken;
    PROCESS_INFORMATION pi;
    STARTUPINFO &sui;
    char cmd[300], Misc[100];
    BOOL rc;
    DWORD id;

    CDialog::OnInitDialog();

    pMainDlg = this;
    DdeInstance = 0;

    m_DocOpened = FALSE;
    m_DocID = 0;

    ( (CButton*)GetDlgItem( IDC_PRINT ) )->EnableWindow( FALSE );

    SetIcon(m_hIcon, FALSE);

    // Start up the OnDemand client

    // /I - Enable DDE Interface
    // /W - Window placement (N = hidden)
    // /V - Disable anticipation
    // /B - Disable User Confirmation
    strcpy( cmd, "g:\\ars32\\arsgui32.exe /I /W N /V /B /l g:\\ars32\\locale\\enu" );

    memset( &sui, 0, sizeof(STARTUPINFO) );
    sui.cb = sizeof(STARTUPINFO);
    rc = CreateProcess( NULL, cmd, NULL, NULL, FALSE, CREATE_NEW_CONSOLE, NULL, NULL, &sui, &pi );
    if ( !rc )
    {
        id = GetLastError( );
    }
}

```

```

FormatMessage( FORMAT_MESSAGE_FROM_SYSTEM | FORMAT_MESSAGE_IGNORE_INSERTS,
               NULL, id, 0, cmd, sizeof(cmd), NULL );
sprintf( Misc, "CreateProcessFailed - %s", cmd );
AfxMessageBox( Misc );
Misc[0] = '\\0';
}
else
{
    // Start a dde conversation with the client.

    if ( DdeInstance == 0 )
    {
        FARPROC pfnDdeCallBack;
        pfnDdeCallBack = MakeProcInstance( (FARPROC)DdeCallBack, pApp->m_hInstance );
        DdeInitialize( &DdeInstance,
                      (PFNCALLBACK)pfnDdeCallBack,
                      APPCLASS_STANDARD | APPCMD_CLIENONLY,
                      0L );
    }
    hsz1 = DdeCreateStringHandle( DdeInstance, ARS_DDE_SERVICE, 0 );
    hsz2 = DdeCreateStringHandle( DdeInstance, ARS_DDE_TOPIC, 0 );
    for ( int j = 0; j < 1000; j++ )
    {
        hDdeConv = DdeConnect( DdeInstance, hsz1, hsz2, NULL );
        if ( hDdeConv != NULL )
            break;
    }
    DdeFreeStringHandle( DdeInstance, hsz1 );
    DdeFreeStringHandle( DdeInstance, hsz2 );
    if ( hDdeConv == NULL )
        AfxMessageBox( "Unable to connect to ARSGUI32." );

    else
    {
        int k;

        // Begin sending dde commands to the client.

        Misc[0] = '/';
        Misc[1] = ARS_DDE_SWITCH_HANDLE;
        sprintf( &Misc[2], "%ld", (long)(char far *)pApp->m_pMainWnd->m_hWnd );
        strcat( Misc, " " );
        strcat( Misc, "/" );
        k = strlen( Misc );
        Misc[k++] = ARS_DDE_SWITCH_CLIENT_NAME;
        strcpy( &Misc[k], "DDE Partner 1" );
        DoDdeCommand( ARS_DDE_CMD_ENABLE_SWITCH_FOCUS, Misc );

        DoDdeCommand( ARS_DDE_CMD_LOGON, "/S gunnar /U demo /P" );

        DoDdeCommand( ARS_DDE_CMD_OPEN_FOLDER, "/F Credit Card Statementss" );

        DoDdeCommand( ARS_DDE_CMD_SEARCH_FOLDER, "" );

        if ( DoDdeCommand( ARS_DDE_CMD_GET_NUM_DOCS_IN_LIST, "" ) )
        {
            num_hits = atoi( RequestedData );
            for ( l = 0; l < num_hits; l++ )
            {
                Misc[0] = '/';
                Misc[1] = ARS_DDE_DOC_NUMBER;
                sprintf( &Misc[2], "%ld", l );

                if ( DoDdeCommand( ARS_DDE_CMD_GET_DOC_VALUES, Misc ) )
                {
                    for ( pToken = strtok( RequestedData, ARS_DDE_DATA_SEPARATOR ),
                        Misc[0] = '\\0';

```

```

        pToken != NULL;
        pToken = strtok( NULL, ARS_DDE_DATA_SEPARATOR ) )
    {
        strcat( Misc, pToken );
        strcat( Misc, " - " );
    }
    if ( Misc[0] != '\0' )
    {
        j = pList->InsertString( -1, Misc );
        pList->SetItemData( j, (DWORD)1 );
    }
    else
        break;
}

DoAdviseLoop( ARS_DDE_ADVISE_LOOP_1, FALSE );
}

if ( DoDdeCommand( ARS_DDE_CMD_GET_PRINTERS, "/L" ) )
{
    for ( pToken = strtok( RequestedData, ARS_DDE_DATA_SEPARATOR );
        pToken != NULL;
        pToken = strtok( NULL, ARS_DDE_DATA_SEPARATOR ) )
        pPrinterList->InsertString( -1, pToken );

    pPrinterList->SetCurSel( 0 );
}

return TRUE;
}

void CMainDlg::OnPaint()
{
    if (IsIconic())
    {
        CPaintDC dc(this); // device context for painting

        SendMessage(WM_ICONERASEBKGD, (WPARAM) dc.GetSafeHdc(), 0);

        // Center icon in client rectangle
        int cxIcon = GetSystemMetrics(SM_CXICON);
        int cyIcon = GetSystemMetrics(SM_CYICON);
        CRect rect;
        GetClientRect(&rect);
        int x = (rect.Width() - cxIcon + 1) / 2;
        int y = (rect.Height() - cyIcon + 1) / 2;

        // Draw the icon
        dc.DrawIcon(x, y, m_hIcon);
    }
    else
    {
        CDialog::OnPaint();
    }
}

// The system calls this to obtain the cursor to display while the user drags
// the minimized window.
HCURSOR CMainDlg::OnQueryDragIcon()
{
    return (HCURSOR) m_hIcon;
}

void CMainDlg::OnDbClickDoclist()

```

```

{
    CListBox * pDocsList;
    char printer[100];
    char Misc[100];

    pDocsList = (CListBox*)GetDlgItem( IDC_DOCLIST );

    if( m_DocOpened )
    {
        sprintf( Misc, "/D %d", m_DocID );
        DoDdeCommand( ARS_DDE_CMD_CLOSE_DOC, Misc );
    }

    Misc[0] = '/';
    Misc[1] = ARS_DDE_DOC_NUMBER;
    sprintf( &Misc[2], "%d", (int)pDocsList->GetCurSel() );
    if ( DoDdeCommand( ARS_DDE_CMD_OPEN_DOC, Misc ) )
    {
        m_DocID = atol( RequestedData );
        m_DocOpened = TRUE;

        DoDdeCommand( ARS_DDE_CMD_SHOW_WINDOW, "/W" );
    }
    else
    {
        m_DocID = 0;
        m_DocOpened = TRUE;
    }

    GetDlgItem( IDC_PRINTERS )->GetWindowText( printer, sizeof(printer) );
    if( printer != NULL && printer[0] != '\0' )
        ( CButton*)GetDlgItem( IDC_PRINT )->EnableWindow( TRUE );
}

void CMainDlg::OnPrint()
{
    char printer[100];
    char Misc[100];

    GetDlgItem( IDC_PRINTERS )->GetWindowText( printer, sizeof(printer) );

    Misc[0] = '\0';
    sprintf( Misc, "/L %s", printer );
    DoDdeCommand( ARS_DDE_CMD_PRINT_DOC, Misc );
}

void CMainDlg::OnCloseDlg()
{
    char Misc[100];

    if( m_DocOpened )
    {
        sprintf( Misc, "/D %d", m_DocID );
        DoDdeCommand( ARS_DDE_CMD_CLOSE_DOC, Misc );
    }

    DoDdeCommand( ARS_DDE_CMD_CLOSE_FOLDER, "" );
    DoDdeCommand( ARS_DDE_CMD_LOGOFF, "" );
    DoDdeCommand( ARS_DDE_CMD_EXIT, "" );

    EndDialog(0);
}

```

Appendix C. Microsoft Visual Basic OLE sample program

This program sample is provided on an as-is basis. A licensee of the OnDemand product is free to copy, revise, modify, and make derivative works of this program sample as they see fit.

Global variables used by the sample program

```
Option Explicit
Global Const defini = "vbarsole.ini" 'Default ini file name
Global Const defstanza = "VBARSOLE" 'Default stanza

' The following constants were obtained from arsoleex.h
Global Const ARS_OLE_USER_MSG_MODE_SHOW = 1
Global Const ARS_OLE_USER_MSG_MODE_SUPPRESS = 2

Global Const ARS_OLE_FIND_FIRST = 1
Global Const ARS_OLE_FIND_PREV = 2
Global Const ARS_OLE_FIND_NEXT = 3

