

IBM Content Manager OnDemand
Version 10 Release 1

Windows Client Customization Guide



Note

Before using this information and the product that it supports, read the information in [“Notices” on page 279](#).

This edition applies to the following products and to all subsequent releases and modifications until otherwise indicated in new editions:

- Version 10 Release 1 of IBM® Content Manager OnDemand for Multiplatforms (product number 5724-J33), IBM Content Manager OnDemand for z/OS (product number 5724-J33) and Version 7 Release 3 of IBM Content Manager OnDemand for i (product number 5770-RD1)

© **Copyright 2017 - 2018 All Rights Reserved. UNICOM Systems, Inc. – a division of UNICOM Global.**

© **Copyright International Business Machines Corporation 1996, 2018.**

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

Contents

ibm.com® and related resources.....	ix
Contacting IBM.....	ix
Accessibility information for Content Manager OnDemand.....	x
Chapter 1. Overview of the Content Manager OnDemand OLE Control.....	1
View multiple documents for a single folder.....	1
Header file.....	1
Return Code.....	2
Chapter 2. Methods.....	3
AboutBox method.....	3
ActivateFolder method.....	3
AnnotateDoc method.....	4
CancelOperation method.....	5
ChangePassword method.....	6
ClearFolderSearchFields method.....	7
CloseAllFolders method.....	8
CloseDoc method.....	9
CloseFolder method.....	10
CopyBitmap method.....	10
CopyText method.....	11
DeleteDoc method.....	12
FindStringInDoc method.....	13
GetAnnotationForDoc method.....	15
GetAnnotationStatus method.....	16
GetControlId method.....	17
GetDocAnnotation method.....	18
GetDocBackgroundColor method.....	20
GetDocCurrentPage method.....	21
GetDocDisplayValue method.....	22
GetDocDisplayValues methods.....	23
GetDocImageColor method.....	25
GetDocImageIntensity methods.....	26
GetDocNumPages method.....	28
GetDocRotation method.....	29
GetDocScrollPositions method.....	30
GetDocType method.....	31
GetDocZoom method.....	32
GetFolderDisplayFieldName method.....	33
GetFolderDisplayFieldNames method.....	34
GetFolderFieldName method.....	36
GetFolderFieldNames method.....	37
GetFolderName method.....	38
GetFolderNames method.....	39
GetFolderSearchFieldName method.....	40
GetFolderSearchFieldNames method.....	42
GetNumDocAnnotations method.....	43
GetNumDocsInList method.....	45
GetNumFolderDisplayFields method.....	47
GetNumFolderFields method.....	49
GetNumFolders method.....	51

GetNumFolderSearchFields method.....	53
GetNumServerPrinters method.....	55
GetNumServers method.....	56
GetResourceCacheMode method.....	57
GetServerName method.....	58
GetServerNames method.....	59
GetServerPrinter method.....	61
GetServerPrinterInfo method.....	62
GetStoreDocInvalidFieldNum method.....	63
GetTypeForDoc method.....	64
IsDocHorzScrollRequired method.....	65
Logoff method.....	68
Logon method.....	69
OnSysColorChange method.....	70
OpenDoc method.....	71
OpenFolder method.....	74
PrintDoc method.....	75
RetrieveDoc method.....	77
ScrollDocHorz method.....	79
ScrollDocVert method.....	82
SearchFolder method.....	85
SetDefaultFolderSearchFields method.....	87
SetDocBackgroundColor method.....	88
SetDocCurrentPage method.....	89
SetDocImageColor method.....	90
SetDocImageIntensity method.....	91
SetDocRotation method.....	92
SetDocZoom method.....	93
SetFolderCloseMemoryRelease method.....	95
SetFolderSearchFieldData method.....	95
SetLogonReturnOnFailure method.....	98
SetResourceCacheMode method.....	99
SetRightButtonMenu method.....	100
SetSelectionMode method.....	102
SetServerPrinterData method.....	103
SetUserMessageMode method.....	105
ShowFolder method.....	106
ShowWaitCursorDuringCancelableOperation method.....	107
StoreDoc method.....	108
UndoFind method.....	110
UpdateDoc method.....	111
WasOperationCancelled method.....	112
Chapter 3. OLE Events.....	115
FolderSearchCompleted method.....	115
FolderClosed method.....	115
DocOpened method.....	115
DocClosed method.....	115
AreaSelected method.....	116
AreaDeselected method.....	116
UserCommand(long CommandID).....	116
Chapter 4. Content Manager OnDemand customization overview.....	117
Chapter 5. Command line.....	119
Starting the Content Manager OnDemand client.....	119
Parameter syntax.....	119

Parameters.....	119
Change Password —/C new password.....	119
Code Page Override — /A codepage_number.....	119
Disable Anticipation —/V.....	120
Disable Close Folder —/Z.....	120
Disable Exit — /K.....	120
Disable Logoff or Password Change —/X.....	120
Disable Update Servers —/Y.....	120
Disable User Confirmation —/B.....	120
Enable DDE Interface — /I number,path,resid.....	121
Folder Name —/F name.....	121
Free Memory When Folder Closed —/Q.....	121
Language ID Override — /L.....	121
Logon Password —/P password.....	121
Logon Server Name —/S name.....	122
Logon User ID —/U id.....	122
Maximum Open Folders —/O number.....	122
Product Title —/T name.....	122
Window Placement —/W placement.....	122
CD-ROM Mastering.....	123

Chapter 6. Dynamic Data Exchange (DDE) and DDE Management Library..... 125

Invoking the Content Manager OnDemand client from another Windows application.....	125
Content Manager OnDemand invocation and DDEML initialization.....	126
DDEML termination.....	127
DDEML transactions.....	128

Chapter 7. Content Manager OnDemand DDE commands..... 131

ACTIVATE_DOC command.....	131
ACTIVATE_FOLDER command.....	132
ANNOTATE_DOC command.....	132
ARRANGE_DOCS command.....	133
CHANGE_PASSWORD command.....	134
CLEAR_FIELDS command.....	135
CLOSE_ALL_DOCS commands.....	136
CLOSE_ALL_FOLDERS command.....	136
CLOSE_DOC command.....	137
CLOSE_FOLDER command.....	137
COPY_DOC_PAGES command.....	138
DELETE_DOC command.....	139
DESELECT_DOC command.....	140
DISABLE_SWITCH command.....	141
ENABLE_SWITCH command.....	141
EXIT command.....	142
GET_DISPLAY_FIELDS command.....	143
GET_DOC_VALUES command.....	143
GET_FOLDER_FIELDS command.....	144
GET_FOLDERS command.....	145
GET_NUM_DOCS_IN_LIST command.....	146
GET_NUM_DOC_PAGES command.....	146
GET_PRINTERS command.....	147
GET_QUERY_FIELDS commands.....	148
GET_SERVERS commands.....	148
LOGOFF command.....	149
LOGON command.....	150
OPEN_DOC command.....	151
OPEN_FOLDER command.....	152

PRINT_DOC command.....	154
RESTORE_DEFAULTS commands.....	156
RETRIEVE_DOC command.....	157
SEARCH_FOLDER command.....	158
SELECT_DOC command.....	158
SET_FIELD_DATA command.....	159
SET_FOCUS command.....	161
SET_HELP_PATH command.....	161
SET_USER_MSG_MODE command.....	162
SHOW_WINDOW command.....	163
STORE_DOC command.....	164
UPDATE_DOC command.....	166
Chapter 8. Return Codes.....	169
Chapter 9. DDEML advise loop.....	171
Chapter 10. External applications and dynamic link libraries.....	173
Chapter 11. Related documents.....	181
Chapter 12. Program Information File.....	187
Chapter 13. Document Audit Facility.....	189
Create the DAF control file.....	189
AUDIT section.....	189
Folder section.....	190
Define the report.....	190
Defining the application group.....	190
Defining applications.....	191
Defining Folders.....	191
Control access to the DAF.....	191
Using the DAF.....	192
Chapter 14. Modifying client behavior through the Registry.....	193
Chapter 15. Integration with Monarch.....	195
Configuring the client.....	195
Adding the Registry key.....	196
Exporting the Registry key.....	199
Using multiple Monarch model files.....	199
Configuring Setup.....	200
Copying client software.....	201
Adding subdirectories.....	201
Copying Monarch files.....	201
Running Setup.....	202
Running Monarch from Content Manager OnDemand.....	202
Chapter 16. Installing client software on a network.....	203
Sharing Content Manager OnDemand clients among multiple users.....	203
Installation directories.....	203
Chapter 17. Distributing user-defined files.....	205
Copy Content Manager OnDemand client software to the server.....	205
Configuring the server to install user-defined files.....	205
Storing user-defined files on the server.....	206

Installing the Content Manager OnDemand client.....	206
Chapter 18. Using response files.....	207
Format of a response file.....	207
Creating a response file.....	207
Installing software using a response file.....	207
Verifying software installation.....	208
Using a response file to install Content Manager OnDemand software.....	208
Chapter 19. IsDocHorzScrollRequired method.....	209
Chapter 20. Mapping AFP Fonts.....	213
When you need to map fonts.....	213
Viewer files supplied for mapping fonts.....	214
Steps for mapping fonts to the Viewer.....	214
Syntax rules for the Viewer font definition files.....	215
Coded Font file.....	215
Character Set Definition file.....	216
Code Page Definition file.....	218
Alias file.....	219
Support for TrueType Fonts.....	221
Chapter 21. Full report browse and reprint.....	223
Chapter 22. ICU converters.....	225
Chapter 23. Troubleshooting.....	227
StoreDoc() API returns error code 2.....	227
How to allow a user to print AFP™ files without opening them with the workbench or navigation plug-in.....	227
Appendix A. Microsoft™ Visual Basic DDE sample program.....	229
Global variables used by the sample program.....	229
Entry point for the sample program.....	230
Appendix B. Microsoft™ Visual C++ DDE sample program.....	245
Global variables used by the sample program.....	245
Appendix C. Microsoft™ Visual Basic OLE sample program.....	255
Global variables used by the sample program.....	255
Appendix D. Microsoft™ Visual C++ OLE sample program.....	265
Notices.....	279
Trademarks.....	281
Privacy policy considerations	283
Index.....	285

ibm.com[®] and related resources

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

Support and assistance

From ibm.com, click **Support & downloads** and select the type of support that you need. From the Support Portal, you can search for product information, download fixes, open service requests, and access other tools and resources.

IBM Knowledge Center

See your online product information in IBM Knowledge Center at https://www.ibm.com/support/knowledgecenter/SSEPCD_10.1.0/com.ibm.ondemandtoc.doc/ondemandmp_10.1.0.htm.

PDF publications

See the following PDF publications for your product at <http://www.ibm.com/support/docview.wss?uid=swg27050851>.

Contacting IBM

For general inquiries, call 800-IBM-4YOU (800-426-4968). To contact IBM customer service in the United States or Canada, call 1-800-IBM-SERV (1-800-426-7378).

For more information about how to contact IBM, including TTY service, see the Contact IBM website at <http://www.ibm.com/contact/us/>.

Accessibility information for Content Manager OnDemand

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

Chapter 1. Overview of the Content Manager OnDemand OLE Control

IBM 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.

Note: OLE Controls are supported by the 32-bit OnDemand client. OLE Controls are not supported by the 64-bit OnDemand client.

View 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` header file.

Chapter 2. Methods

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

AboutBox method

The AboutBox method is used to display the About Box.

Method

```
void AboutBox ( )
```

Description

The Content Manager OnDemand About Box is displayed.

Return Value

None

C/C ++

The following example displays the Content Manager OnDemand About Box.

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

Visual Basic

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

ActivateFolder method

The ActivateFolder method is used to create active folders.

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 codes.

Also see

OpenFolder, CloseFolder, and CloseAllFolders methods

C/C ++

```
CArsOle * pArsCtrl;  
short rc;  
  
:  
:  
  
rc = pArsCtrl->ActivateFolder( "Henry's Folder" );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
:  
:
```

Visual Basic

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

AnnotateDoc method

The AnnotateDoc method is used to create annotations.

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 codes.

Also see

OpenDoc method

C/C ++

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

Visual Basic

```
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

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

Method

short CancelOperation ()

Description

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

Return Value

Refer to return codes.

Also see

SearchFolder, OpenDoc, WasOperationCancelled, and ShowWaitCursorDuringCancelableOperation methods

C/C ++

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

Visual Basic

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

ChangePassword method

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

Content Manager OnDemand changes the logon password for the current user.

Return Value

Refer to return codes.

Also see

Logon method

C/C++

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

Visual Basic

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

ClearFolderSearchFields method

This method allows the search fields for the active folder are cleared.

Method

short ClearFolderSearchFields()

Description

The search fields for the active folder are cleared.

Return Value

Refer to return codes.

Also see

OpenFolder, and SearchFolder methods.

C/C++

```
CArsOle * pArsCtrl;  
short rc;  
  
:  
:
```

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

.
.
```

Visual Basic

```
Dim rc As Integer

.
.
.

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

.
.
.
```

CloseAllFolders method

The method allows all open folders to be closed.

Method

```
short CloseAllFolders()
```

Description

All open folders are closed. All open documents associated with the folders are closed.

Return Value

Refer to return codes.

Also see

OpenFolder and CloseFolder methods

C/C ++

```
CArsOle * pArsCtrl;
short rc;

.
.

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

.
.
```

Visual Basic

```
Dim rc As Integer
```

```

.
.
.

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

.
.
.

```

CloseDoc method

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

Method

short CloseDoc()

Description

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

Return Value

Refer to return codes.

Also see

OpenDoc method

C/C ++

```

CArsOle * pArsCtrl;
short rc;

.
.

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

.
.

```

Visual Basic

```

Dim rc As Integer

.
.
.

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

.
.
.

```

CloseFolder method

The active folder is closed.

Method

short CloseFolder()

Description

The method 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 codes.

See Also

OpenFolder and CloseAllFolders methods

C/C ++

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

Visual Basic

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

CopyBitmap method

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

Method

short CopyBitmap()

Description

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

Return Value

Refer to return codes.

See Also

CopyText and SetSelectionMode methods

C/C ++

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

Visual Basic

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

CopyText method

Method

short CopyText()

Description

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

Return Value

Refer to return codes.

See Also

CopyBitmap and SetSelectionMode methods

C/C ++

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

Visual Basic

```
Dim rc As Integer

:

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

:
```

DeleteDoc method

Content Manager OnDemand deletes the specified document from the database.

Method:

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

Parameters

DocIndex

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

Description

Since the document numbers may have changed, information from a previous GetDocDisplayValues method might not be valid.

Return Value:

Refer to return codes.

See Also:

GetNumDocsInList method

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

A search is conducted for the text string beginning on the specified page.

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 nonzero, 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

The variable pointed to by pFound is set to nonzero 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 wraps the document from end to beginning or beginning to end. A previous or next find does not fail. If there is a single occurrence in the document, these will find the same string.

Return Value

Refer to return codes.

See Also

OpenDoc, and UndoFind methods

C/C ++

```
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

```
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 method

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 user ID 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 codes.

See Also:

GetNumDocAnnotation and GetDocAnnotation

```
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 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 If
End For
Next j
```

GetAnnotationStatus method

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 is 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 codes.

See Also:

AnnotateDoc, GetNumDocAnnotations, GetDocAnnotations, GetDocAnnotation, and GetAnnotationForDoc methods

C/C++

```
VARIANT status;
CArsOle * pArsCtrl;
short rc;

.
.
.

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

if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

.
.
.
```

Visual Basic

```
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

The identifier of the control is returned in the specified variable.

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

This control identifier can be used to reference information associated with a different Content Manager OnDemand Ole Control. Refer to GetControlId for a discussion of control IDs.

Return Value

Refer to return codes.

Also see

OpenDoc method

the

C/C++

The following example retrieves the control ID.

```
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

```
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 method

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 user ID 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.

See Also

`GetNumDocAnnotations` and `GetAnnotationForDoc` methods

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

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 nonzero 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 codes.

See Also

SetDocBackgroundColor method

C/C++

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

```
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

```
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

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

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 codes.

See Also

SetDocCurrentPage, GetDocNumPages methods

C/C++

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

```

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

```
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 method

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 codes.

See Also:

`GetNumDocsInList`, `GetNumFolderDisplayFields`, `GetDocDisplayFieldNames`, `GetDocDisplayValues`, and `OpenDoc` methods

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.GetFolderDisplayName(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 methods

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 codes.

See Also

GetNumDocsInList, GetNumFolderDisplayFields, GetDocDisplayValue, and OpenDoc methods

C/C++

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

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

.
.

```

GetDocImageColor method

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

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 nonzero 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 codes.

See Also

SetDocImageColor method

C/C++

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

```
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

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

GetDocImageIntensity methods

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 nonzero 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 codes.

See Also

SetDocImageIntensity method

C/C++

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

```
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

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

·
·

GetDocNumPages method

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 codes.

See Also

OpenDoc, GetDocCurrentPage, and SetDocCurrentPage methods

C/C ++

```
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

```
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

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 nonzero 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 codes.

