

IBM Content Manager Client for Windows



Client for Windows Programming Reference

Version 8 Release 2

IBM Content Manager Client for Windows



Client for Windows Programming Reference

Version 8 Release 2

Note

Before using this information and the product it supports, read the information in Appendix B, "Notices", on page 75.

Second Edition (March 2003)

This edition applies to Version 8 Release 2 of IBM Content Manager (product number 5697-G28) and to all subsequent releases and modifications until otherwise indicated in new editions.

© **Copyright International Business Machines Corporation 1993, 2003. All rights reserved.**

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

Contents

About this reference	v
Who should use this reference	v
How this reference is organized	v
Style conventions	v
Key concepts	vi
Where to find more information	vi
Information included in your product package	vi
Support available on the Web	vii
How to send your comments	viii

Chapter 1. Using the Automation

interfaces	1
Background information about Automation	1
Properties	1
Methods	2
Content Manager Client for Windows objects	2
Application object	2
Document object	2
Documents collection object	3
Error object	3
Image object	3
Item object	3
Items collection object	3
Programming tips	3
Releasing objects	4
Handling errors	4
Property and argument types	4
Sample Visual Basic program	5

Chapter 2. Automation interface

properties and methods	7
Application object	7
Application object properties	7
Application object methods	9
Document object	20
Document object properties	20
Document object methods	21
Documents collection object	24
Documents collection object properties	24
Documents collection object methods	25
Error object	27
Error object properties	27
Image object	29
Image object properties	29
Image object methods	29
Item object	30
Item object properties	30
Item object methods	33
Items collection object	41
Items collection object properties	41
Items collection object methods	41

Chapter 3. Client for Windows user exit

routines	43
-----------------	-----------

Alternate search	43
Alternate search purpose	43
Alternate search parameters	43
Internal representation	44
Return values	44
Comments	44
Query sort	44
Query sort purpose	44
Query sort parameters	45
Internal representation	45
Return values	48
Comments	48
Save record	48
Save record purpose	49
Save record parameters	49
Internal representation	49
Return values	51
Comments	51
Change SMS	52
Change SMS purpose	52
Change SMS parameters	52
Internal representation	52
Return values	55
Comments	55
Get attribute value list	55
Get attribute value list purpose	55
Get attribute value list parameters	55
Return values	57
Comments	57
Get value list length	57
Get value list length purpose	57
Get value list length parameters	57
Return values	58
Comments	58
Add items to actions menu	58
Add items to actions purpose	58
Add items to actions parameters	58
Internal representation	58
Return values	59
Comments	59
Enable or disable menu items	59
Enable or disable purpose	59
Enable or disable parameters	59
Internal representation	60
Return values	61
Comments	61
User option function	61
User option function purpose	61
User option function parameters	62
Internal representation	62
Return values	63
Comments	63

	Appendix A. OLE API mappings from		Item object	70
	Content Manager version 7 to Content		Items object	73
	Manager version 8.			
	Application object		Appendix B. Notices	75
	Document object.		Trademarks	77
	Documents object			
	Error object		Index	79
	Image object			

About this reference

This reference provides information for the Content Manager licensed programs. Detailed program reference information is provided about the application programming interface (API) functions for the Content Manager Client for Windows interfaces.

This reference assumes that you are familiar with the concepts detailed in *Planning and Installing Your Content Manager System* and the *Content Manager System Administration Guide*. It is highly recommend that you review these manuals before proceeding.

Who should use this reference

This reference is for programmers, analysts, and other technical support staff who design and implement document management systems.

This reference is intended for Microsoft® Windows® programmers.

How this reference is organized

This reference contains the following sections:

- Chapter 1, “Using the Automation interfaces”, on page 1, describes the Content Manager Client for Windows objects and programming tips.
- Chapter 2, “Automation interface properties and methods”, on page 7, describes the properties and methods associated with Content Manager Client for Windows objects.
- Chapter 3, “Client for Windows user exit routines”, on page 43, describes the use and parameters associated with user exit routines.