Global Const ARS_OLE_OPR_EQUAL = 1
Global Const ARS_OLE_OPR_NOT_EQUAL = 2
Global Const ARS_OLE_OPR_LESS_THAN = 3
Global Const ARS_OLE_OPR_LESS_THAN_OR_EQUAL = 4
Global Const ARS_OLE_OPR_GREATER_THAN = 5
Global Const ARS_OLE_OPR_GREATER_THAN_OR_EQUAL = 6
Global Const ARS_OLE_OPR_BETWEEN = 7
Global Const ARS_OLE_OPR_NOT_BETWEEN = 8
Global Const ARS_OLE_OPR_IN = 9
Global Const ARS_OLE_OPR_NOT_IN = 10
Global Const ARS_OLE_OPR_LIKE = 11
Global Const ARS_OLE_OPR_NOT_LIKE = 12
Global Const ARS_OLE_RC_SUCCESS = 0
Global Const ARS_OLE_RC_NO_MEMORY = 1
Global Const ARS_OLE_RC_SERVER_ERROR = 2
Global Const ARS_OLE_RC_USER_CANCELLED = 3
Global Const ARS_OLE_RC_INVALID_DIRECTORY = 4
Global Const ARS_OLE_RC_UNAUTHORIZED_OPERATION = 5
Global Const ARS_OLE_RC_NOT_SUPPORTED = 6
Global Const ARS_OLE_RC_FILE_ERROR = 7
Global Const ARS_OLE_RC_ALREADY_LOGGED_ON = 8
Global Const ARS_OLE_RC_NOT_LOGGED_ON = 9
Global Const ARS_OLE_RC_FOLDER_ALREADY_OPEN = 10
Global Const ARS_OLE_RC_FOLDER_NOT_OPEN = 11
Global Const ARS_OLE_RC_UNKNOWN_FOLDER = 12
Global Const ARS_OLE_RC_NO_FOLDERS_AVAILABLE = 13
Global Const ARS_OLE_RC_DOC_NOT_OPEN = 14
Global Const ARS_OLE_RC_DOC_ALREADY_OPEN = 15
Global Const ARS_OLE_RC_NO_DOC_AVAILABLE = 16
Global Const ARS_OLE_RC_OPEN_DOC_FAILED = 17
Global Const ARS_OLE_RC_DOC_CANNOT_HORZ_SCROLL = 18
Global Const ARS_OLE_RC_INVALID_DOC_INDEX = 19
Global Const ARS_OLE_RC_INVALID_CONTROL_ID = 20
Global Const ARS_OLE_RC_INVALID_FIELD = 21
Global Const ARS_OLE_RC_INVALID_OPERATOR = 22
Global Const ARS_OLE_RC_INVALID_MESSAGE_MODE = 23
Global Const ARS_OLE_RC_INVALID_ZOOM_PERCENT = 24
Global Const ARS_OLE_RC_INVALID_PAGE_NUMBER = 25
Global Const ARS_OLE_RC_INVALID_ROTATION = 26
Global Const ARS_OLE_RC_INVALID_COLOR = 27
Global Const ARS_OLE_RC_INVALID_COPIES = 28
Global Const ARS_OLE_RC_INVALID_ORIENTATION = 29
```

```

Global Const ARS_OLE_RC_INVALID_PRINTER = 30
Global Const ARS_OLE_RC_INVALID_FIND_TYPE = 31
Global Const ARS_OLE_RC_ERROR_DURING_PRINT = 32
Global Const ARS_OLE_SCROLL_LINEUP = 0
Global Const ARS_OLE_SCROLL_LINELEFT = 0
Global Const ARS_OLE_SCROLL_LINEDOWN = 1
Global Const ARS_OLE_SCROLL_LINERIGHT = 1
Global Const ARS_OLE_SCROLL_PAGEUP = 2
Global Const ARS_OLE_SCROLL_PAGELEFT = 2
Global Const ARS_OLE_SCROLL_PAGEDOWN = 3
Global Const ARS_OLE_SCROLL_PAGERIGHT = 3
Global Const ARS_OLE_SCROLL_THUMBPOSITION = 4
Global Const ARS_OLE_SCROLL_THUMBTRACK = 5
Global Const ARS_OLE_SCROLL_TOP = 6
Global Const ARS_OLE_SCROLL_LEFT = 6
Global Const ARS_OLE_SCROLL_BOTTOM = 7
Global Const ARS_OLE_SCROLL_RIGHT = 7
Global Const ARS_OLE_SCROLL_ENDSCROLL = 8

Global Const DocZoom = 110

Global server As String           'Server name
Global userid As String           'userid
Global password As String         'password
Global folder As String           'folder

Global doc_id As Integer
Global doc_values(0 To 8) As String

Global OpenDoc As Boolean
Global VertScroll10ld As Integer
Global HorzScroll10ld As Integer

'Define the Windows APIs used by the program
Declare Function GetPrivateProfileInt Lib "kernel32" Alias "GetPrivateProfileIntA"
    (ByVal lpApplicationName As String, ByVal lpKeyName As String, ByVal nDefault As Long,
    ByVal lpFileName As String) As Long
Declare Function GetPrivateProfileString Lib "kernel32" Alias "GetPrivateProfileStringA"
    (ByVal lpApplicationName As String, ByVal lpKeyName As Any, ByVal lpDefault As String,
    ByVal lpReturnedString As String, ByVal nSize As Long, ByVal lpFileName As String) As Long
Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
'Declare Function GetPrivateProfileString Lib "kernel32" (ByVal sname$, ByVal Kname$, ByVal Def$,
    ByVal ret$, ByVal Size%, ByVal FName$) As Integer

Public Sub Main()
    Dim rc As Integer

    Load frmMain
    Load frmInit

    doc_id = 0
    OpenDoc = False
    VertScroll10ld = 0
    HorzScroll10ld = 0

    'Disable "View" buttons
    frmMain.cmdViewStmt1.Enabled = False
    frmMain.cmdViewStmt2.Enabled = False
    frmMain.cmdViewStmt3.Enabled = False

    'Because we need the ocx file and the arssck32.dll
    'which reside in the ars directory I will require
    'that this exe and its ini file also reside in the
    'ars install directory.

    'I should check for ini file existance first.

```

```

'Try to find the "Server" name in the ini file
server = fncParmGet(defstanza, "Server", defini)
If Len(server) = 0 Then
    MsgBox "Cannot find Server in " + defini
End
End If

'Try to find the "Userid" name in the ini file
userid = fncParmGet(defstanza, "Userid", defini)
If Len(userid) = 0 Then
    MsgBox "Cannot find Userid in " + defini
End
End If

'Try to find the "Server" name in the ini file
password = fncParmGet(defstanza, "Password", defini)
If Len(password) = 0 Then
    password = " "
End If

'Try to find the "Folder" name in the ini file
folder = fncParmGet(defstanza, "Folder", defini)
If Len(folder) = 0 Then
    MsgBox "Cannot find Folder in " + defini
End
End If

'The following call is for debug.
rc = frmMain.ArsOle.SetUserMessageMode(ARS_OLE_USER_MSG_MODE_SHOW)

frmInit.Show
frmInit.pnlStatus.Caption = "Logging on to Server..."

'Attempt to logon to the specified server.
rc = frmMain.ArsOle.Logon(server, userid, password)
If rc <> ARS_OLE_RC_SUCCESS Then
    frmInit.pnlStatus.Caption = ""
    MsgBox "Cannot Logon to server " + server + "; rc = " + Str(rc)
End
End If

frmInit.SetFocus

'Attempt to open the folder specified in the ini file.
frmInit.pnlStatus.Caption = "Opening folder..."
rc = frmMain.ArsOle.OpenFolder(folder)
If rc <> ARS_OLE_RC_SUCCESS Then
    frmMain.pnlStatus.Caption = ""
    MsgBox "Cannot open folder " + folder + "; rc = " + Str(rc)
    frmMain.ArsOle.Logoff
End
End If

frmInit.SetFocus
frmInit.pnlStatus.Caption = ""
frmInit.Hide

frmMain.Show
End Sub

'This function returns information from the ini file.
Function fncParmGet(ByVal stanza As String, ByVal keyname As String, ByVal inifile As String)
    Dim Default, result As String
    Dim rc As Integer

    Default = ""
    result = Space$(255)

```

```

rc = GetPrivateProfileString(stanza, keyname, Default, result, Len(result), inifile)
If rc Then
    fncParmGet = Trim$(result)
    If Len(fncParmGet) > 1 Then
        fncParmGet = Left$(fncParmGet, Len(fncParmGet) - 1)
    End If
Else
    fncParmGet = ""
End If

End Function

'This function is only used to dummy up the date paid
'field of the form because for the
'demo, which uses the 'Baxter Bay Credit' folder,
'we cannot get this information from the database.
'This function adds approximately 20 days to the statement
'date field (which is passed in).
Public Function fncParseDate(ByVal stmtdate As String)
    Dim date_array(1 To 3) As String
    Dim searchch, workline, workchar As String
    Dim txtptr, lenstring, i As Integer
    Dim pay_day, pay_month, pay_year As Integer

    txtptr = 0
    searchch = Chr(47)
    workline = ""
    lenstring = Len(stmtdate)