See Also

SetDocRotation method

C/C++

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

```
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

```
Dim rc As Integer  
Dim rotation, changeable As Variant  
  
:  
:  
  
rc = ArsOle.GetDocRotation (rotation, changeable)
```

```

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

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

:
:

```

GetDocScrollPositions method

The current scroll positions are returned in the specified variables.

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 scroll positions assume that the scroll ranges have been set to ARS_OLE_SCROLL_RANGE.

Return Value

Refer to return codes.

C/C++

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

```

CArsOle * pArsCtrl;
CScrollBar * pHorzScollBar, * 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;

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

```



```
:  
:
```

Visual Basic

```
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 If  
  
rc = ArsOle.GetDocScrollPositions( horz_pos, vert_pos )  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End If  
  
sbHorz.Value = horz_pos  
sbVert.Value = vert_pos  
  
:  
:
```

GetDocType method

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

Method

```
short GetDocType(  
    long Index,  
    VARIANT * pType,  
    LPUNKNOWNpExtension )
```

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 is 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 codes.

See Also

GetTypeForDoc

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

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

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 codes.

See Also

SetDocZoom method

C/C++

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

```
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

```

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 method

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 GetFolderDisplayFieldName.

pName

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

Description

The specified field name is returned in pName.

GetNumFolderDisplayFields or GetFolderDisplayFieldName 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 codes.

See Also

GetNumFolderDisplayFields or GetFolderDisplayFieldName methods

Visual Basic

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.GetFolderDisplayName(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 method

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 GetNumFolderDisplayFields 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 codes for an explanation of the return code.

See Also

GetNumFolderDisplayFields, and 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++

```
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 method

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 codes.

See Also

`GetFolderFieldNames`, `GetNumFolderFields`, or `StoreDoc`

Visual Basic

```
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 If

ReDim FieldNames (num_fields - 1)

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

:
```

GetFolderFieldNames method

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 codes.

See Also

GetFolderFieldName, GetNumFolderFields, and StoreDoc

C/C++

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.

```
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 method

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 codes.

See Also:

GetNumFolders, GetFolderNames, and OpenFolder

Visual Basic

```
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 method

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 codes.

See Also

`GetNumFolders`, `GetFolderName`, and `OpenFolder` methods

C/C++

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.

```
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 method

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 codes.

See Also

`GetNumFolderSearchFields`, `GetFolderSearchFieldNames`, `SetFolderSearchFieldData`, and `SearchFolder`

Visual Basic

```
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 method

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 codes.

See Also

`GetNumFolderSearchFields`, `GetFolderSearchFieldName`, `SetFolderSearchFieldData`, and `SearchFolder` methods

C/C++

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 ArsOle * 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

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

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 codes.

See Also

GetNumDocAnnotation and GetAnnotationForDoc methods

C/C++

The following example retrieves the annotations 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;
```

Visual Basic

```
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 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 If
End If
```

' Process Annotation

Next j

GetNumDocsInList method

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

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 codes.

See Also

GetDocDisplayValues and OpenDoc methods

C/C ++

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

```
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->GetNumDocInList( &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

```

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

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

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 codes.

See Also

GetNumFolderDisplayNames

C/C ++

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

```

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->GetNumFolderDisplayFields( (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

```

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

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

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 codes.

See Also

GetFolderFieldName and GetFolderFieldNames

C/C++

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.

```
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

```
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

The number of folders available for the current server is returned in the specified variable.

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

This value can be used with the GetFolderNames method to prepare for opening a folder.

Return Value

Refer to return codes.

See Also

GetFolderNames and OpenFolder methods

C/C ++

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.

```
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

```

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

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

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

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 codes.

See Also

GetFolderSearchFieldNames, SetFolderSearchFieldData, and SearchFolder

C/C++

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.

```
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->GetFolderSearchFileNames( (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

```

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

The number of server printers available is returned in the specified variable.

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

This value can be used with the GetServerPrinter and GetServerPrinterInfo methods.

Return Value

Refer to return codes.

See Also

GetServerPrinter and GetServerPrinterInfo methods

C/C++

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
}

```

```
.  
. .  
.
```

Visual Basic

```
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

The number of servers available for logon is returned in the specified variable.

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

This value can be used with the GetServerNames method to prepare for a logon.

Return Value

Refer to return codes.

See Also

GetServerNames and Logon methods

C/C ++

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

```
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

```

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

The requested mode is set.

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 are not 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 is required.

Description

The requested mode is set.

Return Value

Refer to return codes.

See Also

RetrieveDoc method

C/C++

The following example sets the resource cache mode to retain.

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

Visual Basic

```
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 method

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 codes.

See Also

`GetNumServers`, `GetServerNames`, and `Logon`

Visual Basic

```
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 method

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 codes.

See Also

GetNumServers, GetServerName, and Logon methods

C/C++

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

```
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 method

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 codes.

See Also

`GetNumServerPrinters`, and `GetServerPrinterInfo` methods

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

```
CArsole * pArsoCtrl;  
  
short rc, j, num_prts;  
char name[ 200 ];  
VARIANT vari, type;  
.  
.  
.  
  
rc = pArsoCtrl->GetNumServerPrinters( &vari );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
num_prts = vari.iVal;  
  
for ( j = 0; j < num_prts; j++ )  
{  
    rc = pArsoCtrl->GetServerPrinter( j, name, type );  
    if ( rc != ARS_OLE_RC_SUCCESS )  
        ERROR;  
  
    // Process name and type  
  
}
```

.

GetServerPrinterInfo method

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 codes.

See Also

`GetNumServerPrinters` and `GetServerPrinter` methods

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

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.

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 codes:

- 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

Return Value

Refer to return codes.

See Also

StoreDoc method

C/C++

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

```
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

```

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 method

Retrieves the document type. 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

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

Return Value

Refer to return codes.

See Also

GetDocType method

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

.
.
.
```

IsDocHorzScrollRequired method

If the width of the document data exceeds the width of the control window, the result variable is set to a nonzero value; otherwise, to zero.

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

The displayed width of the document depends on the inherent width of its data, the type of data (that is, AFP versus Linedata), and the zoom factor.

Return Value

Refer to return codes.

See Also

ScrollDocHorz method

C/C++

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.

```
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

```

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

```

```

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

```

Logoff method

Logs off from the current server.

Method

```
short Logoff()
```

Description

Logs off from the current server.

Return Value

Refer to return codes.

See Also

Logon method

C/C ++

The following example performs a logoff.

```

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

```

Visual Basic

```

Dim rc As Integer
.
.

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

```

.
.

Logon method

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.

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 pServerName and pUserId point to meaningful pPassword strings and 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 codes.

See Also

GetNumServer, GetServerNames, Logoff, and SetLogonReturnOnFailure methods

C/C ++

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

```
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

```

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 method

Colors used for Content Manager OnDemand dialog boxes are updated to the current system colors. This method intended for use with C/C++.

Method

```
void OnSysColorChange()
```

Description

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

C/C++

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

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

OpenDoc method

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.

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 is taken from the specified file, but a resource group is 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.

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 codes.

See Also

`GetNumFolderDisplayFields`, `GetFolderDisplayFieldNames`, `GetNumDocsInList`, `GetDocDisplayValues`, `CloseDoc`, `CancelOperation`, `WasOperationCancelled`, and `ShowWaitCursorDuringCancelableOperation` methods

C/C++

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

```
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

```
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.GetFolderDisplayName(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

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

Method

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

Parameters

pFolderName

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

Description

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 codes.

See Also

GetNumFolders, GetFolderNames, CloseFolder, and CloseAllFolder methods

C/C++

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.

```
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

```
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

The page of the specified document is printed.

Method

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

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 nonzero, 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 nonzero, 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 of the specified document are printed.

Return Value

Refer to return codes.

See Also

GetNumDocsInList, and OpenDoc methods

C/C++

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

```
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

```
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

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.

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

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 might 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 codes.

See Also

`GetNumDocsInList`, and `SetResourceCacheMode` methods

C/C++

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.

```
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

```
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
Next count

:
:
```

ScrollDocHorz method

The document data is horizontally scrolled as indicated by Type and the current scroll position is returned in the variable pointed to by pPosition.

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

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 codes.

See Also

IsDocHorzScrollRequired and ScrollDocVert method

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 ++

```
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

```

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

```

ScrollDocVert method

The document data is vertically scrolled as indicated by Type and the current scroll position is returned in the variable pointed to by pPosition.

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

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 codes.

See Also

ScrollDocHorz method

C/C ++

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

```
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

```

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 - VertScrollOld
    If Diff = sbVert.LargeChange Then
        ScrollCode = ARS_OLE_SCROLL_PAGEDOWN
        rc = ArsOle.ScrollDocVert(ScrollCode, NewPos)
        VertScrollOld = NewPos
        sbVert.Value = NewPos
    ElseIf Diff = -sbVert.LargeChange Then
        ScrollCode = ARS_OLE_SCROLL_PAGEUP
        rc = ArsOle.ScrollDocVert(ScrollCode, NewPos)
        VertScrollOld = NewPos
        sbVert.Value = NewPos
    ElseIf Diff = sbVert.SmallChange Then
        ScrollCode = ARS_OLE_SCROLL_LINEDOWN
        rc = ArsOle.ScrollDocVert(ScrollCode, NewPos)
        VertScrollOld = NewPos
        sbVert.Value = NewPos
    ElseIf Diff = -sbVert.SmallChange Then

```

```

        ScrollCode = ARS_OLE_SCROLL_LINEUP
        rc = ArsOle.ScrollDocVert(ScrollCode, NewPos)
        VertScrollOld = NewPos
        sbVert.Value = NewPos
    Else
        ScrollCode = ARS_OLE_SCROLL_THUMBPOSITION
        NewPos = sbVert.Value
        rc = ArsOle.ScrollDocVert(ScrollCode, NewPos)
        VertScrollOld = sbVert.Value
    End If
End Sub

```

SearchFolder method

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

Method:

```

short SearchFolder(
    boolean Append )

```

Parameters

Append

If nonzero, 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 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 codes.

See Also

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

C/C ++

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.

```

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

```

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

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

Method

```
short SetDefaultFolderSearchFields()
```

Description

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

Return Value

Refer to return codes.

See Also

OpenFolder, and SearchFolder method

C/C ++

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

```

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

```

Visual Basic

```
Dim rc As Integer
```

```

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

```

```

.
.

```

SetDocBackgroundColor method

The document is displayed with the new background color.

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_GRAY

```

Description

The document is displayed with the new background color.

Return Value:

Refer to return codes.

See Also

SetDocBackgroundColor method

C/C++

The following example sets the document background color to gray.

```

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

```

Visual Basic

```

Dim rc As Integer

```

```

.
.

```

```

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

.
.

```

SetDocCurrentPage method

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.

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 codes.

See Also

GetDocCurrentPage, and GetDocNumPages method

C/C ++

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

```

CArsOle * pArsCtrl;
CScrollBar * pHorzScollBar, * pVertScrollBar;
short rc;
VARIANT horz_position, vert_position;
.
.
rc = pArsCtrl->SetDocCurrentPage( 46 );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.

rc = pArsCtrl->GetDocScrollPosition( &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

```
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

:
```

SetDocImageColor method

The document is displayed with the new image color.

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 codes.

See Also

GetDocImageColor method

C/C++

The following example sets the document image color to red.

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

Visual Basic

```
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

The document is displayed with the new image intensity.

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 codes.

See Also

GetDocImageIntensity method

C/C++

The following example sets the document image intensity to light.

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

Visual Basic

```
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

The document is displayed at the new rotation.

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 codes.

See Also

GetDocRotation method

C/C++

The following example rotates the document to 90 degrees.

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

Visual Basic

```
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

The document is displayed at the new zoom percent and the new scroll positions are returned in the specified variables.

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 scroll positions assume that the scroll ranges have been set to ARS_OLE_SCROLL_RANGE.

Return Value

Refer to return codes.

See Also

GetDocZoom method

C/C++

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

```
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

```
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 = ArsOle.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

Determines whether memory is to be released when a folder is closed. By default, the setting is false.

Method

```
short SetFolderCloseMemoryRelease(  
    boolean Release )
```

Parameters

Release

If nonzero, 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 codes.

See Also

CloseFolder method

C/C ++

The following example causes memory to be released when a folder is closed.

```
CArsOle * pArsCtrl;  
.  
.  
.  
pArsCtrl->SetFolderCloseMemoryRelease( TRUE );  
.  
.  
.
```

Visual Basic

```
.  
.  
.  
ArsOle.SetFolderCloseMemoryRelease (True)  
.  
.  
.
```

SetFolderSearchFieldData method

The search operator and values are set for the specified field for the active folder.

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 values are set for the specified field for the active folder.

Return Value

Refer to return codes.

See Also

GetNumFolderSearchFields and GetFolderSearchFieldNames, SearchFolder method

C/C++

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.

```
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

```

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

Determines the action to be taken when a logon fails. By default, the setting is false.

Method

```

short SetLogonReturnOnFailure(
    boolean Return )

```

Parameters

Return

If nonzero, indicates that control is to be returned when a logon fails; otherwise, that the Content Manager 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 codes.

See Also

Logon method

C/C ++

The following example sets the action to be taken when a logon fails.

```

CArsOle * pArsCtrl;
.
.
.
pArsCtrl->SetLogonReturnOnFailure( TRUE );
.
.
.

```

Visual Basic

```

.
.

```

```
.  
ArsOle.SetLogonReturnOnFailure (True)  
. .  
.
```

SetResourceCacheMode method

The requested mode is set.

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 are not 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 is required.

Description

The requested mode is set.

Return Value

Refer to return codes.

See Also

RetrieveDoc method

C/C++

The following example sets the resource cache mode to retain.

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

Visual Basic

```
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

Content Manager OnDemand creates a menu to be displayed when the user clicks the right mouse button in the document window.

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

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 are 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), clear.

This keyword is optional and is ignored for separator items. The default is for the item to be clear.

text


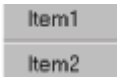
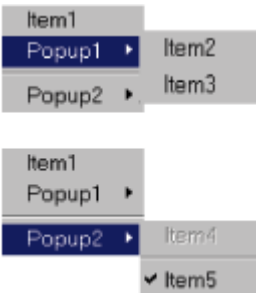
The text keyword specifies the text of the item. The text, which may include embedded blanks, must be contained in single quotation marks. If a single quote is part of the text, two consecutive single quotation marks must be specified.

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 through 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'	
level=0 id=44 text='Item1'\n level=0 id=1\n level=0 id=45 text='Item2'	
level=0 id=32457 text='Item1'\n level=0 id=0 text='Popup1'\n level=1 id=32458 text='Item2'\n level=1 id=32459 text='Item3'\n level=0 id=1\n level=0 id=0 text='Popup2'\n level=1 id=32460 enabled=0 text='Item4'\n level=1 id=1\n level=1 id=32461 checked=1 text='Item5'	

Return Value

Refer to return codes.

See Also

UserCommand event

C/C++

The following example shows how to create a right button menu that contains the command Copy Text.

```
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

```
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

Subsequent mouse selection by the user is performed in the specified mode.

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 Content Manager 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 Content Manager OnDemand client GUI.

Description

Subsequent mouse selection by the user is performed in the specified mode.

Return Value

Refer to return codes.

See Also

CopyBitmap and CopyText methods

C/C++

The following example sets the selection mode to area selection.

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

Visual Basic

```
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

The data associated with each index defaults to the empty string. Any data set using this method is retained until changed.

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 might be null.

Description

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 codes.

See Also

PrintDoc method

C/C++

The following example sets the FAX receiver name for server printers.

```
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

```
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

:
:

```

SeyUserMessageMode method

Subsequent Content Manager OnDemand exception messages resulting from OLE Control operations are displayed or suppressed as requested.

Method

```

short SeyUserMessageMode(
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 codes.

C/C++

The following example suppresses Content Manager OnDemand exception messages resulting from OLE Control operations.

```

CArsOle * pArsCtrl;
short rc;
:
:
rc = pArsCtrl->SetUserMessageMode( ARS_OLE_USER_MSG_MODE_SUPPRESS );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
:
:

```

Visual Basic

```

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

The Content Manager OnDemand Folder dialog box is displayed or hidden.

Method

```

short ShowFolder(
    boolean show,
    short Left,
    short top,
    short Right,
    short Bottom )

```

Parameters

Show

If nonzero, 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 nonzero 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 codes.

See Also

OpenFolder and ActivateFolder method

C/C++

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.

```

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

```

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

During cancelable operations, Content Manager OnDemand normally displays a wait cursor.

Method

```
short ShowWaitCursorDuringCancelableOperation(
    boolean Show)
```

Parameters

Show

If zero, indicates that Content Manager 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

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 codes.

See Also

CancelOperation method

C/C++

The following example causes a wait cursor not to be displayed during a cancelable operation.

```
CArsOle * pArsCtrl;  
.  
.  
pArsCtrl->ShowWaitCursorDuringCancelableOperation( FALSE );  
.  
.
```

Visual Basic

```
.  
.  
.  
ArsOle.ShowWaitCursorDuringCancelableOperation (False)  
.  
.  
.
```

StoreDoc method

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.

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 Content Manager 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 Content Manager 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 methods.

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

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.

If the return code is one of the following codes:

```
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 codes.

See Also

GetNumFolderFields, GetFolderFieldName, GetFolderFieldNames, and GetStoreDocInvalidFieldNum method

C/C++

```
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

```

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

Highlighting is removed from the current found string.

Method

```
short UndoFind()
```

Description

Highlighting is removed from the current found string.

Return Value

Refer to return codes.

See Also

FindStringInDoc

C/C ++

The following example removes highlighting from a found string.

```

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

```



```
.  
.
```

Visual Basic

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

UpdateDoc method

Content Manager OnDemand converts the folder field value to an application group field and updates the data from the specified value in the database.

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 are 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 codes.