Style conventions

To help you understand the text, this reference uses the following conventions.

Convention	Stands for
<i>Italic</i>	Field names in data structures. Names of books as references. Parameter names in interface functions. Terms defined for the first time in the reference
<i>ITALIC UPPER CASE</i>	The maximum length of a field
Bold	Interface function names (example: ChildCompArray)
BOLD UPPER CASE	Field values to specify. Parameter values to specify.

Restriction: The names of some data structures or symbolic constants might conflict with the naming conventions used by your applications. If this happens, please adjust your naming conventions to prevent conflicts.

For parameter entry, where the *Client for Windows Programming Reference* manual indicates ‘not supported’, the user must enter a place-holder value in accordance with the data-type field, even though the data will not actually be used by the program. For parameter value:

- BOOL = enter a null or a zero

- CHAR = enter a letter
- ULONG or USHORT = enter any number or a zero
- For all other parameters = enter null

Key concepts

Content Manager Version 8 expands the existing library server data-modelling capabilities to allow a component to have one or more child components to form a hierarchy. Each child component can in turn have children of their own, resulting in a tree of components. However, only one level of child components is viewable in the Client for Windows.

The flat data-structures previously used to represent index classes and key fields are replaced by new classes to represent the tree structure of components and attributes.

The following terms are new for Version 8:

Item type

An item type in Version 8 is analogous to an index class in Version 7. The internal representation of item types has changed in Version 8. What the user sees as a single object internally is split into two. The item type definition, which consists of system-defined properties of the item, and at least one component, called the root component, contains the attributes.

Item type is often used to refer to entities, because the user sees the *item type* definition and the component as a single unit.

Component

A component is the piece of an item type or entity that contains the information to logically define the entity. It is a collection of one or more attributes that together describe the entity. Components have certain properties associated with them such as a unique ID, a component name, description, and so forth.

Where to find more information

Your product package includes a complete set of information to help you plan for, install, administer, and use your system. Product documentation and support are also available on the Web.

Information included in your product package

The product package contains an information center and each publication in portable document format (.PDF).

The information center

The product package contains an information center that you can install when you install the product. For information about installing the information center see *Planning and Installing Your Content Management System*.

The information center includes the documentation for Content Manager, Enterprise Information Portal, and VideoCharger. Topic-based information is organized by product and by task (for example, Administration). In addition to the provided navigation mechanism and indexes, a search facility also aids retrievability.

PDF publications

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

Table 1 shows the Content Manager publications included with IBM Content Manager.

Table 1. Content Manager publications

File name	Title	Publication number
install	<i>Planning and Installing Your Content Management System</i> ¹	GC27-1332-01
migrate	<i>Migrating to Content Manager Version 8</i>	SC27-1343-01
sysadmin	<i>System Administration Guide</i>	SC27-1335-01

When you order IBM Content Manager, you also receive Enterprise Information Portal. Or, you can separately order Enterprise Information Portal. Table 2 shows the Enterprise Information Portal publications that are included with the product.

Table 2. Enterprise Information Portal publications

File name	Title	Publication number
apgwork	<i>Application Programming Guide for Windows</i> ¹	SC27-1347-01
ecliinst	<i>Installing, Configuring, and Managing the eClient</i>	SC27-1350-02
eipinst	<i>Planning and Installing Information Integrator for Content</i>	GC27-1345-01
eipmanag	<i>Managing Information Integrator for Content</i>	SC27-1346-01
messcode	<i>Messages and Codes</i> ²	SC27-1349-01

Notes:

1. The *Application Programming Guide for Windows* contains information about programming applications for both Content Manager and Enterprise Information Portal.
2. *Messages and Codes* contains the messages and codes for Content Manager and Enterprise Information Portal.

Support available on the Web

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

www.ibm.com/software/data/cm/

www.ibm.com/software/data/eip/

The documentation is included in softcopy with the product. To access product documentation on the Web, click **Library** on the product Web site.