    'Extract chars to the first '/'
    For i = 1 To lenstring
        workchar = Mid$(stmtdate, i, 1)
        'When a '/' is found, store result, reset
        If workchar = searchch Then
            txtptr = txtptr + 1
            date_array(txtptr) = workline
            workline = ""
        'Otherwise, keep building the work string
        Else
            workline = workline + workchar
        End If
    Next

    If Len(workline) > 0 Then
        txtptr = txtptr + 1
        date_array(txtptr) = workline
    End If

    'date_array contains three elements, the first is the month
    'number, the second is the day of the month, and the third is
    'the year. Simply check if the day of the month plus 20
    'is greater than 28, if so the difference becomes the new
    'day of the month, and we increment the month number.
    pay_day = Int(date_array(2)) + 20
    pay_month = Int(date_array(1))
    pay_year = Int(date_array(3))
    If pay_day > 28 Then
        pay_day = pay_day - 28
        pay_month = pay_month + 1
        If pay_month > 12 Then
            pay_month = 1
            pay_year = pay_year + 1
        End If
    End If

    fncParseDate = LTrim(Str(pay_month)) + "/" + LTrim(Str(pay_day)) + "/" + LTrim(Str(pay_year))

```

End Function

```
Private Sub cmdCustInfo_Click()
    Dim rc As Integer
    Dim acct_num, ini_str As String
    Dim first_num, second_num, third_num As Integer
    Dim temp As String
    Dim numdocs As Variant

    If OpenDoc Then
        pnlStatus.Caption = "Closing document..."
        rc = ArsOle.CloseDoc()
        pnlStatus.Caption = ""
    End If

    'Clear the payment record fields
    pnlStmtDate1.Caption = ""
    pnlStmtDate2.Caption = ""
    pnlStmtDate3.Caption = ""
    pnlBalance1.Caption = ""
    pnlBalance2.Caption = ""
    pnlBalance3.Caption = ""
    pnlDatePaid1.Caption = ""
    pnlDatePaid2.Caption = ""
    pnlDatePaid3.Caption = ""

    'Clear the customer information fields
    pnlNameData = ""
    pnlSOSData = ""
    pnlDOBData = ""
    pnlMNameData = ""
    pnlAddrData1 = ""
    pnlAddrData2 = ""
    pnlPhoneData = ""

    'Disable "View" buttons
    cmdViewStmt1.Enabled = False
    cmdViewStmt2.Enabled = False
    cmdViewStmt3.Enabled = False

    'Look up the account number, contained in the pnlAcctnumData text field
    'in the arsvblan.ini file. If found, read the respective
    'fields. If not found display error message.
    acct_num = txtAcctnumData.Text

    'Do at least a little validation.
    If Len(acct_num) <> 11 Then
        MsgBox "Correct format for account # is 000-000-000"
        Exit Sub
    End If

    'If we have gotten to here we know that we have an account
    'number of the format 000-000-000. If either of the first
    'two sections of the number are non-zero or if the third
    'section is not between 001-046 then default to the account
    'number 000-000-001.
    first_num = Int(Mid(acct_num, 1, 3))
    second_num = Int(Mid(acct_num, 5, 3))
    third_num = Int(Mid(acct_num, 9, 3))
    If first_num <> 0 Or second_num <> 0 Or third_num > 46 Then
        acct_num = "000-000-001"
    ElseIf third_num = 0 Then
        MsgBox "Invalid account number!"
        Exit Sub
    End If

    ini_str = fncParmGet(acct_num, "Name", defini)
```

```

If Len(ini_str) = 0 Then
    MsgBox "'Name' field not found for acct#" + acct_num + "in " + ininame
    Exit Sub
End If
pnlNameData.Caption = " " + ini_str

ini_str = fncParmGet(acct_num, "SSN", defini)
If Len(ini_str) = 0 Then
    MsgBox "'SSN' field not found for acct#" + acct_num + "in " + ininame
    Exit Sub
End If
pnlSSNData.Caption = " " + ini_str

ini_str = fncParmGet(acct_num, "DOB", defini)
If Len(ini_str) = 0 Then
    MsgBox "'DOB' field not found for acct#" + acct_num + "in " + ininame
    Exit Sub
End If
pnlDOBData.Caption = " " + ini_str

ini_str = fncParmGet(acct_num, "MaidenName", defini)
If Len(ini_str) = 0 Then
    MsgBox "'MaidenName' field not found for acct#" + acct_num + "in " + ininame
    Exit Sub
End If
pnlMNameData.Caption = " " + ini_str

ini_str = fncParmGet(acct_num, "StreetAddress", defini)
If Len(ini_str) = 0 Then
    MsgBox "'StreetAddress' field not found for acct#" + acct_num + "in " + ininame
    Exit Sub
End If
pnlAddrData1.Caption = " " + ini_str

ini_str = fncParmGet(acct_num, "CityStateZip", defini)
If Len(ini_str) = 0 Then
    MsgBox "'CityStateZip' field not found for acct#" + acct_num + "in " + ininame
    Exit Sub
End If
pnlAddrData2.Caption = " " + ini_str

ini_str = fncParmGet(acct_num, "PhoneNum", defini)
If Len(ini_str) = 0 Then
    MsgBox "'PhoneNum' field not found for acct#" + acct_num + "in " + ininame
    Exit Sub
End If
pnlPhoneData.Caption = " " + ini_str

'We are changing customer accounts so before we get new customer
'information, close old customers open documents.
If doc_id <> 0 Then
    rc = ArsOle.CloseDoc
    If rc <> ARS_OLE_RC_SUCCESS Then
        pnlStatus.Caption = ""
        MsgBox "Cannot set folder search criteria; rc = " + Str(rc)
        ArsOle.CloseFolder
        ArsOle.Logoff
    End If
End If

pnlStatus.Caption = "Searching folder..."
rc = ArsOle.SetFolderSearchFieldData("Account", ARS_OLE_OPR_EQUAL, acct_num, "")

If rc <> ARS_OLE_RC_SUCCESS Then
    pnlStatus.Caption = ""
    MsgBox "Cannot set folder search criteria; rc = " + Str(rc)

```

```

        Ars01e.CloseFolder
        Ars01e.Logoff
    End
End If

rc = Ars01e.SearchFolder(0)
If rc <> ARS_OLE_RC_SUCCESS Then
    pnlStatus.Caption = ""
    MsgBox "Search folder failed; rc = " + Str(rc)
    Ars01e.CloseFolder
    Ars01e.Logoff
End
End If

rc = Ars01e.GetNumDocsInList(numdocs)

rc = Ars01e.GetDocDisplayValue(numdocs - 1, 0, temp)
pnlStmtDate1.Caption = temp
pnlDatePaid1.Caption = fncParseDate(temp)
rc = Ars01e.GetDocDisplayValue(numdocs - 2, 0, temp)
pnlStmtDate2.Caption = temp
pnlDatePaid2.Caption = fncParseDate(temp)
rc = Ars01e.GetDocDisplayValue(numdocs - 3, 0, temp)
pnlStmtDate3.Caption = temp
pnlDatePaid3.Caption = fncParseDate(temp)

rc = Ars01e.GetDocDisplayValue(numdocs - 1, 3, temp)
pnlBalance1.Caption = temp
rc = Ars01e.GetDocDisplayValue(numdocs - 2, 3, temp)
pnlBalance2.Caption = temp
rc = Ars01e.GetDocDisplayValue(numdocs - 3, 3, temp)
pnlBalance3.Caption = temp

'Enable "View" buttons
cmdViewStmt1.Enabled = True
cmdViewStmt2.Enabled = True
cmdViewStmt3.Enabled = True

pnlStatus.Caption = ""
End Sub

Private Sub cmdExit_Click()
    'If OpenDoc Then
    '    Ars01e.CloseDoc
    'End If
    'Ars01e.CloseFolder
    'Ars01e.Logoff
End
End Sub

Private Sub cmdViewStmt1_Click()
    Dim numdocs As Variant

    rc = Ars01e.GetNumDocsInList(numdocs)

    If OpenDoc Then
        pnlStatus.Caption = "Closing document..."
        rc = Ars01e.CloseDoc()
        pnlStatus.Caption = ""
        vscrollDoc.Value = 0
        hscrollDoc.Value = 0
    End If

    pnlStatus.Caption = "Retrieving document..."
    rc = Ars01e.OpenDoc(numdocs - 1, "", 0)
    If rc <> ARS_OLE_RC_SUCCESS Then
        pnlStatus.Caption = ""
    End If
End Sub

```

```

        MsgBox "Open document failed; rc = " + Str(rc)
        ArsOle.CloseFolder
        ArsOle.Logoff
    End
End If
pnlStatus.Caption = ""

OpenDoc = True

rc = ArsOle.SetDocZoom(DocZoom, horzPos, vertPos)
vscrollDoc.Value = vertPos
hscrollDoc.Value = horzPos
End Sub