See Also

GetNumFolderFields, GetFolderFieldName, and GetFolderFieldNames method

C/C++

The following example updates the Balance field of the first document within the document list of the active folder.

```
CArsOle * pArsCtrl;  
short rc;  
  
:  
:  
  
rc = pArsCtrl->UpdateDoc( 0,  
                          "Balance",  
                          "123.45" );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
:  
:
```

Visual Basic

```
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

If the most recent SearchFolder, OpenDoc, or RetrieveDoc operation was cancelled, the result variable is set to a nonzero value; otherwise, to zero.

Method

```
short WasOperationCancelled(  
    VARIANT * pCancelled )
```

Parameter

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 nonzero value; otherwise, to zero.

Return Value

Refer to return codes.

See Also

CancelOperation method

C/C++

The following example searches a folder and determines whether the search was cancelled.

```
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

```
Dim rc As Integer
Dim cancelled As Variant

:

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

rc = ArsOle.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 method

This event is fired when a folder search has completed.

The search is initiated:

- By the container application invoking the SearchFolder method
- By the user if the ShowFolder method was used to display the Content Manager OnDemand Folder dialog box.

FolderClosed method

This event is fired when a folder is closed.

The close is 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
- By destruction of the control

DocOpened method

This event is fired when a document is opened.

The open is 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 method

This event is fired when a document is closed.

The close is initiated:

- By the container application invoking the CloseDoc method
- By the folder associated with the document being closed
- By a logoff
- By destruction of the control

AreaSelected method

This event is fired when the user selects an area of the screen.

This is possible only if the `SetSelectionMode` method is used to set the selection mode to `ARS_OLE_SELECTION_MODE_AREA` or `ARS_OLE_SELECTION_MODE_TEXT`.

AreaDeselected method

This event is fired when a selection is removed.

The removal is caused by:

- 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
- 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.

Chapter 4. Content Manager OnDemand customization overview

The Content Manager OnDemand application can be customized by specifying command line parameters.

You can customize applications 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

You can start the product, show parameter syntax and other tasks with the command line.

This section describes:

- Starting the Content Manager OnDemand Windows Client.
- The parameter syntax rules used for the command line parameters.
- The parameters recognized by the Content Manager OnDemand Windows Client.

Starting the Content Manager OnDemand client

Content Manager OnDemand client can be started by double-clicking on the program icon on your desktop.

Parameter syntax

Several rules apply to command line parameters.

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.

```
C:\<install dir>\arsgui.exe /S pikes /U user1 /P pass1 /C newpass
```

where <install dir> is your OnDemand Client installation directory path.

Code Page Override — /A codepage_number

If this parameter is specified, Content Manager 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.

```
C:\<install dir>\arsgui.exe /A 1392
```

where <install dir> is your OnDemand Client installation directory path.

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.

```
C:\<install dir>\arsgui.exe /V
```

where <install dir> is your OnDemand Client installation directory path.

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.

```
C:\<install dir>\arsgui.exe /Z
```

where <install dir> is your OnDemand Client installation directory path.

Disable Exit —/K

If this parameter is specified, Content Manager OnDemand disables the Exit menu item.

```
C:\ARS\ARSGUI /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.

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.

```
C:\<install dir>\arsgui.exe /X
```

where <install dir> is your OnDemand Client installation directory path.

Disable Update Servers —/Y

If this parameter is specified, Content Manager OnDemand disables the Update Servers button on the Logon dialog box.

```
C:\<install dir>\arsgui.exe /Y
```

where <install dir> is your OnDemand Client installation directory path.

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.

```
C:\<install dir>\arsgui.exe /B
```

where <install dir> is your OnDemand Client installation directory path.

Enable DDE Interface — /I number,path,resid

If this parameter is specified, Content Manager OnDemand enables the DDE interface and prepares to accept commands from another application.

number

Must be an integer between 1 and 5. It indicates the number of DDE-application switch menu items to be provided. 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.

```
C:\ARS\ARSGUI /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 can be used in combination with the Logon Server Name, Logon Password, Disble Logoff, Password Change, Maximum Open Folders, and Logon User ID, parameters to limit a user to operations with a single folder on a single server.

```
C:\<install dir>\arsgui.exe /F Credit Card Statements
```

where <install dir> is your OnDemand Client installation directory path.

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.

```
C:\<install dir>\arsgui.exe /Q
```

where <install dir> is your OnDemand Client installation directory path.

Language ID Override — /L

If this parameter is specified, Content Manager 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.

```
C:\<install dir>\arsgui.exe /L JPN
```

where <install dir> is your OnDemand Client installation directory path.

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.

```
C:\<install dir>\arsgui.exe /S pikes /U user1 /P pass1
```

where <install dir> is your OnDemand Client installation directory path.

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.

```
C:\<install dir>\arsgui.exe /S pikes /U user1 /P pass1
```

where <install dir> is your OnDemand Client installation directory path.

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.

```
C:\<install dir>\arsgui.exe /S pikes /U user1 /P pass1
```

where <install dir> is your OnDemand Client installation directory path.

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 is 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.

```
C:\<install dir>\arsgui.exe /O 1 /Q
```

where <install dir> is your OnDemand Client installation directory path.

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.

```
C:\<install dir>\arsgui.exe /T Joe's Windows Application
```

where <install dir> is your OnDemand Client installation directory path.

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 parameters:

- 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.

```
C:\<install dir>\arsgui.exe /W 5,10,90,80
```

where <install dir> is your OnDemand Client installation directory path.

CD-ROM Mastering

Content Manager OnDemand enables you to extract data from a server, and transfer it onto CDs, which are easily distributable. When you do this, each CD you create becomes a Content Manager OnDemand server, and the way you access data from the CD is virtually identical to the way you access the same data from a regular network server.

CD-ROM Mastering functionality is enabled automatically when you install the 32-bit OnDemand Client. CD-ROM Mastering is not supported by the 64-bit OnDemand Client.

If you need to disable CD-ROM Mastering for a particular user, you need to change a key in the registry. For each user for whom you would like this functionality disabled, change the following registry key to 0 (zero) under HKEY_CURRENT_USER:

```
HKEY_CURRENT_USER\Software\IBM\OnDemand32\Client\CDROM\VERSION = 0
```

If you want to disable the CD-ROM Mastering functionality for all users, then change the following registry key to 0 (zero) under HKEY_LOCAL_MACHINE:

```
HKEY_LOCAL_MACHINE\SOFTWARE\IBM\OnDemand Clients V10.1\CDROM\VERSION = 0
```

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.

Note: DDE is supported by the 32-bit OnDemand client. DDE is not supported by the 64-bit OnDemand client.

Invoking the Content Manager OnDemand client from another Windows application

This invocation method is typically used when the DDE interface is required.

The Content Manager OnDemand client 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\\ARSGUI "
```

```

"/T Special Content Manager 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, 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.

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.

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 XTYPE_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 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. 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;
    HDDEDATA 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

Content Manager OnDemand makes the document the active document by bringing its document window to the top and giving it the input focus.

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 Codes

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.

Refer to the DoDdeCommand function.

```
DoDdeCommand( "ACTIVATE_DOC", "/D 1234", NULL );
```

ACTIVATE_FOLDER command

Content Manager OnDemand makes the specified folder the active 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.

Refer to the DoDdeCommand function.

```
DoDdeCommand( "ACTIVE_FOLDER", "Mary's Folder", NULL );
```

ANNOTATE_DOC command

Content Manager OnDemand adds the annotation to the specified document.

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. The values associated with a particular document number can be retrieved by using the GET_DOC_VALUES command.

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.

Refer to 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

Content Manager OnDemand arranges the document windows as indicated by the parameter.

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

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.

Refer to the DoDdeCommand function.

```
DoDdeCommand( "ARRANGE_DOCS", "/V", NULL );
```

CHANGE_PASSWORD command

Content Manager OnDemand changes the logon password for the active user.

Command	Parameters
CHANGE_PASSWORD	<i>/C Current Password /N New Password /V 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

Content Manager 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 the DoDdeCommand function.

```
DoDdeCommand( "CHANGE_PASSWORD", "/C tt1sd /N sfd45r /V sfd45r", NULL );
```

CLEAR_FIELDS command

Content Manager OnDemand clears the search criteria entry windows for the active folder.

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 the DoDdeCommand function.

```
DoDdeCommand( "CLEAR_FIELDS", "", NULL );
```

CLOSE_ALL_DOCS commands

Content Manager OnDemand closes all open documents, destroys all document windows, and displays the active folder window.

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.

Refer to the DoDdeCommand function.

```
DoDdeCommand( "CLOSE_ALL_DOCS", "", NULL );
```

CLOSE_ALL_FOLDERS command

Content Manager OnDemand closes all open 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

Refer to the DoDdeCommand function.

```
DoDdeCommand( "CLOSE_ALL_FOLDERS", "", NULL );
```

CLOSE_DOC command

Content Manager OnDemand closes the document by destroying the document window

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

Refer to the DoDdeCommand function.

```
DoDdeCommand( "CLOSE_ALL_DOCS", "/D 1234", NULL );
```

CLOSE_FOLDER command

Content Manager OnDemand closes the active folder. If there are other open folders, one of them becomes the active 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

Refer to the DoDdeCommand function.

```
DoDdeCommand( "CLOSE_FOLDER", "", NULL );
```

COPY_DOC_PAGES command

Content Manager OnDemand copies the specified pages from the active document to the specified file.

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 in the data stream format; otherwise, the pages are copied as ASCII text (for AFP and line data only). This parameter is optional.

Action

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_ACTIVITY_DOC

Return Data

None

Refer to 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

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. The values associated with a particular document number can be retrieved by using the GET_DOC_VALUES command. This parameter is required.

The *doc number* can 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 documents 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 the DoDdeCommand function.

```
DoDdeCommand( "DELETE_DOC", "23", NULL );
```

DESELECT_DOC command

Content Manager OnDemand deselects (removes highlight from) the Document List line that corresponds to the specific document number.

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. The values associated with a particular document number can be retrieved by using the GET_DOC_VALUES command. This parameter is required.

The *doc number* can 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

Refer to the DoDdeCommand function.

```
DoDdeCommand( "DESELECT_DOC", "15", NULL );
```

DISABLE_SWITCH command

Content Manager OnDemand disables a toolbar icon and a menu item that allow the user to transfer window focus to the client application.

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 DDE Interface `--I number, path, resid`. 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

Refer to the DoDdeCommand function.

```
DoDdeCommand( "DISABLE_SWITCH", "", NULL );
```

ENABLE_SWITCH command

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.

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. 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

If the client application has established an Advise Loop, 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

Refer to 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

Content Manager OnDemand attempts to terminate when this command is received.

Command	Parameters
EXIT	None

Parameters

None

Action

If at least one document is open, and the Disable User Confirmation command line parameter has not been specified, 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

Refer to the DoDdeCommand function.

```
DoDdeCommand( "EXIT", "", NULL );
```

GET_DISPLAY_FIELDS command

Content Manager OnDemand returns a list of the display fields for the active folder.

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

11

ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data

The list is a null-terminated character string with the tab character separating each field name.

Refer to 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

Content Manager OnDemand returns a list of the document values for the specified document within the document list of the active folder.

Command	Parameters
GET_DOC_VALUES	<i>/N doc number /S</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. 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

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.

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.

Refer to 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

Content Manager OnDemand returns a list of the folder fields for the active folder.

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

The list is a null-terminated character string with the tab character separating each field name.

Refer to 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

Content Manager OnDemand returns a list of the available 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

The list is a null-terminated character string with the tab character separating each folder name.

Refer to 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

Content Manager OnDemand returns the number of documents currently in the document list of the active folder.

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

The returned null-terminated string can be converted to a long integer.

Example

Refer to 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

Content Manager OnDemand returns the number of pages in the active document.

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.

Refer to the DoDdeCommand function.

```
char data[50];
long num_pages;

if ( DoDdeCommand( "GET_NUM_DOC_PAGES.", "", data ) )
    num_pages = atol( data );
```

GET_PRINTERS command

Content Manager OnDemand returns a list of the available printers, either Local or Server printers as indicated by the parameter.

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

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.

Refer to 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 commands

Content Manager OnDemand returns a list of the query fields for the active folder.

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

The list is a null-terminated character string with the tab character separating each field name.

Refer to 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 commands

Content Manager OnDemand returns a list of the available 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

The list is a null-terminated character string with the tab character separating each server name.

Refer to 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

Content Manager OnDemand performs a 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

Refer to the DoDdeCommand function.

```

DoDdeCommand( "LOGOFF", "", NULL );

```

LOGON command

Content Manager OnDemand attempts a logon with the specified server name, user ID, and password.

Command	Parameters
LOGON	<i>/S server name /U user id /P password /R</i>

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 Content Manager OnDemand Logon dialog box is displayed.

Action

A list of the available server names can be retrieved by using the GET_SERVERS command. 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.

Refer to the DoDdeCommand function.

```
char parms[200];

sprintf( parms,
         "/S %s /U %s /P %s",
         "server26",
         "horatio",
         "vixotwc" );

DoDdeCommand( "LOGON", parms, NULL );
```


OPEN_DOC command

Content Manager OnDemand opens the document by displaying the first page in a document window.

Command	Parameters
OPEN_DOC	/N <i>doc number</i> /P <i>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. The values associated with a particular document number can be retrieved by using the GET_DOC_VALUES command.

P

Specifies the fully-qualified path of a file containing the document data. This parameter is required. If this parameter is specified, Content Manager OnDemand does not access the database to retrieve a document, but it retrieves 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.

Refer to the DoDdeCommand function.

```
char doc_id[21];  
DoDdeCommand( "OPEN_DOC", "/N 23", doc_id );
```

OPEN_FOLDER command

Content Manager OnDemand attempts to open the specified 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 SearchCriteria 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 SearchCriteria 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 SearchCriteria 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 SearchCriteria 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 SearchCriteria 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 SearchCriteria 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 SearchCriteria 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 SearchCriteria 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 DocumentList 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 DocumentList of the **Search Criteria and Document List** dialog box. This parameter is optional.

Obutton text

Specifies the text for a button which is to appear on the SearchCriteria of the **Search Criteria and Document List** dialog box. This parameter is optional.

If the text includes an &, the next 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.

1button text

Specifies the text for a button which is to appear on the SearchCriteria of the **Search Criteria and Document List** dialog box. This parameter is optional.

If the text includes an &, the next 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.

2button text

Specifies the text for a button which is to appear on the SearchCriteria of the **Search Criteria and Document List** dialog box. This parameter is optional.

If the text includes an &, the next 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.

3button text

Specifies the text for a button which is to appear on the SearchCriteria of the **Search Criteria and Document List** dialog box. This parameter is optional.

If the text includes an &, the next 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.

4button text

Specifies the text for a button which is to appear on the SearchCriteria of the **Search Criteria and Document List** dialog box. This parameter is optional.

If the text includes an &, the next 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.

5button text

Specifies the text for a button which is to appear on the SearchCriteria of the **Search Criteria and Document List** dialog box. This parameter is optional.

If the text includes an &, the next 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.

6button text

Specifies the text for a button which is to appear on the DocumentList of the **Search Criteria and Document List** dialog box. This parameter is optional.

If the text includes an &, the next 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.

7button text

Specifies the text for a button which is to appear on the DocumentList of the **Search Criteria and Document List** dialog box. This parameter is optional.

If the text includes an &, the next 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.

8button text

Specifies the text for a button which is to appear on the DocumentList of the **Search Criteria and Document List** dialog box. This parameter is optional.

If the text includes an &, the next 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.

button text

Specifies the text for a button which is to appear on the DocumentList of the **Search Criteria and Document List** dialog box. This parameter is optional.

If the text includes an &, the next 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.

Action

A list of the available folder names can be retrieved by using the GET_FOLDERS command. 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

Refer to the DoDdeCommand function.

```
DoDdeCommand( "OPEN_FOLDER", "/F Mary's Folder", NULL );
```

PRINT_DOC command

Command	Parameters
PRINT_DOC	/N doc number /L printer name / Sprinter name /C copies /R orientation /K margins /M

Parameters

Ndoc 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. The values associated with a particular document number can be retrieved by using the GET_DOC_VALUES command.

This parameter is optional. If it is not specified, the active document is printed.

Lprinter name

Specifies a Local printer name. The names of the available local printers can be determined by using the GET_PRINTER command.

This parameter is optional, but either this or the S parameter must be specified.

Sprinter

Specifies a Server printer name. The names of the available server printers can be determined by using the GET_PRINTER.

This parameter is optional, but either this or the L parameter must be specified.

Ccopies

Specifies the number of copies of the document to be printed. The value must be a number between 1 and 100.

Rorientation

This parameter is optional. If not specified, one copy is printed.

Specifies the document orientation for printing. For *orientation*, specify one of the following parameters:

- 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.

Kmargins

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 pages 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

Refer to 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 commands

Content Manager OnDemand sets the search criteria entry windows for the active folder to their default values.

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

Refer to the DoDdeCommand function.

```
DoDdeCommand( "RESTORE_DEFAULTS", "", NULL );
```

RETRIEVE_DOC command

Content Manager OnDemand retrieves the resource group and/or document data and copies or appends it to the specified files.

Command	Parameters
RETRIEVE_DOC	<i>/N doc number /P doc path /R resgrp path /C combined path /A</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. The values associated with a particular document number can be retrieved by using the GET_DOC_VALUES command.

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

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 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

Content Manager OnDemand searches the active folder using the current search criteria.

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

Refer to the DoDdeCommand function.