An HTML-based documentation interface, called Enterprise Documentation Online (EDO), is also available from the Web. It currently contains the API reference information. Go to the Enterprise Information Portal Library Web page for information about accessing EDO.

How to send your comments

Your feedback helps IBM to provide quality information. Please send any comments that you have about this publication or other Content Manager or Enterprise Information Portal documentation. You can use either of the following methods to provide comments:

- Send your comments from the Web. Visit the IBM Data Management Online Reader's Comment Form (RCF) page at:
www.ibm.com/software/data/rcf
You can use the page to enter and send comments.
- Send your comments by e-mail to comments@vnet.ibm.com. Be sure to include the name of the product, the version number of the product, and the name and part number of the book (if applicable). If you are commenting on specific text, include the location of the text (for example, a chapter and section title, a table number, a page number, or a help topic title).

Chapter 1. Using the Automation interfaces

This section provides an overview of the use and abilities of the Content Manager Client for Windows. By using the provided interfaces, based on OLE 2.0 Automation, you can enable another Windows-based application to:

- Log on to Content Manager
- Perform document and folder searches
- Display table of contents (TOCs, also known as item lists) for search results, folders, or worklists
- Display and annotate documents
- Add, delete, and modify files associated with a document
- Perform simple workflow operations on documents or folders
- Retrieve the schema information for a Content Manager system
- Update index information for documents and folders

Background information about Automation

Automation enables an application's command operations to be manipulated from outside that application. The Content Manager Client for Windows provides Automation objects that can be manipulated from programs built using programming environments such as Visual Basic (Version 3.0 or above), Visual C++, and Power Builder.

To manipulate the Content Manager Client for Windows objects, you need to know the properties and methods for each object. See Chapter 2, "Automation interface properties and methods" for a list of the properties and methods associated with all Content Manager Client for Windows objects.

To use the Client's OLE interfaces, your program must create an OLE automation Application object from the client application by executing the following command:

- `CreateObject("ICMClient.Application")` in Visual Basic
- `CreatedDispatch("ICMClient.Application")` in C++

The Application object that is returned from this call can be used for subsequent OLE calls. For an example, see "Sample Visual Basic program" on page 5.

Attention: The operating system always sends the OLE requests to the first instance of the Client for Windows executable (ICMClient.exe), even if multiple instances are running.

Properties

Properties are similar to Visual Basic variables, except that properties are located inside the Content Manager Client for Windows objects. Just as you can read or write variables, you can set (that is, write) or get (that is, read) properties.

Not all properties are read/write properties; some properties are read-only and others are write-only. For example:

- The Visible property of the Application object is a read/write property. It can be used to find out whether the program is currently visible on the screen. If the

value of the property is set to True, the program is currently visible. Setting the value of the Visible property to False causes the program to be hidden.

- The Name property of the Item object is a read-only property. It contains the name by which Content Manager refers to the item.
- The Password property of the Application object is write-only. It sets the password if the Client is not already logged on.

Methods

Methods are similar to Visual Basic procedures or function procedures. You can call a method to perform an operation inside Content Manager Client for Windows (that is, invoke a command operation). For example, the OpenWorklist method of the Application class displays worklists.

Content Manager Client for Windows objects

The Content Manager Client for Windows Automation objects are designed according to Microsoft guidelines. Therefore, as is the case with all applications that follow these guidelines, Content Manager Client for Windows has an Application object, a Documents collection object, and a Document object.

Attention: "Document", in this instance, refers to the Microsoft Windows document, not a Content Manager document.

Content Manager Client for Windows has an Item object to manage multiple Items and to provide information and interfaces to Content Manager items such as documents, folders, and worklists. In addition, the Content Manager Client for Windows provides an Image object that holds the Content Manager document that is currently open in the image viewer and an information-only object called Error. The Error object is provided to allow applications to determine what errors have occurred.

The Content Manager Client for Windows Automation interface can be used with any programming language that supports Automation.

Application object

The main object that the Content Manager Client for Windows exposes is the Application object. After a program obtains access to the Application object, it can access or create all other Content Manager Client for Windows objects.