Private Sub cmdViewStmt2_Click()
    Dim numdocs As Variant

    rc = ArsOle.GetNumDocsInList(numdocs)

    If OpenDoc Then
        pnlStatus.Caption = "Closing document..."
        rc = ArsOle.CloseDoc()
        pnlStatus.Caption = ""
        vscrollDoc.Value = 0
        hscrollDoc.Value = 0
    End If

    pnlStatus.Caption = "Retrieving document..."
    rc = ArsOle.OpenDoc(numdocs - 2, "", 0)
    If rc <> ARS_OLE_RC_SUCCESS Then
        pnlStatus.Caption = ""
        MsgBox "Open document failed; rc = " + Str(rc)
        ArsOle.CloseFolder
        ArsOle.Logoff
    End
    End If
    pnlStatus.Caption = ""

    OpenDoc = True

    rc = ArsOle.SetDocZoom(DocZoom, horzPos, vertPos)
End Sub

Private Sub cmdViewStmt3_Click()
    Dim numdocs As Variant

    rc = ArsOle.GetNumDocsInList(numdocs)

    If OpenDoc Then
        pnlStatus.Caption = "Closing document..."
        rc = ArsOle.CloseDoc()
        pnlStatus.Caption = ""
        vscrollDoc.Value = 0
        hscrollDoc.Value = 0
    End If

    pnlStatus.Caption = "Retrieving document..."
    rc = ArsOle.OpenDoc(numdocs - 3, "", 0)
    If rc <> ARS_OLE_RC_SUCCESS Then
        pnlStatus.Caption = ""
        MsgBox "Open document failed; rc = " + Str(rc)
        ArsOle.CloseFolder
        ArsOle.Logoff
    End
    End If
    pnlStatus.Caption = ""

```

```

    OpenDoc = True

    rc = Ars01e.SetDocZoom(DocZoom, horzPos, vertPos)
End Sub

Private Sub Form_Unload(Cancel As Integer)
    If OpenDoc Then
        Ars01e.CloseDoc
    End If
    Ars01e.CloseFolder
    Ars01e.Logoff
End Sub
End Sub

Private Sub hscrollDoc_Change()
    Dim Diff As Integer
    Dim rc As Integer
    Dim ScrollCode As Integer
    Dim NewPos As Variant

    NewPos = 0
    Diff = hscrollDoc.Value - HorzScroll101d
    If Diff = hscrollDoc.LargeChange Then
        ScrollCode = ARS_OLE_SCROLL_PAGERIGHT
        rc = Ars01e.ScrollDocHorz(ScrollCode, NewPos)
        hscrollDoc.Value = NewPos
    ElseIf Diff = -hscrollDoc.LargeChange Then
        ScrollCode = ARS_OLE_SCROLL_PAGELEFT
        rc = Ars01e.ScrollDocHorz(ScrollCode, NewPos)
        hscrollDoc.Value = NewPos
    ElseIf Diff = hscrollDoc.SmallChange Then
        ScrollCode = ARS_OLE_SCROLL_LINERIGHT
        rc = Ars01e.ScrollDocHorz(ScrollCode, NewPos)
        hscrollDoc.Value = NewPos
    ElseIf Diff = -hscrollDoc.SmallChange Then
        ScrollCode = ARS_OLE_SCROLL_LINELEFT
        rc = Ars01e.ScrollDocHorz(ScrollCode, NewPos)
        hscrollDoc.Value = NewPos
    Else
        ScrollCode = ARS_OLE_SCROLL_THUMBPOSITION
        NewPos = hscrollDoc.Value
        rc = Ars01e.ScrollDocHorz(ScrollCode, NewPos)
        HorzScroll101d = hscrollDoc.Value
    End If

    HorzScroll101d = hscrollDoc.Value
End Sub

Private Sub vscrollDoc_Change()
    Dim Diff As Integer
    Dim rc As Integer
    Dim ScrollCode As Integer
    Dim NewPos As Variant

    NewPos = 0
    Diff = vscrollDoc.Value - VertScroll101d
    If Diff = vscrollDoc.LargeChange Then
        ScrollCode = ARS_OLE_SCROLL_PAGEDOWN
        rc = Ars01e.ScrollDocVert(ScrollCode, NewPos)
        VertScroll101d = NewPos
        vscrollDoc.Value = NewPos
    ElseIf Diff = -vscrollDoc.LargeChange Then
        ScrollCode = ARS_OLE_SCROLL_PAGEUP
        rc = Ars01e.ScrollDocVert(ScrollCode, NewPos)
        VertScroll101d = NewPos
        vscrollDoc.Value = NewPos
    End If

```

```

ElseIf Diff = vscrollDoc.SmallChange Then
    ScrollCode = ARS_OLE_SCROLL_LINEDOWN
    rc = Ars01e.ScrollDocVert(ScrollCode, NewPos)
    VertScroll101d = NewPos
    vscrollDoc.Value = NewPos
ElseIf Diff = -vscrollDoc.SmallChange Then
    ScrollCode = ARS_OLE_SCROLL_LINEUP
    rc = Ars01e.ScrollDocVert(ScrollCode, NewPos)
    VertScroll101d = NewPos
    vscrollDoc.Value = NewPos
Else
    ScrollCode = ARS_OLE_SCROLL_THUMBPOSITION
    NewPos = vscrollDoc.Value
    rc = Ars01e.ScrollDocVert(ScrollCode, NewPos)
    VertScroll101d = vscrollDoc.Value
End If
End Sub

```

Appendix D. Microsoft Visual C++ OLE sample program

This program sample is provided on an as-is basis. A licensee of the OnDemand product is free to copy, revise, modify, and make derivative works of this program sample as they see fit.

```
#include "stdafx.h"
#include [winspool.h]

#include "vcole32.h"
#include "MainDlg.h"
#include "AttrsDlg.h"

static CMainDlg * pMainDlg;

#define COLOR_MAP struct _ColorMap
COLOR_MAP
{
    short color;
    char * pText;
};

static COLOR_MAP Colors[;]; =
{ { ARS_OLE_COLOR_BLACK, "Black" },
  { ARS_OLE_COLOR_WHITE, "White" },
  { ARS_OLE_COLOR_RED, "Red" },
  { ARS_OLE_COLOR_BLUE, "Blue" },
  { ARS_OLE_COLOR_GREEN, "Green" },
  { ARS_OLE_COLOR_YELLOW, "Yellow" },
  { ARS_OLE_COLOR_GREY, "Grey" },
  { ARS_OLE_COLOR_CYAN, "Cyan" },
  { ARS_OLE_COLOR_MAGENTA, "Magenta" } };

#define NUM_COLORS ( sizeof(Colors) / sizeof(COLOR_MAP) )

#define ERROR_MAP struct _ErrorMap
ERROR_MAP
{
    short code;
    char * pMsg;
};

static ERROR_MAP Errors[;]; =
{ { ARS_OLE_RC_NO_MEMORY, "insufficient memory" },
  { ARS_OLE_RC_UNKNOWN_FOLDER, "unknown folder" },
  { ARS_OLE_RC_NO_FOLDERS_AVAILABLE, "no folders availble" },
  { ARS_OLE_RC_SERVER_ERROR, "server error" },
  { ARS_OLE_RC_FOLDER_ALREADY_OPEN, "folder already open" },
  { ARS_OLE_RC_NOT_LOGGED_ON, "not logged on" },
  { ARS_OLE_RC_ALREADY_LOGGED_ON, "already logged on" },
  { ARS_OLE_RC_INVALID_DIRECTORY, "invalid directory" },
  { ARS_OLE_RC_FOLDER_NOT_OPEN, "folder not open" },
  { ARS_OLE_RC_DOC_ALREADY_OPEN, "document already open" },
  { ARS_OLE_RC_DOC_NOT_OPEN, "no document is open" },
  { ARS_OLE_RC_OPEN_DOC_FAILED, "open doc failed" },
  { ARS_OLE_RC_UNAUTHORIZED_OPERATION, "unauthorized operation" },
  { ARS_OLE_RC_USER_CANCELLED, "user cancelled operation" },
  { ARS_OLE_RC_INVALID_INDEX, "invalid index" },
  { ARS_OLE_RC_INVALID_FIELD, "invalid field" },
  { ARS_OLE_RC_INVALID_OPERATOR, "invalid operator" },
  { ARS_OLE_RC_INVALID_MESSAGE_MODE, "invalid message mode" },
  { ARS_OLE_RC_INVALID_ZOOM_PERCENT, "invalid zoom percent" },
  { ARS_OLE_RC_DOC_CANNOT_HORZ_SCROLL, "cannot horz scroll" },
```