```
DoDdeCommand( "SEARCH_FOLDER", "", NULL );
```

SELECT_DOC command

Content Manager OnDemand selects (highlights) the Document List line that corresponds to the specific document number.

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. The values associated with a particular document number can be retrieved by using the GET_DOC_VALUES command.

This parameter is required.

The *doc number* can 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

Refer to the DoDdeCommand function.

```
DoDdeCommand( "SELECT_DOC", "-1", NULL );
```

SET_FIELD_DATA command

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.

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.

O

Specifies the search operator to be used for the field. It must be one of the following choices:

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

Refer to 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

Content Manager OnDemand becomes the active window.

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

Refer to the DoDdeCommand function.

```
DoDdeCommand( "SET_FOCUS", "", NULL );
```

SET_HELP_PATH command

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.

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

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

Refer to the DoDdeCommand function.

```
DoDdeCommand( "SET_HELP_PARM", "C:\DDEAPPL\DDEAPPL.HLP", NULL);
```

SET_USER_MSG_MODE command

Content Manager OnDemand displays or suppresses subsequent user messages as specified by the parameters.

Command	Parameters
SET_USER_MSG_MODE	/M <i>mode</i>

Parameters

M

Specifies the user message mode.

For *mode*, specify one of the following choices:

- 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

Refer to the DoDdeCommand function.

```
DoDdeCommand( "SET_USER_MSG_MODE", "/M 2", NULL);
```

SHOW_WINDOW command

Content Manager OnDemand shows and/or positions its main window as specified by the parameter value.

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 Windows Placement command line parameter.

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

Refer to the DoDdeCommand function.

```
DoDdeCommand( "SHOW_WINDOW", "/W 25,0,75,100", NULL );
```

STORE_DOC command

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.

Command	Parameters
STORE_DOC	<i>/P doc path /G appl group name /A appl name /n 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 group 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.

The number and order of folder fields can be determined by using the GET_FOLDER_FIELDS command.

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.

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 codes:

- ARS_DDE_RC_INVALID_INTEGER_FIELD
- ARS_DDE_RC_INVALID_DATE_FIELD
- ARS_DDE_RC_INVALID_DECIMAL_FIELD
- ARS_DDE_RC_INVALID_APPLGRP_FIELD_TYPE
- ARS_DDE_RC_TOO_MANY_VALUE_CHARS

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, if the second folder field is invalid, Content Manager OnDemand returns a 1.

C/C++

Refer to the DoDdeCommand function.

```
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

```
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 )
```

UPDATE_DOC command

Content Manager OnDemand converts the folder field value to an application group field and updates the data from the specified value in the database.

Command	Parameters
UPDATE_DOC	<i>/N doc number /F field name /V 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.

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.

Refer to the DoDdeCommand function.

```
DoDdeCommand( "UPDATE_DOC", "/N 1 /F Balance /V 123.45", NULL );
```

Chapter 8. Return Codes

DDE commands have return codes for various commands.

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 user ID, 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 might create a DDEML Advise Loop in order to be informed when various events occur.

The `DDEMLDdeClientTransaction` 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 might 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 and can occur due to the following actions:

- 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.

Chapter 10. External applications and dynamic link libraries

For the Windows client, Content Manager OnDemand provides menu and toolbar extensions that allow a 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:
Value Names and Value Data for external applications and Dynamic link Library Keys in the registry.

<i>Table 1: External Applications and Dynamic Link Library Keys in the Registry</i>		
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 are specified. The DLLs can be divided between the HKEY_CURRENT_USER and HKEY_LOCAL_MACHINE keys. The key names can 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 is not 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 ('*') is 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 a document is being viewed and the folder associated with the active 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.

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 a document is being viewed and the folder associated with the active 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, enablement is controlled by the Folders value.

Doc

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 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.

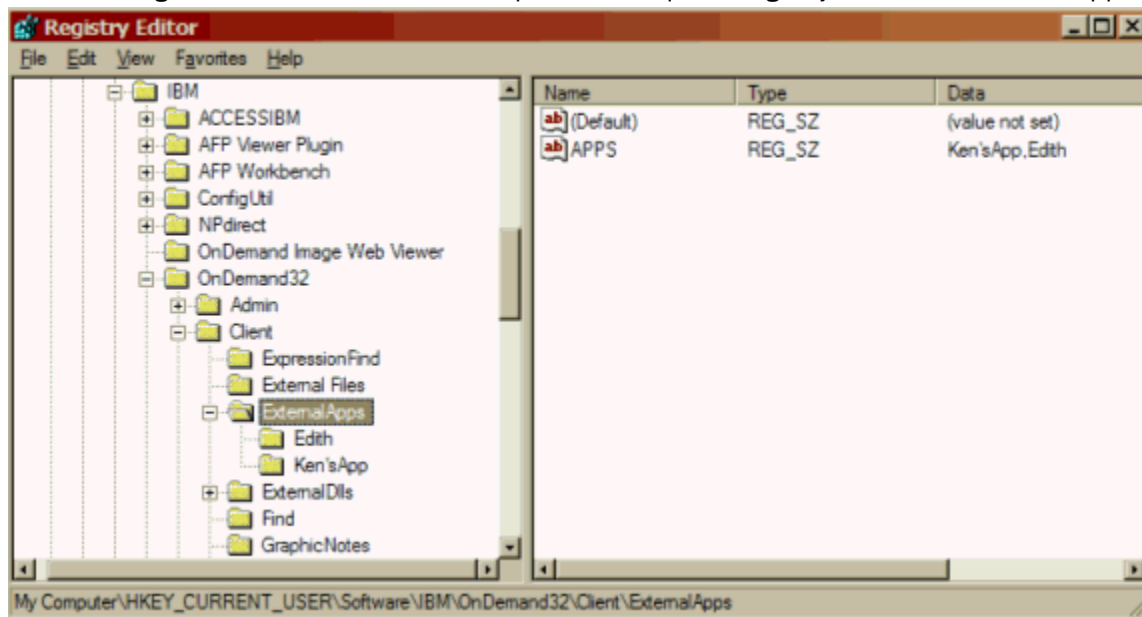


Figure 1: Example of External Applications in the Registry Part 1 of 3

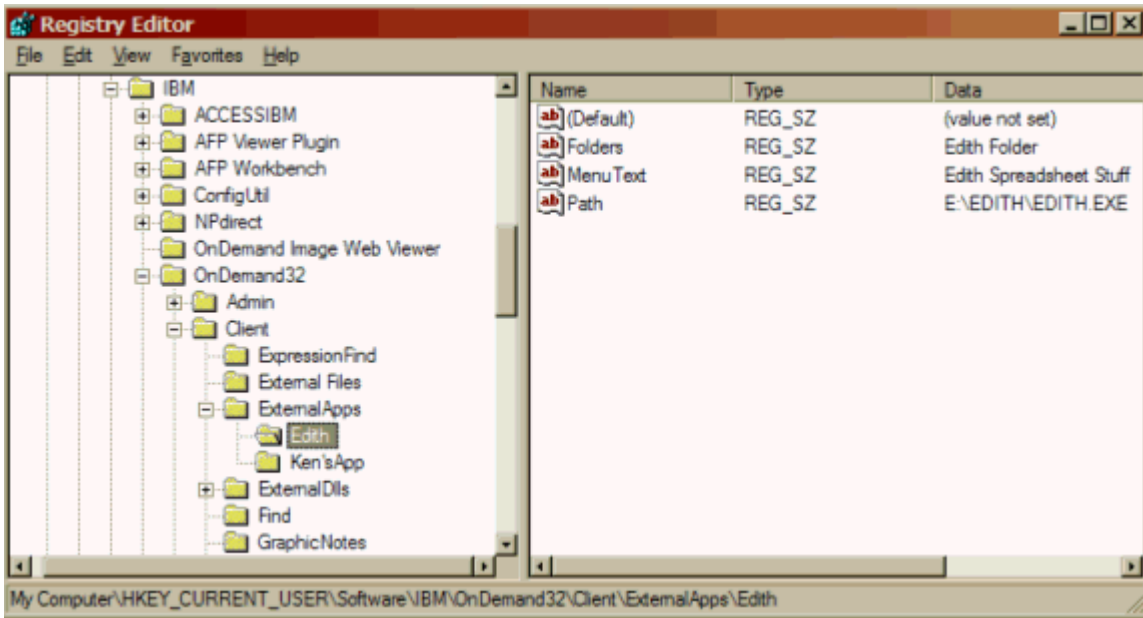


Figure 2: Example of External Applications in the Registry Part 2 of 3

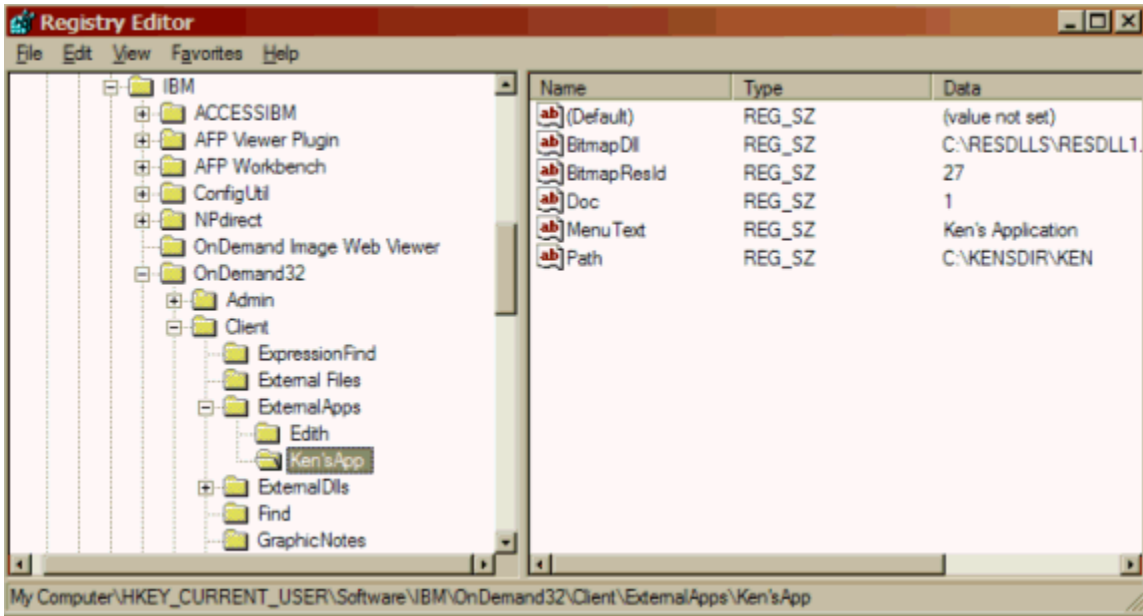


Figure 3: Example of External Applications in the Registry Part 3 of 3

The following two screens show an example of the required registry entries for External DLLs.

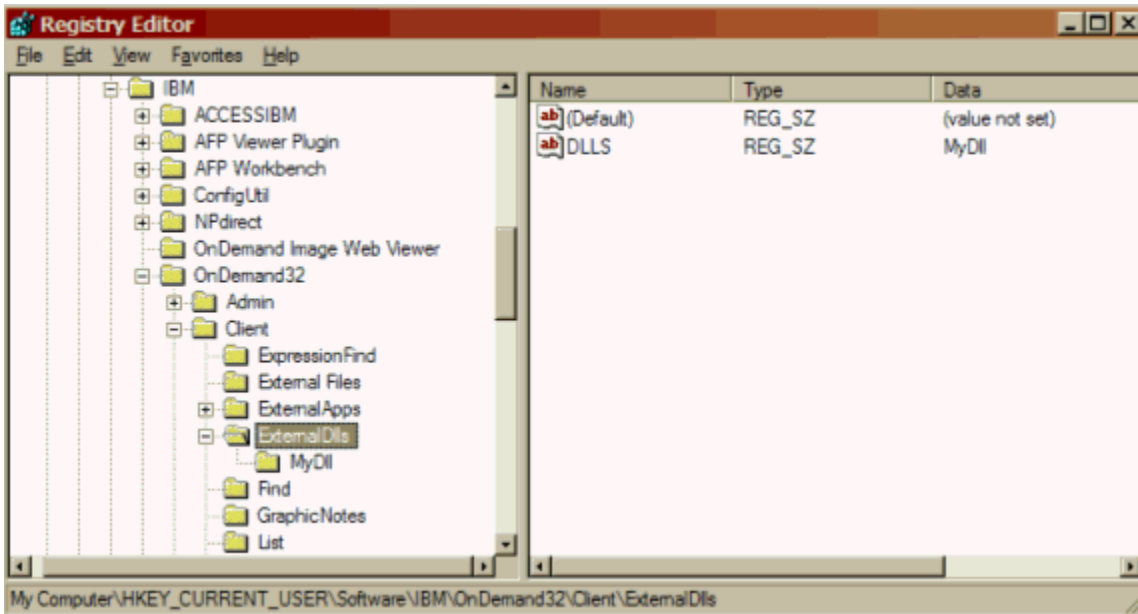


Figure 4: Example of External DLLs in the Registry Part 1 of 2

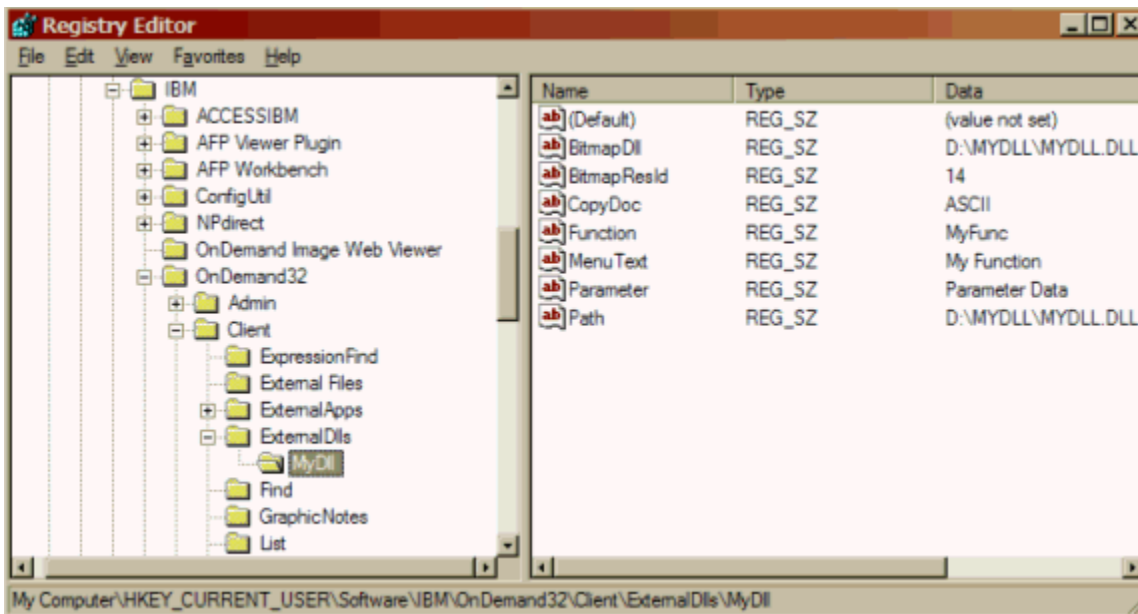


Figure 5: Example of External DLLs in the Registry Part 2 of 2

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

The path specified with the Path value in the registry.