For example, the Logon method is invoked to log on to Content Manager, and the Quit method is invoked to exit the program. Therefore, programs designed to interface with Content Manager Client for Windows must first create the Application object.

After the Content Manager Client for Windows is running, you can use it to interact with a Content Manager server. You can create items using the Application object, you can work with folder and worklist TOCs (also known as item lists) using the Document object, and you can display documents using the Image and Document objects.

Document object

After you create a Document object through the OpenTOC or OpenDocument method of the Documents collection object, the object can be displayed, and a number of methods can be executed. For example, you can query any of the items that are currently selected in the Document TOC.

Documents collection object

The Documents collection object is a collection of Document objects. It can be compared to an array of TOCs (folders, search results, or worklists) and Content Manager documents. The TOCs and Content Manager documents are represented by Document objects. When you open a Document object using OLE Automation, you can use the Documents object method `OpenTOC` or `OpenDocument`, with an Item object as a parameter. Also, another way to open a Document object is to manually open a document or folder (right-click -> **Open**), which automatically adds a new Document object to the Documents collection.

Error object

If an error occurs, all of the information for the error is stored in this object, including Content Manager return codes. See Table 5 on page 27 for OLE interface return codes.

Image object

The Image object represents a special document. It is the currently visible Content Manager document. The Image object is opened by calling its `OpenDocument` method with an Item object as a parameter.

Item object

The Item object represents a Content Manager item such as a document, folder, or worklist. The Item object enables you to display the item (by passing it as a parameter to other objects), query its item type, and attributes, and perform a number of other actions.

Items collection object

The Items collection object is a list of related Items. For example, the Document method `Selections` returns the Items collection object containing all of the items that you selected. The Items collection object has methods that return or delete a specific Item object from the collection.

You can have more than one Items collection object defined at one time. However, it is your responsibility to keep track of the Items collection objects, because the only way to work with an Items collection object is when it is returned from a method.

Programming tips

To integrate your application with the Content Manager Client for Windows using the Automation interface, the development environment for your application must be able to access Automation objects. Applications that can access Automation objects are, but not limited to:

- Microsoft Visual Basic
- Microsoft Visual C++
- Power Builder
- Microsoft Excel
- Microsoft Access

The following sections provide details for programming with Automation, including information about releasing objects and handling errors.

Releasing objects

Programming with Automation requires attention to object release; programs that allocate objects are responsible for freeing the objects after use. For example, a Content Manager Client for Windows object is created in Visual Basic using this operation:

```
Dim MyItem As Object
Set MyItem = MyApp.GetWorklistFromName("Worklist Name")
```

In this operation, the Content Manager Client for Windows allocates memory to hold the Item object and returns a pointer to the object. The pointer is stored in the MyItem variable. To release the Item object, use the following statement:

```
Set MyItem = Nothing
```

In this operation, the Content Manager Client for Windows releases the memory it previously allocated for the Item object. Failure to release objects will result in the Content Manager Client for Windows eventually running out of memory. Also, the Content Manager Client for Windows will not exit if any objects are left open.

Handling errors

The Content Manager Client for Windows throws an exception when it detects an error. In Visual Basic, exceptions can be caught with the OnError statement. Programs that count on exceptions to catch errors do not need to check the return value after calling a method.

A viable strategy for processing the Content Manager Client for Windows errors is to execute an OnErrorResumeNext statement at program startup and to test the value of the built-in Visual Basic Err variable upon return from a method. When Err is non-zero, an error has occurred and the Error object can be consulted to obtain the details (the Error object is a property of the Application object). The Error object contains the actual error codes and the error message string.

Most methods return an error status. The type of this status is VT_I4, which in Visual Basic translates to the Long data type. The error status is either zero (successful) or non-zero (error detected).

Property and argument types

For OLE Automation types, see Chapter 2, "Automation interface properties and methods". Table 3 shows a list of property and argument types that can be translated into Visual Basic types and Visual C++ types.