```

    { ARS_OLE_RC_INVALID_PAGE_NUMBER,      "invalid page number" },
    { ARS_OLE_RC_INVALID_CONTROL_ID,       "invalid other control" },
    { ARS_OLE_RC_INVALID_ROTATION,        "invalid rotation" },
    { ARS_OLE_RC_NO_DOC_AVAILABLE,        "no document for hit" },
    { ARS_OLE_RC_NOT_SUPPORTED,           "not supported" },
    { ARS_OLE_RC_FILE_ERROR,              "file error" },
    { ARS_OLE_RC_INVALID_COPIES,          "invalid copies" },
    { ARS_OLE_RC_INVALID_ORIENTATION,     "invalid orientation" },
    { ARS_OLE_RC_INVALID_PRINTER,         "invalid printer" },
    { ARS_OLE_RC_INVALID_FIND_TYPE,       "invalid find type" },
    { ARS_OLE_RC_ERROR_DURING_PRINT,      "error during print" },
    { ARS_OLE_RC_INVALID_COLOR,           "invalid color" } };

#define NUM_ERRORS ( sizeof(Errors) / sizeof(ERROR_MAP) )

BEGIN_MESSAGE_MAP(CMainDlg, CDialog)
    //{AFX_MSG_MAP(CMainDlg)
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    ON_BN_CLICKED(IDC_PRINT, OnPrint)
    ON_LBN_DBLCLK(IDC_DOCLIST, OnDb1clkDoclist)
    ON_BN_CLICKED(IDC_CLOSE, OnCloseDlg)
    ON_WM_HSCROLL()
    ON_WM_VSCROLL()
    ON_WM_CLOSE()
    ON_BN_CLICKED(IDC_ATTRIBUTES, OnSetDocAttrs)
    //}AFX_MSG_MAP
END_MESSAGE_MAP()

BEGIN_EVENTSINK_MAP(CMainDlg, CDialog)
    //{AFX_EVENTSINK_MAP(CMainDlg)
    ON_EVENT(CMainDlg, IDC_ARSCtrl, 4, OnFolderSearchCompletedArsctrl, VTS_NONE)
    ON_EVENT(CMainDlg, IDC_ARSCtrl, 3, OnDocOpenedArsctrl, VTS_NONE)
    ON_EVENT(CMainDlg, IDC_ARSCtrl, 1, OnDocClosedArsctrl, VTS_NONE)
    //}AFX_EVENTSINK_MAP
END_EVENTSINK_MAP()

CMainDlg::CMainDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CMainDlg::IDD, pParent)
{
    //{AFX_DATA_INIT(CMainDlg)
    // NOTE: the ClassWizard will add member initialization here
    //}AFX_DATA_INIT
    // Note that LoadIcon does not require a subsequent DestroyIcon in Win32
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

CMainDlg::CMainDlg()
{
}

void CMainDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{AFX_DATA_MAP(CMainDlg)
    // NOTE: the ClassWizard will add DDX and DDV calls here
    //}AFX_DATA_MAP
}

////////////////////////////////////
// CMainDlg message handlers

BOOL CMainDlg::OnInitDialog()
{
    VARIANT    var;
    short      rc;

```

```

char    Misc[1024];
CArsOle * pArsCtrl;
int      index;

pMainDlg = this;

m_DocOpened = FALSE;

( (CButton*)GetDlgItem( IDC_PRINT ) )->EnableWindow( FALSE );
( (CButton*)GetDlgItem( IDC_ATTRIBUTES ) )->EnableWindow( FALSE );

SetIcon(m_hIcon, FALSE);

pArsCtrl = (CArsOle*)GetDlgItem( IDC_ARSCtrl );

( (CScrollBar*)GetDlgItem( IDC_HORZ_SCROLLBAR ) )->SetScrollRange( 0, ARS_OLE_SCROLL_RANGE );
( (CScrollBar*)GetDlgItem( IDC_VERT_SCROLLBAR ) )->SetScrollRange( 0, ARS_OLE_SCROLL_RANGE );
( (CScrollBar*)GetDlgItem( IDC_HORZ_SCROLLBAR ) )->ShowScrollBar( FALSE );
( (CScrollBar*)GetDlgItem( IDC_VERT_SCROLLBAR ) )->ShowScrollBar( FALSE );

// Begin calling functions in the OnDemand OLE control

rc = pArsCtrl->SetUserMessageMode( ARS_OLE_USER_MSG_MODE_SHOW );
if ( rc != ARS_OLE_RC_SUCCESS )
    DisplayMsg( rc, "SetUserMessageMode" );

rc = pArsCtrl->Logon( "gunnar", "demo", " " );
if ( rc != ARS_OLE_RC_SUCCESS )
    DisplayMsg( rc, "Logon" );

rc = pArsCtrl->OpenFolder( "Credit Card Statements" );
if ( rc != ARS_OLE_RC_SUCCESS )
{
    DisplayMsg( rc, "OpenFolder" );
    return FALSE;
}

rc = pArsCtrl->GetNumFolderDisplayFields( &var );
if ( rc != ARS_OLE_RC_SUCCESS )
{
    DisplayMsg( rc, "GetNumFolderDisplayFields" );
    return FALSE;
}
m_NumDisplayFields = var.iVal;

rc = pArsCtrl->SearchFolder( FALSE );
if ( rc != ARS_OLE_RC_SUCCESS )
{
    DisplayMsg( rc, "SearchFolder" );
    return FALSE;
}

// Get the list of local printers
CComboBox * pPrintersList;
PRINTER_INFO_2 * pPrinterInfoArray;
DWORD size, num_printer_infos, printer_info_index, port_index;
char * pPortNames, * pIndividualPortName;
pPrintersList = (CComboBox*)GetDlgItem( IDC_PRINTERS );
EnumPrinters( PRINTER_ENUM_LOCAL | PRINTER_ENUM_CONNECTIONS,
              NULL, 2, NULL, 0, &size, &num_printer_infos );
pPrinterInfoArray = (PRINTER_INFO_2*)new char[ size ];
EnumPrinters( PRINTER_ENUM_LOCAL | PRINTER_ENUM_CONNECTIONS,
              NULL, 2, (BYTE*)pPrinterInfoArray, size, &size, &num_printer_infos );
if ( num_printer_infos > 0 )
{
    for ( printer_info_index = 0;
          printer_info_index < num_printer_infos; printer_info_index++ )

```

```

    {
        pPortNames =
            new char[ strlen( pPrinterInfoArray[printer_info_index;].pPortName; ) + 1 ];
        strcpy( pPortNames, pPrinterInfoArray[printer_info_index;].pPortName; );
        for ( pIndividualPortName = strtok( pPortNames, "," ), port_index = 0;
            pIndividualPortName != NULL;
            pIndividualPortName = strtok( NULL, "," ), port_index++ )
        {
            strcpy( Misc, pPrinterInfoArray[printer_info_index;].pPrinterName; );
            strcat( Misc, " on " );
            strcat( Misc, pIndividualPortName );
            index = pPrintersList->AddString( Misc );
        }
        delete pPortNames;
    }
    pPrintersList->SetCurSel( 0 );
}
delete [;]; pPrinterInfoArray;

return TRUE;
}

void CMainDlg::DisplayMsg( short rc, char * pMsg )
{
    int j;
    char Misc[1024;];

    if ( rc == ARS_OLE_RC_SUCCESS )
        AfxMessageBox( pMsg );
    else
    {
        for ( j = 0; j < NUM_ERRORS; j++ )
            if ( Errors[j;].code; == rc )
                break;
        sprintf( Misc, "%s returned '%s'.", pMsg, j < NUM_ERRORS
            ? Errors[j;].pMsg;
            : "***INVALID RETURN CODE***" );
        AfxMessageBox( Misc );
    }
}

void CMainDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
    CDialog::OnSysCommand(nID, lParam);
}

void CMainDlg::OnPaint()
{
    if (IsIconic())
    {
        CPaintDC dc(this); // device context for painting

        SendMessage(WM_ICONERASEBKGND, (LPARAM) dc.GetSafeHdc(), 0);

        // Center icon in client rectangle
        int cxIcon = GetSystemMetrics(SM_CXICON);
        int cyIcon = GetSystemMetrics(SM_CYICON);
        CRect rect;
        GetClientRect(&rect);
        int x = (rect.Width() - cxIcon + 1) / 2;
        int y = (rect.Height() - cyIcon + 1) / 2;

        // Draw the icon
        dc.DrawIcon(x, y, m_hIcon);
    }
    else
    {
        CDialog::OnPaint();
    }
}

```

```

    }
}

// The system calls this to obtain the cursor to display while the user drags
// the minimized window.
HCURSOR CMainDlg::OnQueryDragIcon()
{
    return (HCURSOR) m_hIcon;
}

void CMainDlg::OnPrint()
{
    CArsOle * pArsCtrl;
    CListBox * pDocsList;
    short      rc;
    char        printer[100];

    pArsCtrl = (CArsOle*)GetDlgItem( IDC_ARSCtrl );
    pDocsList = (CListBox*)GetDlgItem( IDC_DOCLIST );