folder

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

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

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

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

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

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

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,
              wchar_t * pFolderName,
              wchar_t * pSelectedText,
              wchar_t * pAttributeAndValue,
              wchar_t * pFieldsAndValues,
              wchar_t * pFilename,
              wchar_t * pParameterData );
```

Where:

page_number

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

A pointer to a null-terminated unicode 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

A pointer to a null-terminated unicode 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

A pointer to a null-terminated unicode 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

A pointer to a null-terminated unicode 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

a pointer to a null-terminated unicode 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

A pointer to a null-terminated unicode 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 a 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 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\OnDemand Clients V10.1 key in the Windows system registry for the following keys and string values:

Value names and Value data for the document keys in the registry.

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 can be any string.

The string values are 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 is not 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 (*) can 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 a document is being viewed and the folder associated with the active 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.

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 a document is being viewed and the folder associated with the active 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, 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

The name of a field defined for the related folder.

operator

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

the first or only value to be used for the field. It may be any string, including an empty string, and can 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

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 choices:

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 in the **Maintain Document Arrangement** menu item.

The following three screen captures show an example of registry entries required to support two related documents.

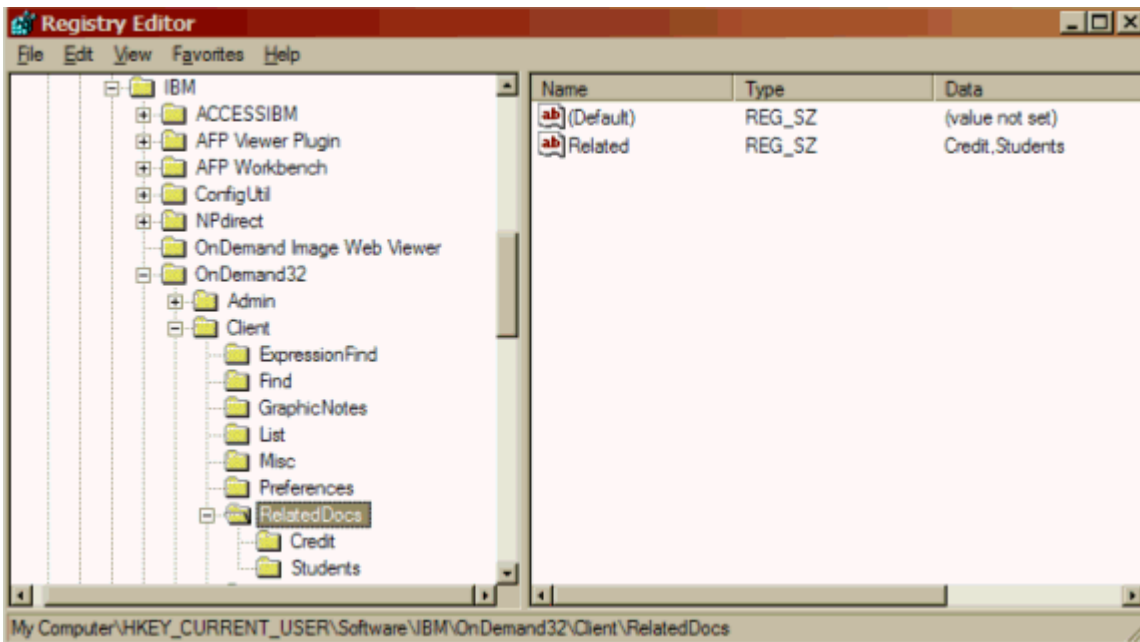


Figure 6: RelatedDocs key showing values for two related documents

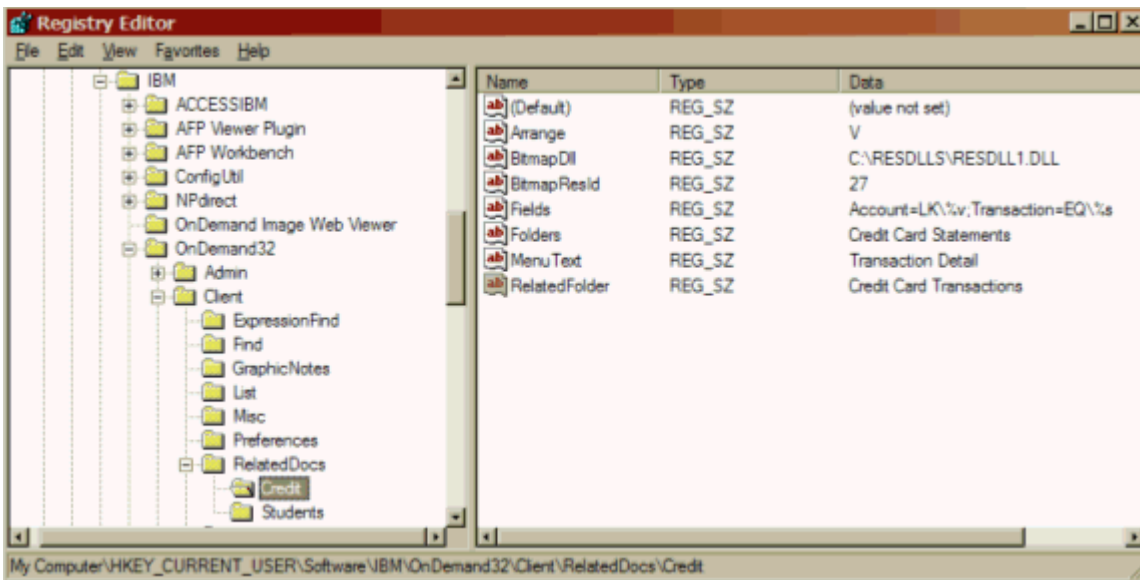


Figure 7: Credit key showing values for the first related document

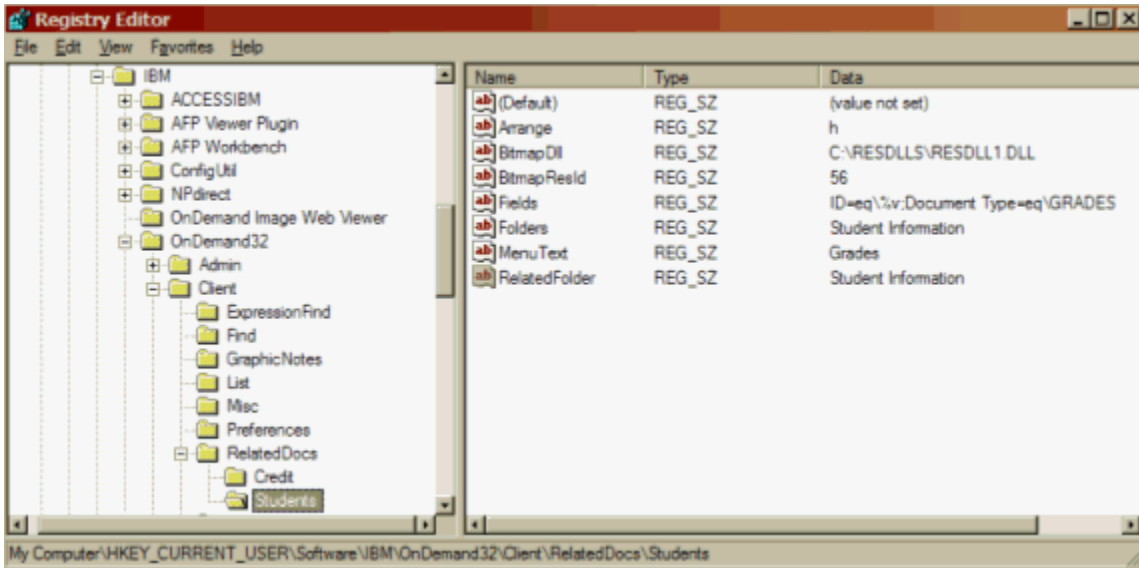


Figure 8: Students key showing values for the second related document

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_LINEn

Specifies each line of the About dialog box text. ABOUT_LINE1 places the first line ABOUT_LINE2 in the ABOUT_LINE8 parameter.

Chapter 13. Document Audit Facility

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 Content Manager 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, Content Manager OnDemand displays up to 8 status buttons per folder, such as **Accept**, **Reject**, **Hold**, and **Escalate**.

Users that need to audit documents must be given permission to update documents.

The following topics contain additional information:

- Creating the DAF control file
- Defining the report
- Control access to the DAF
- Using the DAF

Create the DAF control file

The DAF is controlled by the ARSGUI.CFG file, which you must create and store in the Windows client program directory (\Program Files\IBM\OnDemand Clients\V10.1\bin 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.

Sample DAF control file

```
[AUDIT]
Folders=Invoices

[Invoices]
FOLDER=Invoices - Auditor
AUDIT_FIELD=Status
TEXT1=Accept
TEXT2=Reject
TEXT3=Escalate
TEXT4=Hold
VALUE1=A
VALUE2=R
VALUE3=E
VALUE4=H
COLUMNS=2
```

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.

Important: The total number of characters in the FOLDERS record must not exceed 255.

Folder section

Each FOLDER section contains the following records:

FOLDER

Specifies the name of the folder, exactly as it appears in Content Manager OnDemand. The FOLDER record is required.

AUDIT_FIELD

Specifies the name of the folder field used to audit documents, exactly as it appears in Content Manager OnDemand. The AUDIT_FIELD record is required.

TEXTx

The caption that appears on the command button used to change the status of the document. Up to 8 TEXT settings are permitted.

VALUEx

The value that is stored in the database when the corresponding TEXTx button is clicked. This value is stored in the application group field and must match one of the mapped field values. One VALUE record is required for each TEXT record. Up to 8 VALUE settings are permitted.

COLUMNS

Specifies how many columns the buttons should be arranged in (maximum of 2).

Important: You must restart the Content Manager OnDemand client after creating the DAF control file.

Define the report

You can define a report that uses the DAF.

About this task

The information provided is in addition to all of the other attributes you need to specify when you define a report to Content Manager 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.

Procedure

To define the attributes of the audit field on the **Field Information** page.

1. In the String area, set the Case to **Upper, Type to Fixed, and Length to 1 (one)**.
2. 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.
3. Click the Updateable check box to select the option. This allows the index values for the audit database field to be changed.
4. You must add one set of values for each audit value that will be used in the DAF file. Up to 8 audit values can be specified in the DAF file.

The Database Value must match the value of the VALUE_n record in the folder section of the DAF file. We recommend that the Displayed Value match the value of the TEXT_n value record in the folder section of the DAF file.

Table 3: Examples of Database Values and Displayed Values in the DAF file.

Database Value	DAF File	Displayed Value	DAF File
A	VALUE1=A	Accept	TEXT1=Accept
R	VALUE2=R	Reject	TEXT2=Reject
E	VALUE3=E	Escalate	TEXT3=Escalate
H	VALUE4=H	Hold	TEXT4=Hold

Defining applications

You can define the application on the **Load Information** page.

Procedure

- Add the default value for the audit field on the **Load Information** page.
This is the value that Content Manager OnDemand stores in the database for all documents when a report is loaded into the application group. IBM recommends that you set the default value to N (for None or Not Audited).
The default value must match one of the values defined on the application group **Field Information** page.

Defining Folders

You can define a folder on the **Field Definition** page.

Procedure

- 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.
Map the folder audit field to the application group audit field on the **Field Mapping** page.

Control access to the DAF

There are several ways you can manage access to documents that need to be audited.

About this task

To simplify administration of the system, we encourage you to use groups. For example, some customers 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 might 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 Content Manager OnDemand, authorized users can use the DAF to audit the documents.

About this task

Content Manager OnDemand updates the database with the status field values specified in the DAF control file.

Procedure

To audit a document:

1. Open one of the folders defined in the DAF control file.
2. Search the folder for documents that need to be audited.
For example, search for documents that contain the value **Hold** in the audit field.
3. Select one or more of the documents from the list.
4. Click one of the audit buttons to update the audit field value for each selected document. For example, **Hold**, **Accept**, **Reject**, or **Escalate**.

Chapter 14. Modifying client behavior through the Registry

You must edit the registry on the computer. If there is an error in the registry, the computer might not function properly.

Before you begin

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.

About this task

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

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_

Used to override the initial font used for line data documents. The first time that a line data document is opened after Content Manager OnDemand is initially installed, the client selects a font. If any of these values are specified, Content Manager OnDemand searches 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 Content Manager 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

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

This function allows users to automatically load documents from the Content Manager OnDemand client into Monarch.

This section provides information about how to integrate Monarch with the Content Manager OnDemand Windows client. Monarch is a software program that is available from Datawatch Corporation. The user can then do complex data manipulation from Monarch, such as creating derived columns and generating charts and reports.

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 Content Manager OnDemand. This section describes how to do some of the tasks, but you will need other Content Manager OnDemand information and your Monarch information to do others.

A reference workstation contains an installed copy of the Content Manager OnDemand client with the Monarch DLL defined to Content Manager OnDemand. An administrator uses configuration information from the reference workstation to distribute software to other users. A distribution server is a network file server that contains a copy of the Content Manager OnDemand client installation software in a shared location. Other users on the network use the copy to run the Content Manager OnDemand installation program and install the client from the distribution server to their workstations.

An administrator can use the Content Manager OnDemand Windows client installation program to distribute Monarch files with the Content Manager OnDemand client and configure workstations to run Monarch from the Content Manager OnDemand client. IBM recommends that an administrator configure a copy of the Content Manager OnDemand client software on a reference workstation and configure the installation program on a distribution server. Each user planning to run Monarch from Content Manager OnDemand must have the Monarch software installed on their workstation before they install or upgrade the Content Manager OnDemand client with this function.

An administrator can store the Monarch files in the Content Manager OnDemand Windows client installation directory tree on a distribution server. The installation program copies the Monarch files along with the standard Content Manager 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 Content Manager OnDemand client must be present for the installation program to integrate Content Manager 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 back up 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.

Configuring the client

Before you begin

The following tasks must be completed prior to configuring the client:

- Installed Monarch on the reference workstation. See your Monarch information for details.

About this task

This chapter describes how to configure the client on the reference workstation with the information needed to integrate Monarch with Content Manager OnDemand. If you have not done so already, install a

copy of Monarch. (See your Monarch information for details.) Then install a copy of the Content Manager OnDemand client software. (See additional Content Manager OnDemand information for details.) Next, add the Registry key, values, and value data that define the properties of the Monarch DLL to the Content Manager OnDemand client. Then verify that you have correctly integrated Monarch with Content Manager OnDemand by starting the Content Manager 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.

Adding the Registry key

Placing information in the Registry activates Monarch integration with the Content Manager OnDemand client. If during initialization, Content Manager OnDemand detects this information, it adds menu items and toolbar buttons to the client workspace.

Before you begin

You must install the Content Manager OnDemand Windows client on the reference workstation before you add the Registry key.

About this task

When the user chooses one of the menu items or clicks one of the toolbar buttons, Content Manager OnDemand calls the associated entry point in the Monarch DLL. The Monarch DLL subkey values provides information about the subkey values and value data shown in the Subkey and values for the Monarch DLL. Other DLL subkey values provides information about other values specified.

Subkeys under ExternalDlls contain the information used by Content Manager OnDemand to call the DLLs integrated into the client. The graphic 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, Content Manager OnDemand simply starts Monarch with the current document.

When a user invokes Monarch from the client and no model file is specified in the Registry key, Content Manager 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.

Value and Data	Description
BitmapDll C:\Program Files(x86)\IBM\OnDemand Clients\10.1\bin\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 Content Manager 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
<p>CopyDoc ASCII</p>	<p>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.</p> <p>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.</p> <p>This value is optional. If not provided, no document is provided to Monarch.</p>
<p>Doc 1</p>	<p>Specified as one of the following values:</p> <p>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.</p> <p>1 (one) indicates that the menu item and toolbar button are enabled only when a document is being viewed.</p> <p>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.</p> <p>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.</p> <p>This value is optional. If not provided, a value of 3 (three) is assumed.</p>
<p>Function MonarchFunc</p>	<p>The name of the entry point into the Monarch DLL file. This value must be provided.</p>

Table 5: Monarch DLL subkey values

Value and Data	Description
<p>MenuText Monarch</p>	<p>The text that appears on the menu item used to invoke Monarch from the Content Manager 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.</p>
<p>Path C:\Program Files(x86)\IBM\OnDemand Clients \V10.1\bin\ARSMON.DLL</p>	<p>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.</p>

Table 6: Other DLL subkey values

Value	Description
ExcludeFolders	<p>Specifies the names of one or more Content Manager OnDemand folders. The syntax is the same as the Folders value.</p> <p>The menu item and toolbar button are enabled whenever:</p> <ul style="list-style-type: none"> • 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. <p>This value is optional. If not provided, enabling is controlled by the Folders value.</p>
Folders	<p>Specifies the names of one or more Content Manager 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.)</p> <p>The menu item and toolbar button are enabled whenever:</p> <ul style="list-style-type: none"> • 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. <p>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.</p>
Parameter	<p>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.</p> <p>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 Content Manager 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.</p>

Procedure

To add the Registry key:

- The ExternalDlls subkey contains a list of the DLLs that are integrated with the Content Manager 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 Content Manager 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 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 Content Manager OnDemand Windows client (and tested that you can invoke Monarch from the client), you can export it to a file.

About this task

You store the file in the CUSTOM subdirectory tree under the Content Manager OnDemand Windows client installation directory tree on a distribution server. Once you have placed the file there, users installing the Content Manager OnDemand client will automatically have the Registry key imported into the Registry on their workstation when they run Setup from the distribution server.

Important: If you have other applications integrated with Content Manager OnDemand (that is, you have DLLs from applications other than Monarch defined under the ExternalDlls subkey), contact the IBM support center before you proceed.

Procedure

To export the Registry key:

1. Start the REGEDIT program.
2. Move to the HKEY_CURRENT_USER\Software\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

You can configure the client to run multiple instances of Monarch, with each instance using a different model file.

About this task

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.

Procedure

For each model file that you want to run from the client:

1. Add an identifier to the Dlls string value under the ExternalDlls subkey.

Add a unique identifier for each model file.

The following graphic shows ExternalDlls subkey with multiple identifiers. (We added the identifiers Model1, Model2, and Model3.)

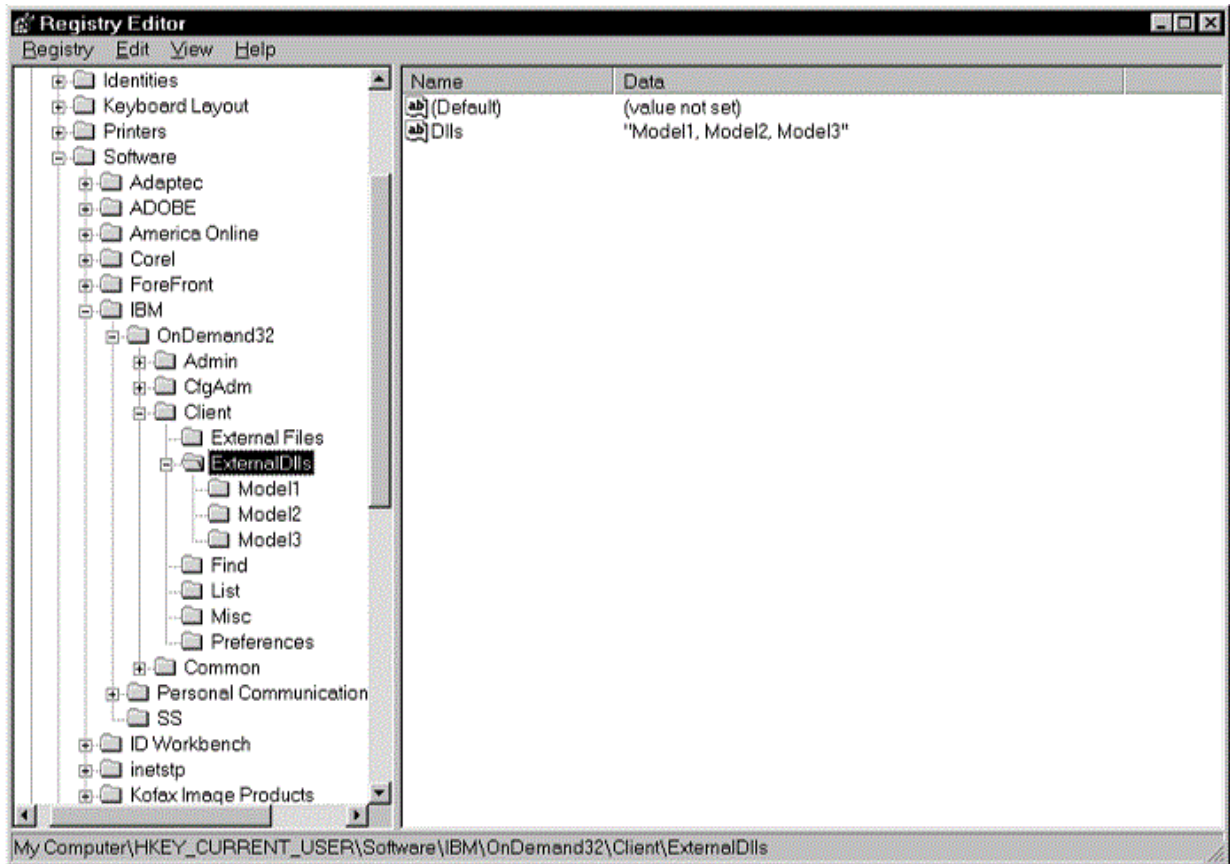


Figure 9: ExternalDlls subkey showing multiple values in Dlls

2. 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

What to do next

When you have finished defining the model files to the client, export the Registry key.

Configuring Setup

You can 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.

Procedure

To configure the client installation program to copy your Monarch model files to the user's workstation:

1. Copy the Monarch files to the custom directories.
2. Extract the latest version of the Content Manager OnDemand Windows client software from the ZIP file to a directory on the distribution server.

3. Add the custom directories to the Content Manager OnDemand Windows client installation directory tree on the distribution server.
4. Share the installation directory tree to make it available to your users.

Copying client software

You must copy the client software to the distribution server.

Procedure

To copy the client software to the distribution server:

1. Log on to the server with a user ID that has administrator permissions.
2. Extract the contents of the Content Manager OnDemand Windows 32-bit client ZIP file (odwin32.zip) or 64-bit client ZIP file (odwin64.zip) to a directory on the server. Use a file extraction method that preserves the ZIP contents directory and file structure on the server. For example, extract the files to the \ODCLIENT directory.

Results

When complete, the target directory (\ODCLIENT 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 Content Manager OnDemand Windows client directory tree on the distribution server.

Procedure

To add subdirectories to hold the Monarch files:

1. Create a custom directory under the Windows client directory. For example: `mkdir \odclient\ars32\custom` (for the 32-bit client) or `mkdir \odclient\ars64\custom` (for the 64-bit client). The following steps use `ars32` but you should use `ars64` if you are using the 64-bit client.
2. Add the `\odclient\ars32\custom\registry` directory.
The directory holds the Registry file (ODMonarch.Reg). When the user runs the Setup program from the distribution server, it imports this file into the Registry on the workstation.
3. Add the `\odclient\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 Content Manager OnDemand client software to the distribution server and creating the custom directories, copy the Monarch files to the subdirectories.

Procedure

To copy the Monarch files to the subdirectories:

1. Copy the `ODMonarch.Reg` file that you created in [“Exporting the Registry key” on page 199](#) to the `\ODCLIENT\ARS32\CUSTOM\REGISTRY` directory (for the 32-bit client) or the `\ODCLIENT\ARS64\CUSTOM\REGISTRY` directory (for the 64-bit client).
2. Copy your Monarch model files to the `\ODCLIENT\ARS32\CUSTOM\MONARCH` or `\ODCLIENT\ARS64\CUSTOM\MONARCH` directory.

What to do next

After you copy the Content Manager 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.

You 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 \ODCLIENT directory on the server and access all of the files in the \ODCLIENT\ARS32 or \ODCLIENT\ARS64 directory tree, you could assign the share name ODMONAR to the \ODCLIENT directory.

Running Setup

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.

Before you begin

Before your users run the Setup program from the distribution server to install the Content Manager OnDemand client and integrate Monarch with the client, they must have a copy of Monarch installed on their workstations.

Back up the Registry on the user's workstation before they run the Setup program.

Procedure

- 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.

Running Monarch from Content Manager OnDemand

After your users install the client from the distribution server, they should test that everything works correctly.

Procedure

1. Have them start the Content Manager OnDemand client.
2. Open a document and then start Monarch using the associated menu command or toolbar button.

Results

Monarch should start and load the document into a separate window.

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.

About this task

This section is of primary interest to administrators responsible for installing and configuring software products.

Sharing Content Manager OnDemand clients among multiple users

There are several ways that you can install Content Manager OnDemand client software.

About this task

Standard or Custom Install

Use to install the Content Manager 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 Content Manager 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 Content Manager 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 Content Manager OnDemand client on each user's workstation.

Distribution install

Use to copy the Content Manager 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.

Installation directories

The Content Manager OnDemand client CD-ROM is organized into directories for each of the clients.

The Content Manager OnDemand Windows client comes with one program for the standard, custom, multiple user, and node installs: `\client\windows\setup.exe`

Directory	Purpose
<ul style="list-style-type: none">• For 32-bit clients on 32-bit Windows and 64-bit clients on 64-bit Windows: <code>\Program Files\IBM\OnDemand Clients\V10.1</code>• For 32-bit clients on 64-bit Windows: <code>\Program Files(x86)\IBM\OnDemand Clients\V10.1</code>	Installation directory. For standard, custom, and multiple user installs, identifies a hard disk on the workstation. For node installs, identifies a network drive.

Table 7: Content Manager OnDemand Windows client installation directories (continued)

Directory	Purpose
<ul style="list-style-type: none"> • For 32-bit clients on 32-bit Windows and 64-bit clients on 64-bit Windows: \Program Files\IBM\OnDemand Clients\V10.1\bin • For 32-bit clients on 64-bit Windows: \Program Files(x86)\IBM\OnDemand Clients\V10.1\bin 	<p>Program directory. Identifies a hard disk on the workstation.</p>
<ul style="list-style-type: none"> • For 32-bit clients on 32-bit Windows and 64-bit clients on 64-bit Windows: \Program Files\IBM\OnDemand Clients\V10.1\Fonts • For 32-bit clients on 64-bit Windows: \Program Files(x86)\IBM\OnDemand Clients\V10.1\Fonts 	<p>AFP font file directory. For standard, custom, multiple user installs, identifies a hard disk on the workstation. For node installs, identifies a network drive.</p>

Chapter 17. Distributing user-defined files

This section provides information about how to configure the Content Manager OnDemand Windows client installation program to distribute user-defined files to your users.

About this task

This section is of primary interest to administrators responsible for installing and configuring software products. A distribution server is a network file server that contains a copy of the Content Manager OnDemand client software in a shared location. Other users on the network use the copy to run standard, custom and distribution installs.

Copy Content Manager OnDemand client software to the server

User-defined files can be stored in the Content Manager OnDemand Windows client directory tree on a distribution server.

Any files (and subdirectories of files) that you store there are copied along with the standard Content Manager 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.
- 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\OnDemand Clients\V10.1` 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.

Configuring the server to install user-defined files

All user-defined files must be stored in the custom subdirectory tree under the Content Manager OnDemand Windows client directory tree on the distribution server.

About this task

By default, the Windows client directory tree is `ars32` (for the 32-bit client) or `ars64` (for the 64-bit client). The Content Manager OnDemand Windows client directory tree on the distribution server is `\client\windows\ars32` or `\client\windows\ars64`. The examples use `ars32` but you should use `ars64` if you are using the 64-bit client.

Procedure

To configure the server to install user-defined files:

1. Create a custom directory under the Windows client directory.
For example: `mkdir \client\windows\ars32\custom`
2. Add one or more of the following subdirectories to the custom directory. The subdirectories you add 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 Content Manager OnDemand client files. The Setup program copies these files and subdirectories to the \Program Files\IBM\OnDemand Clients\V10.1 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\OnDemand Clients\V10.1\FONT directory on the user's workstation.

Storing user-defined files on the server

After you copy the Content Manager OnDemand client software to the distribution server and create the custom directories, you can store files in individual subdirectories.

About this task

For example, store DLL files that you want Content Manager OnDemand to run in the \CLIENT\WINDOWS\ARS32\CUSTOM\FILES directory (for the 32-bit client) or the \CLIENT\WINDOWS\ARS64\CUSTOM\FILES directory (for the 64-bit client).

Installing the Content Manager 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.

About this task

The next time that a user runs the Setup program from the server, the Setup program installs the Content Manager 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.

Chapter 18. Using response files

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 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.

Creating a response file

Response files commonly have an extension of .ISS and are found in the Windows directory.

Procedure

To create a response file:

- Run 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\arsin.iss
```