Table 3. List of Automation Types and their meanings in Visual Basic and C++.

OLE Type	Visual Basic	C++	Description
VT_BOOL	Boolean	long	A logical value with two possible values: TRUE or FALSE.
VT_BSTR	String	Char Array, zero terminated	An ASCII string. Can have any type of character data, but usually holds user readable text.

Table 3. List of Automation Types and their meanings in Visual Basic and C++. (continued)

OLE Type	Visual Basic	C++	Description
VT_DISPATCH	Object	IDispatch*	A reference to an Automation object. Read the method or property to determine what type of object will be returned.
VT_EMPTY	(N/A)	void	No value
VT_I2	Number	short	A short integer. It can be positive or negative. The acceptable range is -32768 to 32767.
VT_I4	Number	long	A long integer. It can be positive or negative. The acceptable range is -2 147 483 648 to 2 147 483 647.
VT_VARIANT (safe array)	Array (Visual Basic 4.0 or later only)	IVariant*	A safe array of objects. In the areas where safe arrays are used, the object type is VT_BSTR.

Sample Visual Basic program

The following sample program shows the code for a Visual Basic program that invokes the client application and causes it to display the first item in the "JudyWork" worklist. The best way to learn from this program is to type it into Visual Basic and then run it.

Attention: This code is written in Visual Basic version 6. It does not work with Visual Basic .NET

To keep the example readable, no error handling has been taken into account:

```
' This example invokes the client application and causes it to display
' a worklist. It then displays the first item in the worklist,
' whether it is a document or a folder.
' Data declarations
  Dim ICMApp As Object
  Dim Worklist As Object
  Dim Docs As Object
  Dim Doc As Object
  Dim Item As Object
' Get the application objects
  Set ICMApp = CreateObject("ICMClient.Application")
' Set login information
  ICMApp.User = "JUDY"
  ICMApp.Password = "mypass"
  ICMApp.Server = "ICMNLADB"
' Log into Content Manager
  ICMApp.Logon
' Get the worklist item
  Set Worklist = ICMApp.GetWorklistFromName("JudyWork")
' Display the worklist
  Set Docs = ICMApp.Documents
```

```

        Set Doc = Docs.OpenTOC(Worklist, True, 0)
' Get next item from worklist
        Set Item = Worklist.NextWorklistItem
' Find out if the item is a folder or a document
        If (Item.Type = 1) Then
            ' Document! Display it.
            ICMApp.Image.OpenDocument Item
        Else
            ' Must be a folder. Display it.
            Docs.OpenTOC Item
        End If
' Clean up
        Set Worklist = Nothing
        Set Docs = Nothing
        Set Doc = Nothing
        Set Item = Nothing
        ICMApp.Quit
        Set ICMApp = Nothing

```

The sample program displays the first item (document or folder) in a worklist. The program takes the following logical steps to return the first item:

1. Loads the client application.
2. Configures the user ID and password that is used to log onto the default client application library server.
3. Logs onto the client application.
4. Retrieves the worklist.
5. Opens the worklist using the Documents object.
6. Gets the first item in the worklist and determines if it is a document or a folder. If it is a folder, it is passed to the Documents object. If it is a document, it is passed to the Image object.
7. Ends the client application.

Chapter 2. Automation interface properties and methods

This section describes the properties and methods associated with the Automation Interface used with the Client for Windows. You can use the properties and methods to interact with the Client for Windows application from an application customized for your business. The objects that you can use are:

- “Application object”
- “Document object” on page 20
- “Documents collection object” on page 24
- “Error object” on page 27
- “Image object” on page 29
- “Item object” on page 30
- “Items collection object” on page 41

You can write OLE automation custom applications in a number of languages, including Microsoft Visual Basic, Microsoft Visual C++, and Sybase PowerBuilder.

Restriction: The custom application must reside on the same Windows machine that the Client for Windows is installed and running on.

Application object

The Application object provides access to application-level information and controls. For example, it provides access to:

- Session services through the Logon and Quit methods
- System information such as all defined attributes, item types, and worklists
- Client for Windows application controls

Application object properties

The Application object has the following properties:

Application

The Application property returns the Application object.