    GetDlgItem( IDC_PRINTERS )->GetWindowText( printer, sizeof(printer) );

    rc = pArsCtrl->PrintDoc (-1,      // the open doc
                           0,        // entire document
                           printer,
                           1,         // local printer (not server)
                           1,         // # of copies
                           ARS_OLE_ORIENTATION_PORTRAIT,
                           .5, .5, .5, .5, // margins
                           0);        // margins in inches
    if ( rc != ARS_OLE_RC_SUCCESS )
    {
        DisplayMsg( rc, "PrintDoc" );
        return;
    }
}

void CMainDlg::OnDbclckDoclist()
{
    CArsOle * pArsCtrl;
    CListBox * pDocsList;
    short      index;
    short      rc;
    char        printer[100];

    pArsCtrl = (CArsOle*)GetDlgItem( IDC_ARSCtrl );
    pDocsList = (CListBox*)GetDlgItem( IDC_DOCLIST );

    if( m_DocOpened )
        pArsCtrl->CloseDoc();

    index = pDocsList->GetCurSel();

    rc = pArsCtrl->OpenDoc( index, NULL, 0 );
    if ( rc != ARS_OLE_RC_SUCCESS )
    {
        DisplayMsg( rc, "OpenDoc" );
        return;
    }

    m_DocOpened = TRUE;

    GetDlgItem( IDC_PRINTERS )->GetWindowText( printer, sizeof(printer) );
    if( printer != NULL && printer[0]; != '\0' )
        ( CButton*)GetDlgItem( IDC_PRINT )->EnableWindow( TRUE );

    ( CButton*)GetDlgItem( IDC_ATTRIBUTES )->EnableWindow( TRUE );
}

```

```

}

void CMainDlg::OnHScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)
{
    VARIANT        var;
    CArsOle         * pArsCtrl;
    short           rc;

    pArsCtrl = (CArsOle*)GetDlgItem( IDC_ARSCROLL );

    var.vt = VT_I2;
    var.iVal = nPos;
    rc = pArsCtrl->ScrollDocHorz( (short)nSBCode, &var; );
    if ( rc != ARS_OLE_RC_SUCCESS )
        DisplayMsg( rc, "ScrollDocHorz" );
    else
        pScrollBar->SetScrollPos( var.iVal );
}

void CMainDlg::OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)
{
    VARIANT        var;
    CArsOle         * pArsCtrl;
    short           rc;

    pArsCtrl = (CArsOle*)GetDlgItem( IDC_ARSCROLL );

    var.vt = VT_I2;
    var.iVal = nPos;
    rc = pArsCtrl->ScrollDocVert( (short)nSBCode, &var; );
    if ( rc != ARS_OLE_RC_SUCCESS )
        DisplayMsg( rc, "ScrollDocVert" );
    else
        pScrollBar->SetScrollPos( var.iVal );
}

void CMainDlg::RefreshDocList( )
{
    VARIANT        var;
    CArsOle         * pArsCtrl;
    ArsOleValue * pValues;
    CListBox * pDocsList;
    char            temp[21];
    short           rc;
    long            num_docs = 0, j;

    pArsCtrl = (CArsOle*)GetDlgItem( IDC_ARSCROLL );
    pDocsList = (CListBox*)GetDlgItem( IDC_DOCLIST );

    if ( pArsCtrl == NULL || pDocsList == NULL )
        return;

    rc = pArsCtrl->GetNumDocsInList( &var; );
    if ( rc != ARS_OLE_RC_SUCCESS )
    {
        DisplayMsg( rc, "GetNumDocsInList" );
        return;
    }
    num_docs = var.lVal;

    pValues = new ArsOleValue[ max( m_NumDisplayFields, 1 ) ];
    pDocsList->ResetContent( );

    for ( j = 0; j < num_docs; j++ )
    {
        rc = pArsCtrl->GetDocDisplayValues( j, (IUnknown*)pValues, m_NumDisplayFields );
    }
}

```

```

        if ( rc != ARS_OLE_RC_SUCCESS )
        {
            DisplayMsg( rc, "GetDocDisplayValues" );
            break;
        }
        sprintf( temp, "%s\\t%s\\t%s", pValues[0;], pValues[3;], pValues[2;] );
        pDocsList->InsertString( -1, temp );
    }
    pDocsList->SetCurSel( 0 );

    delete [;]; pValues;
}

void CMainDlg::OnFolderSearchCompletedArsctrl()
{
    RefreshDocList( );
}

void CMainDlg::OnDocOpenedArsctrl()
{
    VARIANT        var;
    BOOL            required;
    CScrollBar * pHorz, * pVert;
    CArsOle         * pArsCtrl;
    short           rc;

    pArsCtrl = (CArsOle*)GetDlgItem( IDC_ARSCtrl );

    rc = pArsCtrl->GetDocNumPages( &var );
    if ( rc != ARS_OLE_RC_SUCCESS )
    {
        DisplayMsg( rc, "GetDocNumPages" );
        return;
    }
    m_NumPages = var.lVal;

    rc = pArsCtrl->IsDocHorzScrollRequired( &var );
    if ( rc != ARS_OLE_RC_SUCCESS )
    {
        DisplayMsg( rc, "IsDocHorzScrollRequired" );
        return;
    }
    required = var.iVal;

    m_CurrentPage = 1;

    pHorz = (CScrollBar*)GetDlgItem( IDC_HORZ_SCROLLBAR );
    pVert = (CScrollBar*)GetDlgItem( IDC_VERT_SCROLLBAR );

    pHorz->ShowScrollBar( required );
    pVert->ShowScrollBar( TRUE );
    pHorz->SetScrollPos( 0 );
    pVert->SetScrollPos( 0 );
}

void CMainDlg::OnDocClosedArsctrl()
{
    CScrollBar * pBar;

    pBar = (CScrollBar*)GetDlgItem( IDC_HORZ_SCROLLBAR );
    if ( pBar != NULL )
        pBar->ShowScrollBar( FALSE );

    pBar = (CScrollBar*)GetDlgItem( IDC_VERT_SCROLLBAR );
    if ( pBar != NULL )
        pBar->ShowScrollBar( FALSE );
}

```

```

void CMainDlg::OnCloseDlg()
{
    short    rc;
    CArsOle * pArsCtrl;

    pArsCtrl = (CArsOle*)GetDlgItem( IDC_ARSCtrl );

    if( m_DocOpened )
        pArsCtrl->CloseDoc();

    rc = pArsCtrl->CloseFolder( );
    if ( rc != ARS_OLE_RC_SUCCESS )
        DisplayMsg( rc, "CloseFolder" );

    rc = pArsCtrl->Logoff( );
    if ( rc != ARS_OLE_RC_SUCCESS )
        DisplayMsg( rc, "Logoff" );

    EndDialog(0);
}

void CMainDlg::OnSetDocAttrs()
{
    CAttrsDlg dlg;
    dlg.DoModal( );
}

////////////////////////////////////
// CAttrsDlg dialog

CAttrsDlg::CAttrsDlg(CWnd* pParent /*=NULL*/)
: CDialog(CAttrsDlg::IDD, pParent)
{
   //{{AFX_DATA_INIT(CAttrsDlg)
    // NOTE: the ClassWizard will add member initialization here
   //}}AFX_DATA_INIT
}

void CAttrsDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
   //{{AFX_DATA_MAP(CAttrsDlg)
    // NOTE: the ClassWizard will add DDX and DDV calls here
   //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAttrsDlg, CDialog)
   //{{AFX_MSG_MAP(CAttrsDlg)
    ON_BN_CLICKED(IDC_BACK_COLOR, OnBackColor)
    ON_BN_CLICKED(IDC_IMAGE_COLOR, OnImageColor)
    ON_BN_CLICKED(IDC_ROTATION, OnRotation)
    ON_BN_CLICKED(IDC_ZOOM, OnZoom)
   //}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////
// CAttrsDlg message handlers

BOOL CAttrsDlg::OnInitDialog()
{
    CArsOle * pArsCtrl;
    CListBox * pBackList, * pImageList;

```