causes the Setup program to create the `ARSIN .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.

Procedure

To run the Setup program and specify a response file:

- 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\arsin.iss` causes the Setup program to install the software using the ARSIN .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 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\arsin.iss -f2c:\temp \arsin.log` causes the Setup program to write the log file ARSIN.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 parameter section.

About this task

Examine the value of the ResultCode parameter. The return code should be zero (0).

Using a response file to install Content Manager OnDemand software

In general, you would complete the following steps to prepare the Content Manager OnDemand Windows client software for installation using a response file and then install the software on other workstations connected to the network.

Procedure

To install the software on a workstation:

1. Run the Setup program with the -r option to create the response file and the -fl 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 -fl 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. IsDocHorzScrollRequired method

If the width of the document data exceeds the width of the control window, the result variable is set to a nonzero value; otherwise, to zero.

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

The displayed width of the document depends on the inherent width of its data, the type of data (that is, AFP versus Linedata), and the zoom factor.

Return Value

Refer to return codes.

See Also

ScrollDocHorz method

C/C ++

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.

```
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

```

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

```

Chapter 20. Mapping AFP Fonts

The Viewer needs to map the AFP fonts your document was created with to fonts that can be displayed on your workstation. For the Viewer to map the best matching outline fonts to display your AFP document, it needs to know certain characteristics about the fonts that were used to create your document.

Mapping AFP fonts to outline fonts is done with the IBM-supplied font definition files installed as part of the Viewer. These files are loaded into the FONT directory you specified when you installed the Viewer. You may edit them using any workstation editor. The shipped version of the font definition files maps the IBM Core Interchange (Latin only), compatibility, coordinated, Sonoran, and Data1 fonts for you.

If your document uses an AFP font whose family (familyname) is not installed on your workstation, you can use the ALIAS.FNT file (one of the font definition files installed with the Viewer) to substitute that font familyname with a different one. The ALIAS.FNT file remaps several of the AFP fonts to IBM Core Interchange fonts. If you have any outline fonts installed on your workstation, you may want to remove or comment out the font familyname substitutions in the ALIAS.FNT file. If you are using TrueType fonts, you must use the ALIAS.FNT file to map the font name.

The IBM Core Interchange fonts (shipped with AFP Workbench for Windows) are in Type 1 outline format. These fonts are delivered in three type families: Times New Roman, Helvetica, and Courier. Each type family is provided in these character set groups:

Latin

The Latin group is available in 4 typefaces: roman medium, roman bold, italic medium, and italic bold.

Symbols

The Symbols group is available in 2 typefaces: roman medium and roman bold.

Because the IBM Core Interchange fonts are also available for printing with Version 2 of PSF/MVS, PSF/VM and PSF/VSE; and PSF/6000 they help standardize fonts across applications and installations.

If you created your documents with only the unmodified IBM fonts, you will not need to remap fonts to use the Viewer.

When you need to map fonts

If you are using fonts that are not defined to the Viewer, if you have modified the IBM AFP fonts, or if you have created your own AFP fonts, you need to define those fonts in the font definition files in order for documents using those fonts to display correctly with the Viewer.

You need to define fonts:

- If you created a new coded font (or renamed one), you will need to define the coded font in the Coded Font file (ICODED.FNT or CODED.FNT).
- If you created a new character set, you have to define it in the Character Set Definition file (CSDEF.FNT).
- If you created a new code page, you have to define it in the Code Page Definition file (CPDEF.FNT).
- If you have created a new code page or modified a code page by moving characters, you have to create a new Code Page file (*.CNV).

If you only have modified an existing font component, you may not need to perform any of the above steps. For example, if you have only deleted code points in the code page, the font files supplied with the Viewer can be used.

Viewer files supplied for mapping fonts

The following types of files for font support are installed by default in the following subdirectories under the directory in which the Viewer was installed:

File	File Name	Subdirectory	Description
Coded Font files	ICODED.FNT CODED.FNT	\FONT	Specifies which AFP code page and AFP font character set make up the coded font.
Character Set definition file	CSDEF.FNT	\FONT	Defines AFP character set attributes, such as point size. It also maps the font character set to its font global identifier.
Code Page definition file	CPDEF.FNT	\FONT	Maps each AFP code page to a Code Page Map file for the Viewer to use.
Code Page Map file	cpgid.CNV	\FONT\CNV	Defines character identifier mappings. The .CNV files for ICU UCONV are used for character conversion.
Alias File	ALIAS.FNT	\FONT	Maps AFP font type families to Type 1 or TrueType outline font family names.

Note: CODED . FNT is an optional file. The CODED . FNT file is meant to contain coded fonts you have created.

Steps for mapping fonts to the Viewer

After reading the rest of this chapter to determine which font files you need to modify, follow these steps:

1. Gather the information needed to define the fonts in the font definition files. This information described in the following sections of this appendix.
2. Make backup copies of any of the following font definition files that you plan to modify:
 - CSDEF . FNT
 - CPDEF . FNT
 - ICODED . FNT
 - ALIAS . FNT

Note: Backup copies of these files should be made so that you have an unmodified copy in the event something happens to your modified copy that makes it inoperable.

3. Install any other outline fonts you are planning to use with the Viewer.

4. If you have created or modified a code page, use the ICU makeconv tool to build the code page map file.
5. If you have created a new character set, edit the CSDEF .FNT file and add your character set name in the [CHARSET] section. Specify the correct attributes for your font in the CSDEF .FNT. Add the appropriate information in the [FGID] section of the file if you are naming a new font global identifier.
6. If you have created a coded font, create or edit the CODED .FNT file and add your coded font.

Syntax rules for the Viewer font definition files

Syntax rules for the Viewer font definition files are as follows:

- A semicolon (;) in the first column of any of these files will cause the line to be treated as a comment statement and ignored.
- Section headers within files are enclosed in brackets [] and must not be removed or changed.
- All values are case insensitive.
- If a parameter value is invalid and a default value exists, it will be substituted.
- All parameters are positional.
- Blanks are allowed between parameter values.
- Each line must be less than 80 characters in length.

Coded Font file

The Coded Font file (ICODED.FNT) maps AFP coded fonts to their AFP character sets and AFP code pages.

Two Coded Font files can be used with the Viewer:

ICODED.FNT

This file contains definitions for approximately 2500 IBM-supplied coded fonts.

CODED.FNT

You can create this optional file to define a list of any coded fonts you have created. If you create a CODED .FNT file, you must place it in the FONT subdirectory.

If a CODED .FNT file exists in the FONT subdirectory, it is searched first for the coded fonts used in an AFP file. If the coded font name is not found in CODED .FNT or if CODED .FNT does not exist, only the Viewer-supplied ICODED .FNT file is searched.

Figure 13. Example of the partial contents of a CODED .FNT file

```
X?A155N2 = C?A155N1, T1DCDCFS
X?AE10 = C?S0AE10, T1S0AE10
X?GT10 = C?D0GT10, T1D0BASE
X?ST15 = C?D0ST15, T1D0BASE
X?A0770C = C?A07700, T1DCDCFS
X?A0770I = C?A07700, T1GI0361
X0T0550C = C0T05500, T1DCDCFS
```

Coded Font file rules

- A question mark (?) can be used as the wild-card character only for the second character in the coded font name and the character set name. This allows all the character rotations of the coded fonts to be handled with one entry for searching.

Note: A sequential search is performed for the coded font, and the first match is used (including the wild-card character).

- After the coded font name, the character set name must be listed first, followed by the code page name.
- The character set and code page must be separated by a comma.

Character Set Definition file

The Character Set Definition file specifies the character set attributes and font global identifier of the font. It is split into 2 sections, one for character sets [CHARSET] and one for font global identifiers [FGID].

Figure 14. The [CHARSET] section. Example of the character set [CHARSET] section in the Character Set Definition

```
[CHARSET]
;charset = fgid, height, width, strikeover, underline
C?H200A0=2304,110,73,0,0
C?H200D0=2304,140,93,0,0
C?N200B0=2308,120,80,0,0
C?4200B0=416,120,144,0,0
C?D0GT15=230,80,96,0,0
C?A155A0=33207,110,73,0,0
C?A175A0=33227,110,73,0,0
C?T055D0=4407,140,93,0,0
C?T17500=4555,100,67,0,0
C?T17560=4555,60,40,0,0
DEFAULT =2308,80,0
```

The first section identified by the section header [CHARSET] lists each AFP font character set and its corresponding attributes:

- Font global identifier (fgid)
- Font height
- Font width
- Strikeover
- Underline

Attribute	Possible Values	Shipped Default	Description
Fgid	IBM-defined FGID or your own defined FGID within this range: 3840 - 4095 or 65260 - 65534	2308	A unique value that identifies the type family, typeface, and sometimes the point size of the character set.
Height	1 - 990	80	The vertical size of the character set (minimal baseline-to-baseline value) expressed in tenths of a point. For example, a 9-point font would have a height of 90.

Table 9: Character Set Definition File Attribute Values for [CHARSET] (continued)

Attribute	Possible Values	Shipped Default	Description
Width	0 - 99 (currently ignored)	0	The average horizontal size of the characters in 1440th of an inch. Currently, 0 is always used because Windows® determines an appropriate font width based on the font height.
Strikeover	1 (means yes), 0 (means no)	0	A font whose characters all have a line, parallel to the character baseline, placed over the middle of the character.
Underline	1 (means yes), 0 (means no)	0	A font whose characters all have a line, parallel to the character baseline, placed under the character.

The second section, which is identified by the section header [FGID], lists each font global identifier and its corresponding attributes:

- Font type families
- Style
- Weight
- Italic

Figure 15. The [FGID] section. Example of the font global identifier [FGID] section in the Character Set Definition file (CSDEF.FNT).