Data Type: VT_DISPATCH (Application)

This property is read-only.

AttributeTranslation

The AttributeTranslation property returns the AttributeTranslation property value. If the value is set to true, then date, time, and number attributes are formatted according to the local system settings. If the AttributeTranslation property is set to false, then the attribute value is not translated to the local system settings. This attribute affects Item object methods that get and set attribute values.

Data Type: VT_BOOL

This property is read/write.

Documents

The Documents property returns a Documents collection object.

Data Type: VT_DISPATCH (Documents)

This property is read-only.

Error The Error property returns the most recent Error object.

Data Type: VT_DISPATCH (Error)

This property is read-only.

hWnd

The hWnd property returns the client's main window handle.

Data Type: VT_I4

This property is read-only.

Image The Image property returns the Image object that is currently visible in the image viewer.

Data Type: VT_DISPATCH (Image)

This property is read-only.

NewPassword

The NewPassword property is used to change the user's password. You set this property before calling the Logon method. If the user successfully logs on, the user's password is changed. If the Client is already logged then it throws an exception, or else sets the NewPassword property to an input VT_BSTR value.

Data Type: VT_BSTR

This property is write-only.

Password

The Password property is the password to be used when the Logon method is called to log on to the Content Manager library server. The Password property sets the password if the Client is not already logged on, else throws an exception.

Data Type: VT_BSTR

This property is write-only.

Server The Server property returns the server name to which the Server property is set. If the client has already logged on, the Server property is set to the server that the client is logged on and throws an exception, or else it is set to an input VT_BSTR value.

Data Type: VT_BSTR

This property is read/write.

User The User property returns user name to which User property is set. If the client is already logged on, the User property is set to the user that the client is logged as and throws an exception, or else it is set to an input VT_BSTR value.

Data Type: VT_BSTR

This property is read/write.

Visible

The Visible property contains the visible status of the Client frame window. Nonzero (True) if the window was previously visible; 0 (false) if the Client was previously hidden.

Data Type: VT_BOOL

This property is read/write.

Application object methods

The Application object supports the following methods:

Activate

The Activate method attempts to force the client into the foreground.

Parameters: None

Returns: VT_EMPTY

AttributeArray

The AttributeArray method returns a safe array of VT_BSTRs containing the names of all of the attributes defined at the time that the Logon method was executed.

Parameters: None

Returns: VT_VARIANT (safe array of VT_BSTR)

AttributeList

The AttributeList method returns a string with all of the attributes defined at the time that the Logon method was executed. The attributes are separated by the string separator argument.

Parameters: Separator as VT_BSTR

Returns: VT_BSTR

ChildCompArray

The ChildCompArray method returns a safe array of VT_BSTRs containing the child component names that were associated with the specified component at the time that the Logon method was executed.

Parameters: ComponentName as VT_BSTR

Returns: VT_VARIANT (safe array of VT_BSTR)

ChildCompAttributeArray

The ChildCompAttributeArray method returns a safe array of VT_BSTRs containing the names of all of the attribute names that were associated with the specified child component at the time that the Logon method was executed.

Parameters: ComponentName as VT_BSTR

Returns: VT_VARIANT (safe array of VT_BSTR)

ChildCompAttributeList

The ChildCompAttributeList method returns a string with all of the attribute names that were associated with the specified child component at the time that the Logon method was executed. The attribute names are separated by the string separator argument.

Parameters:

ComponentName as VT_BSTR

Separator as VT_BSTR

Returns: VT_BSTR

ChildCompList

The ChildCompList method returns a string with a list of all of the child component names that were associated with a specified component at the

time that the Logon method was executed. The child component names are separated by the string separator argument.

Parameters:

 ComponentName as VT_BSTR

 Separator as VT_BSTR

Returns: VT_BSTR

CreateItem

The CreateItem method creates an Item object of the type specified. This item belongs to NOINDEX item type.