```

CEdit      * pZoom, * pRotation;
VARIANT    var1, var2, var3;
BOOL       chg;
short      rc, j, back_color, image_color, zoom, rotation, min, max;
int        index;
char       data[10];

pArsCtrl = (CArsOLE*)pMainDlg->GetDlgItem( IDC_ARSCtrl );

pBackList = (CListBox*)GetDlgItem( IDC_BACK_COLORS );
pImageList = (CListBox*)GetDlgItem( IDC_IMAGE_COLORS );
pZoom = (CEdit*)GetDlgItem( IDC_ZOOMPCT );
pRotation = (CEdit*)GetDlgItem( IDC_ROT );

rc = pArsCtrl->GetDocBackgroundColor( &var1, &var2 );
if ( rc != ARS_OLE_RC_SUCCESS )
{
    pMainDlg->DisplayMsg( rc, "GetDocBackgroundColor" );
    EndDialog( IDABORT );
    return TRUE;
}
back_color = var1.iVal;
chg = var2.iVal;

rc = pArsCtrl->GetDocImageColor( &var1, &var2 );
if ( rc != ARS_OLE_RC_SUCCESS )
{
    pMainDlg->DisplayMsg( rc, "GetDocImageColor" );
    EndDialog( IDABORT );
    return TRUE;
}
image_color = var1.iVal;
chg = var2.iVal;

for ( j = 0; j < NUM_COLORS; j++ )
{
    index = pBackList->AddString( Colors[j].pText );
    pBackList->SetItemData( index, Colors[j].color );
    if ( Colors[j].color == back_color )
        pBackList->SetCurSel( index );
    index = pImageList->AddString( Colors[j].pText );
    pImageList->SetItemData( index, Colors[j].color );
    if ( Colors[j].color == image_color )
        pImageList->SetCurSel( index );
}

rc = pArsCtrl->GetDocZoom( &var1, &var2, &var3 );
if ( rc != ARS_OLE_RC_SUCCESS )
{
    pMainDlg->DisplayMsg( rc, "GetDocZoom" );
    EndDialog( IDABORT );
    return TRUE;
}
zoom = var1.iVal;
min = var2.iVal;
max = var3.iVal;

sprintf( data, "%d", (int)zoom );
pZoom->SetWindowText( data );

rc = pArsCtrl->GetDocRotation( &var1, &var2 );
if ( rc != ARS_OLE_RC_SUCCESS )
{
    pMainDlg->DisplayMsg( rc, "GetDocRotation" );
    EndDialog( IDABORT );
    return TRUE;
}

```

```

rotation = var1.iVal;
chg = var2.iVal;

sprintf( data, "%d", (int)rotation );
pRotation->SetWindowText( data );

return TRUE;
}

void CAttrsDlg::OnBackColor()
{
    CArsOle * pArsCtrl;
    CListBox * pList;
    VARIANT    var1, var2;
    BOOL       chg;
    short      rc, j, color;

    pArsCtrl = (CArsOle*)pMainDlg->GetDlgItem( IDC_ARSCtrl );
    pList = (CListBox*)GetDlgItem( IDC_BACK_COLORS );

    color = (short)pList->GetItemData( pList->GetCurSel( ) );

    rc = pArsCtrl->SetDocBackgroundColor( color );
    if ( rc != ARS_OLE_RC_SUCCESS )
    {
        pMainDlg->DisplayMsg( rc, "SetDocBackgroundColor" );
        rc = pArsCtrl->GetDocBackgroundColor( &var1, &var2 );
        if ( rc != ARS_OLE_RC_SUCCESS )
            pMainDlg->DisplayMsg( rc, "GetDocBackgroundColor" );
        else
        {
            color = var1.iVal;
            chg = var2.iVal;
            for ( j = 0; j < NUM_COLORS; j++ )
            {
                if ( (short)pList->GetItemData(j) == color )
                {
                    pList->SetCurSel(j);
                    break;
                }
            }
        }
    }
}

void CAttrsDlg::OnImageColor()
{
    CArsOle * pArsCtrl;
    CListBox * pList;
    VARIANT    var1, var2;
    BOOL       chg;
    short      rc, j, color;

    pArsCtrl = (CArsOle*)pMainDlg->GetDlgItem( IDC_ARSCtrl );
    pList = (CListBox*)GetDlgItem( IDC_IMAGE_COLORS );

    color = (short)pList->GetItemData( pList->GetCurSel( ) );

    rc = pArsCtrl->SetDocImageColor( color );
    if ( rc != ARS_OLE_RC_SUCCESS )
    {
        pMainDlg->DisplayMsg( rc, "SetDocImageColor" );
        rc = pArsCtrl->GetDocImageColor( &var1, &var2 );
        if ( rc != ARS_OLE_RC_SUCCESS )
            pMainDlg->DisplayMsg( rc, "GetDocImageColor" );
        else
        {

```

```

        color = var1.iVal;
        chg = var2.iVal;
        for ( j = 0; j < NUM_COLORS; j++ )
        {
            if ( (short)pList->GetItemData(j) == color )
            {
                pList->SetCurSel(j);
                break;
            }
        }
    }
}

void CAttrsDlg::OnRotation()
{
    CArsOle * pArsCtrl;
    CEdit * pRotation;
    VARIANT var1, var2;
    BOOL chg;
    short rc, value;
    char data[10];

    pArsCtrl = (CArsOle*)pMainDlg->GetDlgItem( IDC_ARSCtrl );
    pRotation = (CEdit*)GetDlgItem( IDC_ROT );

    pRotation->GetWindowText( data, sizeof(data) );

    rc = pArsCtrl->SetDocRotation( (short)atoi( data ) );
    if ( rc != ARS_OLE_RC_SUCCESS )
    {
        pMainDlg->DisplayMsg( rc, "SetDocRotation" );
        rc = pArsCtrl->GetDocRotation( &var1, &var2 );
        if ( rc != ARS_OLE_RC_SUCCESS )
            pMainDlg->DisplayMsg( rc, "GetDocRotation" );
        else
        {
            value = var1.iVal;
            chg = var2.iVal;
            sprintf( data, "%d", (int)value );
            pRotation->SetWindowText( data );
        }
    }
    else
        OnZoom();
}

void CAttrsDlg::OnZoom()
{
    CArsOle * pArsCtrl;
    CEdit * pZoom;
    CScrollBar * pHorz, * pVert;
    VARIANT var1, var2, var3;
    BOOL required;
    short rc, value, min, max, horz_pos, vert_pos;
    char data[10];

    pArsCtrl = (CArsOle*)pMainDlg->GetDlgItem( IDC_ARSCtrl );
    pZoom = (CEdit*)GetDlgItem( IDC_ZOOMPCT );

    pZoom->GetWindowText( data, sizeof(data) );

    rc = pArsCtrl->SetDocZoom( (short)atoi( data ), &var1, &var2 );
    horz_pos = var1.iVal;
    vert_pos = var2.iVal;
    if ( rc != ARS_OLE_RC_SUCCESS )
    {

```

```

pMainDlg->DisplayMsg( rc, "SetDocZoom" );
rc = pArsCtrl->GetDocZoom( &var1;, &var2;, &var3; );
if ( rc != ARS_OLE_RC_SUCCESS )
    pMainDlg->DisplayMsg( rc, "GetDocZoom" );
else
{
    value = var1.iVal;
    min = var2.iVal;
    max = var3.iVal;
    sprintf( data, "%d", (int)value );
    pZoom->SetWindowText( data );
}
}

else
{
    rc = pArsCtrl->IsDocHorzScrollRequired( &var1; );
    if ( rc != ARS_OLE_RC_SUCCESS )
    {
        pMainDlg->DisplayMsg( rc, "IsDocHorzScrollRequired" );
        return;
    }
    required = var1.iVal;

    pHorz = (CScrollbar*)pMainDlg->GetDlgItem( IDC_HORZ_SCROLLBAR );
    pVert = (CScrollbar*)pMainDlg->GetDlgItem( IDC_VERT_SCROLLBAR );