```
[FGID]
;fgid = familyname, style, weight, italic
230=Gothic,MODERN,MED,0
416=Courier,MODERN,MED,0
2304=Helvetica,SWISS,MED,0
2308=TimesNewRoman,ROMAN,MED,0
4407=SonoranSerif,ROMAN,MED,0
4555=SonoranSerif,ROMAN,BOLD,1
33207=SonoranSansSerif,SWISS,MED,1
33227=SonoranSansSerif,SWISS,BOLD,1
```

Table 10: Character Set Definition File Attribute Values for [FGID]

Attribute	Description	Possible Values	Shipped Default
Familyname	An outline font name or an AFP type family name.	Any Adobe Type 1 font name or AFP type family name.	TimesNewRoman
Style	The same as a Windows "family". It is approximate to type family plus typeface style in AFP fonts.	SWISS, ROMAN, SCRIPT, MODERN, DISPLAY	ROMAN

Table 10: Character Set Definition File Attribute Values for [FGID] (continued)

Attribute	Description	Possible Values	Shipped Default
Weight	The degree of boldness of a typeface caused by different thickness of the strokes that form a graphic character.	LIGHT, MED, BOLD	MED
Italic	A font whose characters slant to the right.	1 (means yes), 0 (means no)	0

Note:

1. "Familyname" is the same as "type family" in AFP fonts and "typeface name" in Windows.
2. "Style" is the same as Windows "family" and is roughly equivalent to "typeface style" and "type family" in AFP fonts.
3. SWISS is a proportionally spaced font, without serifs.
4. ROMAN is a proportionally spaced font, with serifs.
5. SCRIPT is a fixed-pitch font that is designed to look like handwriting.
6. MODERN is a fixed-pitch font, with or without serifs.
7. DISPLAY is a decorative font.

Character Set Definition file rules

- Parameters must be separated by a comma. Table 10 and Table 11 list the possible values, and shipped default values for each parameter.
- In the [CHARSET] section of the file, only fgid and height (point size) are required.
- In the [FGID] section of the file, only the type familyname and style are required.
- A question mark (?) can be used as the wildcard character only for the second character in the character set name. This allows all the character rotations of the coded fonts to be handled with one entry while searching.

Note: A sequential search is performed for the character set, and the first match is used (including the wildcard character).

- The [CHARSET] section must come before the [FGID] section.
- You can set a default character set. The default character set that is defined in the file must be the last entry in the [CHARSET] section.
- If you add your own AFP font character set to the [CHARSET] section, you must assign it a font global identifier. Font global identifiers that you create must be in the ranges of 3840 - 4095 or 65260 - 65534. If the new character set has the same familyname, style, weight, and italic attributes as an existing character set, you may use the same font global identifier; otherwise, you must add a unique font global identifier to the [FGID] section.

Code Page Definition file

The Code Page Definition file maps the IBM® AFP™ code page name to its code page global identifier (CPGID) and to a Windows® character set.

The section header [CODEPG] is followed by a list of AFP code pages and their parameters. The first parameter in each line is the AFP code page global identifier that maps to a Code Page Map file. (See Code Page Map files for more information about mapping code pages.) The second parameter is the Windows character set that you decide is the best match for your AFP code page. The last line gives the default parameter values to be used when a default is required.

Figure 16. Example of the Code Page Definition file (CPDEF.FNT) contents

```
[CODEPG]
;codepage = cpgid
T1DCDCFS=1003
T1DEBASE=2058
T1D0BASE=2063
T1D0GP12=2085
T1GI0395=2079
T1GPI363=2066
T1V10037=37
T1V10273=273
T1000290=290
T1000310=310
T1000423=423
T1000905=905
DEFAULT =361
```

You can use the NONSTD value to allow the name and size of a font to fully describe the logical font. If the specified font name does not exist, a font from any character set can be substituted for this specified font.

Table 11: Code Page Definition file attribute values

Attribute	Possible Values	Shipped Default
Code Page Global Identifier	IBM-defined CPGID or your own defined CPGID within this range: 65280 - 65534	361

Code Page Definition file rules

- Parameters must be separated by a comma. Table 12 lists the possible values and shipped default values for each parameter.
- Only the first parameter (code page identifier) is required.
- If you create your own code page, you must assign it a unique code page identifier. Leading zeros are not valid. (You may use an IBM® code page global identifier but only if the character-to-hexadecimal code mapping is the same for your code page.)
- You can set a default code page. The default code page that is set within the file must be the last entry in the file.

Alias file

The Alias file contains two sections: one section for font familyname aliases [FONT] and one section for character identifier aliases [CHARID].

The first section, identified by the section header [FONT], lists the font familyname aliases. Font familyname aliases allow you to change all of the requested instances of a font familyname (as defined in the Character Set Definition file) to another font familyname. For example, this file is used to change all requests for the SonoranSerif font (which may not exist on the workstation) to requests for the TimesNewRoman font (which is one of the core fonts shipped with the Viewer).

Adobe provides Type 1 support, however, TrueType fonts can be used with the Viewer. As a backup, a second font (TrueType) can be specified after the Type 1 font name. If the Type 1 font is not found, the TrueType font will be used to display your document.

Note: Be aware that font familyname remapping, especially to TrueType fonts, can cause some misalignment of text characters since the display font is not the same as the font used to create the AFP document. Remapping of one font familyname to a different font familyname with very different

characteristics (such as STYLE) may mean a matching font cannot be found. You will receive an error message if either font substitute cannot be found.

The [FONT] section. This example of the [FONT] section is from the Alias file (ALIAS.FNT).

```
[FONT]
; ***** Requested font = Type 1 font, TrueType font *****
Book=TimesNewRoman,Times New Roman
CourierOverstrike=Courier,Courier New
SonoranSerif=TimesNewRoman,Times New Roman
SonoranSansSerif=Helvetica,Arial
Text=Courier,Courier New
```

The second section, identified by the section header [CHARID], lists the character identifier aliases. Character identifier aliases (also known as glyph identifiers) allow you to change all of the requested instances of a character to another character. For example, since the Windows ANSI character set does not contain the ligature (LF510000), it is not mapped to a character in the code page map files. Instead, it is mapped to NOMATCH 00. If you want to map all occurrences of LF510000 – NOMATCH pair to a lower case f, you could do this in the [CHARID] section of the ALIAS.FNT file with the following entry:

```
LF510000=LF010000
```

If you want to change one specific character for one specific code page, then you may remap the character on that code page to another character.

The Alias file is checked only when a NOMATCH 00 is found in a character mapping.

Note: Using the Alias file for more than a few character substitutions is not recommended as program performance is affected. If many character substitutions are needed, it is better to make changes directly to the mappings in the Code Page Map files that you are using.

The [CHARID] section. This example of the [CHARID] section is from the Alias file (ALIAS.FNT).

```
[CHARID]
LF510000=LF010000
SA000000=SP320000, SP100000
```

Alias file rules

- For family name aliases, all requests for the first family name in the Character Set Definition file have the second family name substituted for them. If the second family name is not found, the TrueType font (the third family name) is requested.
- Only two family name substitutes per line are allowed (to the right of the equal sign), and they must be separated by a comma.
- If multiple mappings are listed in the file for the same family name, only the first match is used.
- The Alias file is processed sequentially and is not chained (for example, if "Century Schoolbook" is set equal to "Times", and "Times" is set equal to "TimesNewRoman", "Century Schoolbook" will not be set to "TimesNewRoman").
- Blanks in family names are treated as characters (for example, "Times New Roman" is not the same font as "TimesNewRoman").
- The [CHARID] section of the Alias file is only used if the second character identifier is NOMATCH 00.
- The character identifier that you want modified in the [CHARID] section must be followed by an equal sign and the character identifier to which it is to be changed. A character remap occurs when the modified character identifier (the character to the left of the equal sign in the [CHARID] section) matches the first character identifier of a non-matching pair in the Code Page Map file.
- Several character identifiers (substitute char id) might be listed to the right of the equal sign separated by commas. The first substitute character identifier is substituted for the modified character identifier unless it does not exist in the Windows® font. If it does not exist, then the next substitute character

identifier is used. If none of the substitute character identifiers exist, the undefined code point is used. If you want to see the contents of the Windows character sets, see the .WCP files in the FONT directory.

- A maximum of four substitute character identifiers are allowed.

Support for TrueType Fonts

TrueType Fonts

You can use TrueType fonts to display your documents. To request a specific TrueType font, use the second font substitution family name in the ALIAS.FNT file as described in Alias file.

TrueType Font Substitution Problems

Make sure that the TrueType font that you requested is installed on your workstation. Font substitutions that occur when fonts are not available might cause unexpected results displaying your files. For example, Courier New is requested in the ALIAS.FNT file and is available with one version of Windows but is not available on another; the same document can appear (view) differently on the two systems (however, the font can be installed on the latest Windows system).

Chapter 21. 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 OnDemand Administrator 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 22. ICU converters

You can create and maintain custom ICU converters to use with the Content Manager OnDemand Client.

To create custom ICU 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 package from the ICU download page.
2. Add the bin directory from the extracted ICU Binary package in your PATH environment variable.
3. Create the custom ICU data directory. From the Content Manager OnDemand Client locale directory, for example, C:\Program Files(x86)\IBM\OnDemand Clients\V10.1\Locale, create a subdirectory with the same name as the ICU data file (for example, icudt53l).
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 converter that was shipped with the Content Manager OnDemand Client, and it is not available from ICU, download the `afpucm36.zip` from the following directory:
<ftp://service.software.ibm.com/software/ondemand/utils/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 the new converter that you want. 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 Content Manager OnDemand Client and rendering the report.

If you have custom ICU converters and want to migrate them to Content Manager OnDemand Client, 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 package. 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 23. Troubleshooting

StoreDoc() API returns error code 2

Symptoms

Your StoreDoc () API returns an error code of 2.

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.

Resolving the problem

Make sure that both attributes of the application group is correctly set.

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 `npoa.f32.dll`, instead, you should use the Content Manager 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 Content Manager 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

You can define the global variables used by the sample program.

This section defines the global variables used by the sample program.

```
Option Explicit
Global Const apptitle = "Content Manager 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 = "ARSGUI.EXE"    'Default Client exe

'Default Client parms
' 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 Content Manager OnDemand Windows client by using the DDE interface.

Some important things to note:

- The commands sent to the Content Manager OnDemand client are not DDE EXECUTE's, they are all DDE REQUEST's. This is important because sending DDE EXECUTE's to the Content Manager OnDemand client results in an error.
- It is important to start the Content Manager OnDemand client with at least the /I option. This enables the DDE interface of the Content Manager 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 Content Manager 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 Content Manager OnDemand client window has 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 Content Manager 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 to 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 Content Manager 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 cmdnd 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 Content Manager 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
    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 Content Manager 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
Else
    '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 Content Manager 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 Content Manager 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 Content Manager OnDemand client with the Disable anticipation
(/V) parameter.
        Call fncDDElink(arstopic, "EXIT", linktype, 3000)
        Call fncDispStatus("Logon to client...failed.")
        End
    End If
    Call fncDispStatus("Logon to Client...done.")
End Sub

'Open up an Content Manager 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 Content Manager 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 Content Manager 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
End Sub

'GUI control used to display customer information.
'We do not obtain the customer information from out of Content Manager
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 Content Manager 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
  cmdViewStmnt1.Enabled = False
  cmdViewStmnt2.Enabled = False
  cmdViewStmnt3.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 nonzero 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 Content Manager OnDemandHandle As Integer

    Call fncDispStatus("Program ending...")

    'Determine if Content Manager OnDemand is loaded
    Content Manager OnDemandHandle = GetModuleHandle(arsgui)
    'If not loaded, then quit, else shutdown
    If Content Manager 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 Content Manager OnDemandHandle As Integer

    Call fncDispStatus("Program ending...")

    'Determine if Content Manager OnDemand is loaded
    Content Manager OnDemandHandle = GetModuleHandle(arsgui)
    'If not loaded, then quit, else shutdown
    If Content Manager 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
    End If
End Function

```

```

        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

    '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 ini file 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
        End If
    End If

```

```
        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 Content Manager 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 <winspool.h>

#include "vcdde32.h"
#include "MainDlg.h"
#include "arsddeex.h" // Shipped with Content Manager 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 HDDEDATA hDdeData, hDdeResult;

HDDEDATA FAR PASCAL DdeCallback ( UINT      iType,
                                  UINT      iFmt,
                                  HCONV     hConv,
                                  HSZ       hsz1,
                                  HSZ       hsz2,
                                  HDDEDATA  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_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
{
    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 Content Manager OnDemand client

// /I - Enable DDE Interface
// /W - Window placement (N = hidden)
// /V - Disable anticipation
// /B - Disable User Confirmation
strcpy( cmd, "g:\\ars32\\arsgui.exe /I /W N /V /B /1 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_CLIENTONLY,
            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 ARSGUI." );

    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 = atol( 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)l );
            }
        }
        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_ICONERASEBKGND, (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::OnDblclckDoclist()
{
    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 Content Manager 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 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 VertScrollOld As Integer
Global HorzScrollOld 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

```

```

VertScrollOld = 0
HorzScrollOld = 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
    End If
End Function

```

```

    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 nonzero 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
        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)
        ArsOle.CloseFolder
        ArsOle.Logoff
        End
    End If

    rc = ArsOle.SearchFolder(0)
    If rc <> ARS_OLE_RC_SUCCESS Then
        pnlStatus.Caption = ""
        MsgBox "Search folder failed; rc = " + Str(rc)
        ArsOle.CloseFolder
        ArsOle.Logoff
        End
    End If

    rc = ArsOle.GetNumDocsInList(numdocs)

    rc = ArsOle.GetDocDisplayValue(numdocs - 1, 0, temp)
    pnlStmtDate1.Caption = temp
    pnlDatePaid1.Caption = fncParseDate(temp)
    rc = ArsOle.GetDocDisplayValue(numdocs - 2, 0, temp)
    pnlStmtDate2.Caption = temp
    pnlDatePaid2.Caption = fncParseDate(temp)
    rc = ArsOle.GetDocDisplayValue(numdocs - 3, 0, temp)
    pnlStmtDate3.Caption = temp
    pnlDatePaid3.Caption = fncParseDate(temp)

    rc = ArsOle.GetDocDisplayValue(numdocs - 1, 3, temp)
    pnlBalance1.Caption = temp
    rc = ArsOle.GetDocDisplayValue(numdocs - 2, 3, temp)
    pnlBalance2.Caption = temp
    rc = ArsOle.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
    '    ArsOle.CloseDoc
    'End If
    'ArsOle.CloseFolder
    'ArsOle.Logoff
    End
End Sub

Private Sub cmdViewStmt1_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 - 1, "", 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)
    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 = ArsOle.SetDocZoom(DocZoom, horzPos, vertPos)
End Sub

Private Sub Form_Unload(Cancel As Integer)
    If OpenDoc Then
        ArsOle.CloseDoc
    End If
    ArsOle.CloseFolder
    ArsOle.Logoff
    End
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 - HorzScrollOld
    If Diff = hscrollDoc.LargeChange Then
        ScrollCode = ARS_OLE_SCROLL_PAGERIGHT
        rc = ArsOle.ScrollDocHorz(ScrollCode, NewPos)
        hscrollDoc.Value = NewPos
    ElseIf Diff = -hscrollDoc.LargeChange Then
        ScrollCode = ARS_OLE_SCROLL_PAGELEFT
        rc = ArsOle.ScrollDocHorz(ScrollCode, NewPos)
    End If
End Sub

```

```

        hscrollDoc.Value = NewPos
    ElseIf Diff = hscrollDoc.SmallChange Then
        ScrollCode = ARS_OLE_SCROLL_LINERIGHT
        rc = ArsOle.ScrollDocHorz(ScrollCode, NewPos)
        hscrollDoc.Value = NewPos
    ElseIf Diff = -hscrollDoc.SmallChange Then
        ScrollCode = ARS_OLE_SCROLL_LINELEFT
        rc = ArsOle.ScrollDocHorz(ScrollCode, NewPos)
        hscrollDoc.Value = NewPos
    Else
        ScrollCode = ARS_OLE_SCROLL_THUMBPOSITION
        NewPos = hscrollDoc.Value
        rc = ArsOle.ScrollDocHorz(ScrollCode, NewPos)
        HorzScrollOld = hscrollDoc.Value
    End If

    HorzScrollOld = 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 - VertScrollOld
    If Diff = vscrollDoc.LargeChange Then
        ScrollCode = ARS_OLE_SCROLL_PAGEDOWN
        rc = ArsOle.ScrollDocVert(ScrollCode, NewPos)
        VertScrollOld = NewPos
        vscrollDoc.Value = NewPos
    ElseIf Diff = -vscrollDoc.LargeChange Then
        ScrollCode = ARS_OLE_SCROLL_PAGEUP
        rc = ArsOle.ScrollDocVert(ScrollCode, NewPos)
        VertScrollOld = NewPos
        vscrollDoc.Value = NewPos
    ElseIf Diff = vscrollDoc.SmallChange Then
        ScrollCode = ARS_OLE_SCROLL_LINEDOWN
        rc = ArsOle.ScrollDocVert(ScrollCode, NewPos)
        VertScrollOld = NewPos
        vscrollDoc.Value = NewPos
    ElseIf Diff = -vscrollDoc.SmallChange Then
        ScrollCode = ARS_OLE_SCROLL_LINEUP
        rc = ArsOle.ScrollDocVert(ScrollCode, NewPos)
        VertScrollOld = NewPos
        vscrollDoc.Value = NewPos
    Else
        ScrollCode = ARS_OLE_SCROLL_THUMBPOSITION
        NewPos = vscrollDoc.Value
        rc = ArsOle.ScrollDocVert(ScrollCode, NewPos)
        VertScrollOld = 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 Content Manager 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_GRAY, "Gray" },
  { 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, OnDblclkDoclist)
    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_ARCTRL, 4, OnFolderSearchCompletedArctrl,
VTS_NONE)
    ON_EVENT(CMainDlg, IDC_ARCTRL, 3, OnDocOpenedArctrl, VTS_NONE)
    ON_EVENT(CMainDlg, IDC_ARCTRL, 1, OnDocClosedArctrl, 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 Content Manager 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, (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::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::OnDblclkDoclist()
{
    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_ARSCtrl );

    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_ARSCtrl );

    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_ARSCtrl );
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.