Parameters: Type as VT_I2

Returns: VT_DISPATCH (Item)

The following constants are valid item types:

- OLEAPI_DOCUMENT(109)
- OLEAPI_FOLDER(110)

DisableMenus

The DisableMenus method allows you to disable menu classes. You specify the menus to be disabled using the *Flags* argument. The valid values for this method are:

DISABLE_ACTION_LIST(0x8000)

Prevents the user from performing any actions, from the server, on an item in a process.

DISABLE_ACTIVATE (0x1000)

Prevents the user from activating an item in suspended status.

DISABLE_ATTRIBUTE_VALUE_CHANGE (0x0040)

Prevents the user from changing to a different item type and from editing the attribute values for the item type. The user can browse the menu and copy the values listed in the window. If you specify this value, the system ignores the DISABLE_ITEM_TYPE_CHANGE flag.

DISABLE_CHANGE_PROCESS(0x4000)

Prevents the user from changing items from one process to another.

DISABLE_CHECKIN (0x001)

Prevents the user from checking in items.

DISABLE_CHECKOUT (0x002)

Prevents the user from checking out items.

DISABLE_DELETE (0x004)

Prevents the user from deleting items.

DISABLE_EXPORT (0x008)

Prevents the user from exporting items.

DISABLE_FOLDER_FUNCTIONS (0x0010)

Prevents the user from adding items to an existing folder, adding items to a new folder, or removing items from a folder.

DISABLE_ITEM_TYPE_CHANGE (0x0020)

Prevents the user from changing to a different item type. The user can still edit the attributes for the item type.

DISABLE_NOTE_APPEND (0x0080)

Prevents the user from editing previously saved notes and from adding new notes. The user can open and copy existing notes in browse mode. When no notes exist, the Note Log window is not displayed. If you specify this value, the system ignores the DISABLE_NOTE_EDIT flag.

DISABLE_NOTE_EDIT (0x0100)

Prevents the user from editing previously saved notes. The user can still add new notes.

DISABLE_OPTIONS (0x0200)

Prevents the user from using the Options-->Preferences menu options.

DISABLE_PRINT (0x0400)

Prevents the user from printing items.

DISABLE_REMOVE_FROM_PROCESS(0x2000)

Prevents the user from removing items from a process.

DISABLE_SEARCH (0x0800)

Prevents the user from searching.

DISABLE_START_PROCESS(0x20000)

Prevents the user from starting items on a process.

DISABLE_SUSPEND (0x10000)

Prevents the user from suspending items.

Parameters: Flags as VT_I4, Hide as VT_BOOL

Returns: VT_EMPTY

ExtendedPrintSetup

The ExtendedPrintSetup method allows the external application to modify the default printing behavior for the client. The options described are extended print features that cannot be configured from the standard user interface.

Parameters:

Comments as VT_BOOL
 Borders as VT_BOOL
 SinglePage as VT_BOOL
 HorizPages as VT_BOOL
 PageNumbers as VT_BOOL
 NumRows as VT_I2
 NumColumns as VT_I2

Returns: VT_EMPTY

Borders

Borders enables or disables a single pixel line around the image. This feature is most useful if you set *SinglePage* to false.

Comments

Comments is an alternate way to disable printing the annotations. This option duplicates the *PrintMarkup* argument in the *Item.PrintItem* method.

HorizPages

HorizPages is used when the *SinglePage* argument is false. *HorizPages* specifies image orientation on the printed page: true for horizontal and false for vertical.

NumRows and NumColumns

NumRows and *NumColumns* are used when the *SinglePage* argument is false. *NumRows* and *NumColumns* define how many images to horizontally and vertically display on a single printed page.

PageNumbers

PageNumbers prints the page number on each image. If *PageNumbers* is set to true, the page number prints in the upper left corner of each image (a page might show more than one image).

SinglePage

You can use *SinglePage* in conjunction with the *NumRows*, *NumColumns*, and *HorizPages* arguments to define how to arrange images on pages. If *SinglePage* is true, only one image prints on each page. If *SinglePage* is false, the other three arguments define how many images to print on each page.