    pHorz->ShowScrollBar( required );
    pHorz->SetScrollPos( horz_pos );
    pVert->SetScrollPos( vert_pos );
}
}

```

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample

programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" www.ibm.com/legal/copytrade.shtml.

Adobe, Acrobat, Portable Document Format (PDF), and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Portions of the OnDemand Windows client program contain licensed software from Pixel Translations Incorporated, © Pixel Translations Incorporated 1990, 2003. All rights reserved.



All rights reserved.

FairCom Corp. and other company, product or service names might be trademarks or service marks of others.

Index

A

about this publication vii
ACTIVATE_DOC command 135
ACTIVATE_FOLDER command 135
ANNOTATE_DOC command 136
ARRANGE_DOCS command 137

C

Change Password parameter 123
CHANGE_PASSWORD command 138
CID 215
Citrix server install 205
CLEAR_FIELDS command 138
client
 distributing user-defined files 211
 installing on a network 205
 integrating with Monarch 195
 network installation 205
 user-defined files 211
CLOSE_ALL_DOCS command 139
CLOSE_ALL_FOLDERS command 139
CLOSE_DOC command 140
CLOSE_FOLDER command 140
Code Page Override /A 123
command line reference
 parameter syntax 123
 parameters 126
 Change Password /C 123
 Disable Anticipation /V 124
 Disable Close Folder /Z 124
 Disable Exit — /K 124
 Disable Logoff and Password
 Change /X 124
 Disable Update Servers /Y 124
 Disable User Confirmation
 /B 124
 Enable DDE Interface /I 125
 Folder Name /F 125
 Free Memory when Folder Closed
 /Q 125
 Language Path /L 125
 Logon Password /P 126
 Logon Server Name /S 126
 Logon User ID /U 126
 Maximum Open Folders /O 126
 Product Title /T 126
 Window Placement /W 127
 Starting Content Manager OnDemand
 32-bit client 123
configuration, installation, and
 distribution 215
Content Manager OnDemand
 customization overview 121
Content Manager OnDemand DDE
 commands
 ACTIVATE_DOC 135
 ACTIVATE_FOLDER 135
 ANNOTATE_DOC 136
 ARRANGE_DOCS 137

Content Manager OnDemand DDE
 commands (*continued*)
 CHANGE_PASSWORD 138
 CLEAR_FIELDS 138
 CLOSE_ALL_DOCS 139
 CLOSE_ALL_FOLDERS 139
 CLOSE_DOC 140
 CLOSE_FOLDER 140
 COPY_DOC_PAGES 141
 DELETE_DOC 142
 DESELECT_DOC 143
 DISABLE_SWITCH 143
 ENABLE_SWITCH 144
 EXIT 145
 GET_DISPLAY_FIELDS 145
 GET_DOC_VALUES 146
 GET_FOLDER_FIELDS 147
 GET_FOLDERS 148
 GET_NUM_DOC_PAGES 149
 GET_NUM_DOCS_IN_LIST 148
 GET_PRINTERS 149
 GET_QUERY_FIELDS 150
 GET_SERVERS 151
 LOGOFF 151
 LOGON 152
 OPEN_DOC 153
 OPEN_FOLDER 154
 PRINT_DOC 157
 RESTORE_DEFAULTS 159
 RETRIEVE_DOC 159
 return codes 171
 SEARCH_FOLDER 160
 SELECT_DOC 161
 SET_FIELD_DATA 162
 SET_FOCUS 163
 SET_HELP_PATH 164
 SET_USER_MSG_MODE 165
 SHOW_WINDOW 165
 STORE_DOC 166
 UPDATE_DOC 168
COPY_DOC_PAGES command 141
copying software to a file server 206
custom install 205
customization overview 121

D

data mining
 with Monarch 195
DDE commands
 ACTIVATE_DOC 135
 ACTIVATE_FOLDER 135
 ANNOTATE_DOC 136
 ARRANGE_DOCS 137
 CHANGE_PASSWORD 138
 CLEAR_FIELDS 138
 CLOSE_ALL_DOCS 139
 CLOSE_ALL_FOLDERS 139
 CLOSE_DOC 140
 CLOSE_FOLDER 140
 COPY_DOC_PAGES 141

DDE commands (*continued*)

 DELETE_DOC 142
 DESELECT_DOC 143
 DISABLE_SWITCH 143
 ENABLE_SWITCH 144
 EXIT 145
 GET_DISPLAY_FIELDS 145
 GET_DOC_VALUES 146
 GET_FOLDER_FIELDS 147
 GET_FOLDERS 148
 GET_NUM_DOC_PAGES 149
 GET_NUM_DOCS_IN_LIST 148
 GET_PRINTERS 149
 GET_QUERY_FIELDS 150
 GET_SERVERS 151
 LOGOFF 151
 LOGON 152
 OPEN_DOC 153
 OPEN_FOLDER 154
 PRINT_DOC 157
 RESTORE_DEFAULTS 159
 RETRIEVE_DOC 159
 return codes 171
 SEARCH_FOLDER 160
 SELECT_DOC 161
 SET_FIELD_DATA 162
 SET_FOCUS 163
 SET_HELP_PATH 164
 SET_USER_MSG_MODE 165
 SHOW_WINDOW 165
 STORE_DOC 166
 UPDATE_DOC 168
DDEML
 termination 132
 transactions 132
DELETE_DOC command 142
DESELECT_DOC command 143
directories 205
Disable Anticipation parameter 124
Disable Close Folder parameter 124
Disable Exit parameter 124
Disable Logoff and Password Change
 parameter 124
Disable Update Servers parameter 124
Disable User Confirmation
 parameter 124
DISABLE_SWITCH command 143
distributing user-defined files 211
distribution install 205, 206
distribution server 195, 205
Dynamic Data Exchange interface
 reference 129

E

Enable DDE Interface parameter 125
ENABLE_SWITCH command 144
EXIT command 145

F

file server
 copying OnDemand software to 206
Folder Name parameter 125
Free Memory when Folder Closed
 parameter 125

G

GET_DISPLAY_FIELDS command 145
GET_DOC_VALUES command 146
GET_FOLDER_FIELDS command 147
GET_FOLDERS command 148
GET_NUM_DOC_PAGES command 149
GET_NUM_DOCS_IN_LIST
 command 148
GET_PRINTERS command 149
GET_QUERY_FIELDS command 150
GET_SERVERS command 151

H

hints and tips 221

I

ICU converters
 converters 219
install
 Citrix server 205
 custom 205
 distribution 205, 206
 local 205
 standard 205
 typical 205
 user-defined files 207
installation
 creating and using response files 215
 response files 215
installation directories 205
installing
 clients on a network 205
 Monarch 195
 network file server 205
 user-defined files 211
integration with Monarch 195
interface reference for DDE 129

L

Language Path parameter 125
local install 205
LOGOFF command 151
LOGON command 152
Logon Password parameter 126
Logon Server Name parameter 126
Logon User ID parameter 126

M

Maximum Open Folders parameter 126
methods
 AboutBox 5
 ActivateFolder 5

methods (*continued*)

AnnotateDoc 6
CancelOperation 7
ChangePassword 8
ClearFolderSearchFields 9
CloseAllFolders 10
CloseDoc 11
CloseFolder 11
CopyBitmap 12
CopyDocPagesToFile 13
CopyText 14
DeleteDoc 15
FindStringInDoc 15
GetAnnotationForDoc 18
GetAnnotationStatus 19
GetControlId 20
GetDocAnnotation 21
GetDocBackgroundColor 23
GetDocCurrentPage 24
GetDocDisplayValue 25
GetDocDisplayValues 26
GetDocImageColor 28
GetDocImageIntensity 29
GetDocNumPages 30
GetDocRotation 31
GetDocScrollPositions 33
GetDocType 34
GetDocZoom 35
GetFolderDisplayFieldName 36
GetFolderDisplayFieldNames 37
GetFolderFieldName 39
GetFolderFieldNames 40
GetFolderName 41
GetFolderNames 42
GetFolderSearchFieldName 44
GetFolderSearchFieldNames 45
GetNumDocAnnotations 47
GetNumDocsInList 48
GetNumFolderDisplayFields 50
GetNumFolderFields 52
GetNumFolders 54
GetNumFolderSearchFields 55
GetNumServerPrinters 58
GetNumServers 59
GetResourceCacheMode 60
GetServerName 61
GetServerNames 62
GetServerPrinter 64
GetServerPrinterInfo 65
GetStoreDocInvalidFieldNum 66
GetTypeForDoc 67
IsDocHorzScrollRequired 68
Logoff 71
Logon 72
OnSysColorChange 73
OpenDoc 74
OpenFolder 77
PrintDoc 78
RetrieveDoc 80
ScrollDocHorz 82
ScrollDocVert 85
SearchFolder 88
SetDefaultFolderSearchFields 90
SetDocBackgroundColor 91
SetDocCurrentPage 92
SetDocImageColor 93
SetDocImageIntensity 94

methods (*continued*)

SetDocRotation 95
SetDocZoom 95
SetFolderCloseMemoryRelease 97
SetFolderSearchFieldData 98
SetLogonReturnOnFailure 100
SetResourceCacheMode 101
SetRightButtonMenu 102
SetSelectionMode 105
SetServerPrinterData 106
SetUserMessageMode 107
ShowFolder 108
ShowWaitCursorDuringCancelable
 Operation 109
StoreDoc 110
UndoFind 113
UpdateDoc 114
WasOperationCancelled 115

Monarch
 integration with 195

N

network file server 205
 copying OnDemand software to 206
network installation 205

O

OLE events
 AreaDeselected 117
 AreaSelected 117
 DocClosed 117
 DocOpened 117
 FolderClosed 117
 FolderSearchCompleted 117
 UserCommand(long CommandID
) 118
OPEN_DOC command 153
OPEN_FOLDER command 154

P

parameter syntax 123
PRINT_DOC command 157
Product Title parameter 126

R

reference workstation 195
report mining
 with Monarch 195
response files 215
RESTORE_DEFAULTS command 159
RETRIEVE_DOC command 159
return codes 171

S

SEARCH_FOLDER command 160
SELECT_DOC command 161
server
 copying OnDemand software to 206
SET_FIELD_DATA command 162
SET_FOCUS command 163

- SET_HELP_PATH command 164
- SET_USER_MSG_MODE command 165
- SHOW_WINDOW command 165
- standard install 205
- Starting Content Manager OnDemand
 - 32-bit 123
- STORE_DOC command 166
- syntax rules for command line
 - parameters 123

T

- terminating the DDE conversation 132
- transactions
 - DDEML 132
 - example 132
- troubleshooting 221
- troubleshooting scenarios
 - error code 2 from the StoreDoc()
 - API 221
- typical install 205

U

- UPDATE_DOC command 168
- user-defined files 207, 211

W

- Window Placement parameter 127
- Windows client
 - CID 215
 - configuration, installation, and distribution 215
 - creating and using response files 215
 - distributing user-defined files 211
 - installing on a network 205
 - integrating with Monarch 195
 - network installation 205
 - response files 215
 - user-defined files 211



Product Number: 5697-N93
5724-J33
5770-RD1

SC19-3357-00