This material may be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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 websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites 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" at <http://www.ibm.com/legal/copytrade.shtml>.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Privacy policy considerations

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, See IBM’s Privacy Policy at <http://www.ibm.com/privacy> and IBM’s Online Privacy Statement at <http://www.ibm.com/privacy/details> the section entitled “Cookies, Web Beacons and Other Technologies” and the “IBM Software Products and Software-as-a-Service Privacy Statement” at <http://www.ibm.com/software/info/product-privacy>.

Index

A

ACTIVATE_DOC command [131](#)
ACTIVATE_FOLDER command [132](#)
AFP files
 navigation plug-ins [227](#)
 workbench [227](#)
ANNOTATE_DOC command [132](#)
application groups
 defining [190](#)
applications
 defining [191](#)
 external [173](#)
ARRANGE_DOCS command [133](#)
ARS_OLE_SCROLL_RANGE [1](#)
ARSGUI.CFG file [189](#)
AUDIT
 DAF file [189](#)
audit facility
 documents [189](#)

C

Change Password parameter [119](#)
CHANGE_PASSWORD command [134](#)
CID [207](#)
Citrix server install [203](#)
CLEAR_FIELDS command [135](#)
client
 distributing user-defined files [205](#)
 installing on a network [203](#)
 installs [206](#)
 integrating with Monarch [195](#)
 network installation [203](#)
 user-defined files [205](#)
client software
 copying [201](#)
CLOSE_ALL_DOCS command [136](#)
CLOSE_ALL_FOLDERS command [136](#)
CLOSE_DOC command [137](#)
CLOSE_FOLDER command [137](#)
Code Page Override /A [119](#)
codes [171](#)
command line
 overview [119](#)
command line reference
 parameter syntax [119](#)
 parameters
 Change Password/C [119](#)
 Disable Anticipation/V [120](#)
 Disable Close Folder/Z [120](#)
 Disable Exit — /K [120](#)
 Disable Logoff and Password Change/X [120](#)
 Disable Update Servers/Y [120](#)
 Disable User Confirmation/B [120](#)
 Enable DDE Interface /I [121](#)
 Folder Name/F [121](#)

command line reference (*continued*)
 parameters (*continued*)
 Free Memory when Folder Closed/Q [121](#)
 Language ID Override — /L [121](#)
 Logon Password/P [121](#)
 Logon Server Name/S [122](#)
 Logon User ID/U [122](#)
 Maximum Open Folders/O [122](#)
 Product Title/T [122](#)
 Window Placement/W [122](#)
 Starting Content Manager OnDemand client [119](#)
commands
 DDE [131](#)
configuration, installation, and distribution [207](#)
configure the client [195](#)
Content Manager OnDemand customization overview [117](#)
Content Manager OnDemand DDE commands
 ACTIVATE_DOC [131](#)
 ACTIVATE_FOLDER [132](#)
 ANNOTATE_DOC [132](#)
 ARRANGE_DOCS [133](#)
 CHANGE_PASSWORD [134](#)
 CLEAR_FIELDS [135](#)
 CLOSE_ALL_DOCS [136](#)
 CLOSE_ALL_FOLDERS [136](#)
 CLOSE_DOC [137](#)
 CLOSE_FOLDER [137](#)
 COPY_DOC_PAGES [138](#)
 DELETE_DOC [139](#)
 DESELECT_DOC [140](#)
 DISABLE_SWITCH [141](#)
 ENABLE_SWITCH [141](#)
 EXIT [142](#)
 GET_DISPLAY_FIELDS [143](#)
 GET_DOC_VALUES [143](#)
 GET_FOLDER_FIELDS [144](#)
 GET_FOLDERS [145](#)
 GET_NUM_DOC_PAGES [146](#)
 GET_NUM_DOCS_IN_LIST [146](#)
 GET_PRINTERS [147](#)
 GET_QUERY_FIELDS [148](#)
 GET_SERVERS [148](#)
 LOGOFF [149](#)
 LOGON [150](#)
 OPEN_DOC [151](#)
 OPEN_FOLDER [152](#)
 PRINT_DOC [154](#)
 RESTORE_DEFAULTS [156](#)
 RETRIEVE_DOC [157](#)
 return codes [169](#)
 SEARCH_FOLDER [158](#)
 SELECT_DOC [158](#)
 SET_FIELD_DATA [159](#)
 SET_FOCUS [161](#)
 SET_HELP_PATH [161](#)
 SET_USER_MSG_MODE [162](#)
 SHOW_WINDOW [163](#)

Content Manager OnDemand DDE commands (*continued*)

STORE_DOC [164](#)
UPDATE_DOC [166](#)
control access
DAC [191](#)
Control IDs [1](#)
copy OnDemand client to server [205](#)
COPY_DOC_PAGES command [138](#)
custom install [203](#)
customization overview [117](#)

D

DAF
auditing documents [192](#)
control access [191](#)
DAF control files [189](#)
DAF file
AUDIT [189](#)
data mining
with Monarch [195](#)
DDE commands
ACTIVATE_DOC [131](#)
ACTIVATE_FOLDER [132](#)
ANNOTATE_DOC [132](#)
ARRANGE_DOCS [133](#)
CHANGE_PASSWORD [134](#)
CLEAR_FIELDS [135](#)
CLOSE_ALL_DOCS [136](#)
CLOSE_ALL_FOLDERS [136](#)
CLOSE_DOC [137](#)
CLOSE_FOLDER [137](#)
COPY_DOC_PAGES [138](#)
DELETE_DOC [139](#)
DESELECT_DOC [140](#)
DISABLE_SWITCH [141](#)
ENABLE_SWITCH [141](#)
EXIT [142](#)
GET_DISPLAY_FIELDS [143](#)
GET_DOC_VALUES [143](#)
GET_FOLDER_FIELDS [144](#)
GET_FOLDERS [145](#)
GET_NUM_DOC_PAGES [146](#)
GET_NUM_DOCS_IN_LIST [146](#)
GET_PRINTERS [147](#)
GET_QUERY_FIELDS [148](#)
GET_SERVERS [148](#)
LOGOFF [149](#)
LOGON [150](#)
OPEN_DOC [151](#)
OPEN_FOLDER [152](#)
PRINT_DOC [154](#)
RESTORE_DEFAULTS [156](#)
RETRIEVE_DOC [157](#)
return codes [169](#)
SEARCH_FOLDER [158](#)
SELECT_DOC [158](#)
SET_FIELD_DATA [159](#)
SET_FOCUS [161](#)
SET_HELP_PATH [161](#)
SET_USER_MSG_MODE [162](#)
SHOW_WINDOW [163](#)
STORE_DOC [164](#)
UPDATE_DOC [166](#)

DDE sample program [229](#)
DDE sample programs [245](#)
DDEML
termination [127](#)
transactions [128](#)
DDEML advise loops [171](#)
define reports [190](#)
defining application groups [190](#)
defining applications [191](#)
defining folders [191](#)
DELETE_DOC command [139](#)
DESELECT_DOC command [140](#)
directories [203](#)
Disable Anticipation parameter [120](#)
Disable Close Folder parameter [120](#)
Disable Exit parameter [120](#)
Disable Logoff and Password Change parameter [120](#)
Disable Update Servers parameter [120](#)
Disable User Confirmation parameter [120](#)
DISABLE_SWITCH command [141](#)
distributing user-defined files [205](#)
distribution install [203](#)
distribution server [195](#), [203](#)
Document Audit Facility [189](#)
documents
view multiple documents [1](#)
viewing [1](#), [181](#)
Dynamic Data Exchange (DDE) interface reference [125](#)
dynamic link libraries [173](#)

E

Enable DDE Interface parameter [121](#)
ENABLE_SWITCH command [141](#)
events [171](#)
EXIT command [142](#)
external applications [173](#)

F

folder
DAF file [190](#)
Folder Name parameter [121](#)
folders
defining [191](#)
Free Memory when Folder Closed parameter [121](#)
full report browse [223](#)
full report reprint [223](#)

G

GET_DISPLAY_FIELDS command [143](#)
GET_DOC_VALUES command [143](#)
GET_FOLDER_FIELDS command [144](#)
GET_FOLDERS command [145](#)
GET_NUM_DOC_PAGES command [146](#)
GET_NUM_DOCS_IN_LIST command [146](#)
GET_PRINTERS command [147](#)
GET_QUERY_FIELDS command [148](#)
GET_SERVERS command [148](#)
global variables [229](#), [245](#)

H

header files [1](#)

I

ICU converters

converters [225](#)

information files

programs [187](#)

install

Citrix server [203](#)

custom [203](#)

distribution [203](#)

local [203](#)

standard [203](#)

typical [203](#)

verifying [208](#)

installation

creating and using response files [207](#)

response files [207](#)

installation directories [203](#)

installing

clients on a network [203](#)

Monarch [195](#)

network file server [203](#)

user-defined files [205](#)

installing the client [206](#)

integration with Monarch [195](#)

interface reference for DDE [125](#)

invocation [126](#)

invoking client from other applications [125](#)

L

libraries

dynamic links [173](#)

local install [203](#)

LOGOFF command [149](#)

LOGON command [150](#)

Logon Password parameter [121](#)

Logon Server Name parameter [122](#)

Logon User ID parameter [122](#)

M

Maximum Open Folders parameter [122](#)

methods

AboutBox [3](#)

ActivateFolder [3](#)

AnnotateDoc [4](#)

CancelOperation [5](#)

ChangePassword [6](#)

ClearFolderSearchFields [7](#)

CloseAllFolders [8](#)

CloseDoc [9](#)

CloseFolder [10](#)

CopyBitmap [10](#)

CopyText [11](#)

DeleteDoc [12](#)

FindStringInDoc [13](#)

GetAnnotationForDoc [15](#)

GetAnnotationStatus [16](#)

methods (*continued*)

GetControlId [17](#)

GetDocAnnotation [18](#)

GetDocBackgroundColor [20](#)

GetDocCurrentPage [21](#)

GetDocDisplayValue [22](#)

GetDocDisplayValues [23](#)

GetDocImageColor [25](#)

GetDocImageIntensity [26](#)

GetDocNumPages [28](#)

GetDocRotation [29](#)

GetDocScrollPositions [30](#)

GetDocType [31](#)

GetDocZoom [32](#)

GetFolderDisplayFieldName [33](#)

GetFolderDisplayFieldNames [34](#)

GetFolderFieldName [36](#)

GetFolderFieldNames [37](#)

GetFolderName [38](#)

GetFolderNames [39](#)

GetFolderSearchFieldNames [42](#)

GetNumDocAnnotations [43](#)

GetNumDocsInList [45](#)

GetNumFolderDisplayFields [47](#)

GetNumFolderFields [49](#)

GetNumFolders [51](#)

GetNumFolderSearchFields [53](#)

GetNumServerPrinters [55](#)

GetNumServers [56](#)

GetResourceCacheMode [57](#)

GetServerName [58](#)

GetServerNames [59](#)

GetServerPrinter [61](#)

GetServerPrinterInfo [62](#)

GetStoreDocInvalidFieldNum [63](#)

GetTypeForDoc [64](#)

IsDocHorzScrollRequired [65](#), [209](#)

Logoff [68](#)

Logon [69](#)

OnSysColorChange [70](#)

OpenDoc [71](#)

OpenFolder [74](#)

PrintDoc [75](#)

RetrieveDoc [77](#)

ScrollDocHorz [79](#)

ScrollDocVert [82](#)

SearchFolder [85](#)

SetDefaultFolderSearchFields [87](#)

SetDocBackgroundColor [88](#)

SetDocCurrentPage [89](#)

SetDocImageColor [90](#)

SetDocImageIntensity [91](#)

SetDocRotation [92](#)

SetDocZoom [93](#)

SetFolderCloseMemoryRelease [95](#)

SetFolderSearchFieldData [95](#)

SetLogonReturnOnFailure [98](#)

SetResourceCacheMode [99](#)

SetRightButtonMenu [100](#)

SetSelectionMode [102](#)

SetServerPrinterData [103](#)

SeyUserMessageMode [105](#)

ShowFolder [106](#)

StoreDoc [108](#)

methods (*continued*)
 UndoFind [110](#)
 UpdateDoc [111](#)
 WasOperationCancelled [112](#)
methodsGetFolderSearchFieldName [40](#)
methodsShowWaitCursor [107](#)
Monarch
 configuring [200](#)
 copying files [201](#)
 integration with [195](#)
 registry keys [200](#)
 running OnDemand [202](#)
 running setups [202](#)
 using multiple files [199](#)
Monarch DLLs [196](#)

N

network file server [203](#)
network installation [203](#)

O

OLE controls [3](#)
OLE Controls
 return codes [2](#)
OLE events
 AreaDeselected [116](#)
 AreaSelected [116](#)
 DocClosed [115](#)
 DocOpened [115](#)
 FolderSearchCompleted [115](#)
 UserCommand(long CommandID) [116](#)
OLE eventsFolderClosed [115](#)
OLE sample programs [255](#), [265](#)
OPEN_DOC command [151](#)
OPEN_FOLDER command [152](#)

P

parameter syntax [119](#)
parameters [119](#)
PRINT_DOC command [154](#)
Product Title parameter [122](#)
Program Information File [187](#)

R

reference workstation [195](#)
registries
 modifying the client [193](#)
registry key
 adding [196](#)
 external DLLs [196](#)
registry keys
 exporting [199](#)
report mining
 with Monarch [195](#)
reports
 browse [223](#)
 define [190](#)
 reprint [223](#)
response files

response files (*continued*)
 creating [207](#)
 formatting [207](#)
 installing software [207](#)
 installing the software [208](#)
RESTORE_DEFAULTS command [156](#)
RETRIEVE_DOC command [157](#)
return codes [169](#)

S

sample programs
 entry points [230](#)
 global variables [229](#), [245](#)
SEARCH_FOLDER command [158](#)
SELECT_DOC command [158](#)
servers
 copying clients [205](#)
 storing files [206](#)
SET_FIELD_DATA command [159](#)
SET_FOCUS command [161](#)
SET_HELP_PATH command [161](#)
SET_USER_MSG_MODE command [162](#)
SHOW_WINDOW command [163](#)
standard install [203](#)
STORE_DOC command [164](#)
subdirectories
 adding [201](#), [205](#)
subkeys [196](#)
syntax rules for command line parameters [119](#)

T

terminating the DDE conversation [127](#)
transactions
 DDEML [128](#)
 example [128](#)
troubleshooting [227](#)
typical install [203](#)

U

UPDATE_DOC command [166](#)
user-defined files
 storing [206](#)

V

Visual Basic DDE sample program [229](#)
Visual Basic OLE sample programs [255](#)
Visual C++ DDE sample programs [245](#)
Visual C++ OLE sample programs [265](#)

W

Window Placement parameter [122](#)
Windows client
 CID [207](#)
 configuration, installation, and distribution [207](#)
 creating and using response files [207](#)
 distributing user-defined files [205](#)
 installing on a network [203](#)
 integrating with Monarch [195](#)

Windows client (*continued*)
network installation [203](#)
response files [207](#)
user-defined files [205](#)



SC19-3357-02