GetActiveView

The *GetActiveView* method returns the active view name.

Parameters: *ItemTypeName* as VT_BSTR

Returns: VT_BSTR

GetDisplayNameFromName

The *GetDisplayNameFromName* method returns the display name.

Parameters:

- *Name* as VT_BSTR
- *Type* as VT_I2

Table 4. Type constants for retrieving display names

Type	Constant
Attribute	101
ItemType	102
ItemableView	103
Process	104
Worklist	105
MIMETYPE	106

Returns: VT_BSTR

GetIDFromName

The GetIDFromName method returns the ID of the name specified. The semantics of the name is identified by the *Type* value.

Parameters: Name as VT_BSTR, VT_I2 as Type

Returns: VT_I4

Type has the following values:

- OLEAPI_ATTRIBUTE(101)
- OLEAPI_ITEMTYPE(102)
- OLEAPI_ITEMTYPEVIEW(103)
- OLEAPI_MIMETYPE(106)

GetItemFromPID

The GetItemFromPID method returns an Item object with the item PID specified.

Parameters: ItemPid as VT_BSTR

Returns: VT_DISPATCH (Item)

GetWorklistFromName

The GetWorklistFromName method returns the Item object associated with the worklist specified in the Name argument.

Parameters: Name as VT_BSTR

Returns: VT_DISPATCH (Item)

IsLoggedOn

The IsLoggedOn method returns TRUE if the client is already logged on, else it returns FALSE.

Parameters: None

Returns: VT_BOOL

IsValueRequired

The IsValueRequired method returns true if a attribute value is required and false if it is not.

Parameters:

 ComponentName as VT_BSTR

 AttributeName as VT_BSTR

Returns: VT_BOOL

ItemTypeArray

The ItemTypeArray method returns a safe array of VT_BSTRs containing the names of all of the item types that were defined at the time that the Logon method was executed.

Parameters: None

Returns: VT_VARIANT (safe array of VT_BSTR)

ItemTypeAttributeArray

The ItemTypeAttributeArray method returns a safe array of VT_BSTRs containing the names of all of the attributes that are associated with the specified item type at the time that the Logon method was executed.

Parameters: ItemTypeName as VT_BSTR

Returns: VT_VARIANT (safe array of VT_BSTR)

ItemTypeAttributeList

The ItemTypeAttributeList method returns a string with all of the attributes that are associated with the specified item type at the time that the Logon method was executed. The attributes are separated by the string separator argument.

Parameters:

 ItemTypeName as VT_BSTR

 Separator as VT_BSTR

Returns: VT_BSTR

ItemTypeList

The ItemTypeList method returns a string with a list of all of the item types defined at the time that the Logon method was executed. The item types are separated by the string separator argument.

Parameters: Separator as VT_BSTR

Returns: VT_BSTR

ItemTypeViewsArray

The ItemTypeViewsArray method returns a safe array of VT_BSTRs containing all of the valid item type views defined at the time that the Logon method was executed.

Parameters: ItemTypeName as VT_BSTR

Returns: VT_VARIANT (safe array of VT_BSTR)

ItemTypeViewsList

The ItemTypeViewsList method returns a string with a list of all of the valid item type views defined at the time that the Logon method was executed. The views are separated by the string separator argument.

Parameters:

 ItemTypeName as VT_BSTR

 Separator as VT_BSTR

Returns: VT_BSTR

Logon The Logon method logs on to Content Manager. If the Client is already logged on to the server, the Client appears in the foreground using the same logon session and this method throws an OLEAPI_RC_ALREADY_LOGGED_ON exception. The OLE interfaces can only use one session of the Client at a time. When the Client logs on to the server for the first time, then it requires all of the following properties to be set: User, Password, and Server. If any one of these properties is not set, then the Client Logon window appears with the already-provided properties filled in. The user must then complete the rest of the required properties in the Logon window.

Parameters: None

Returns: VT_EMPTY

MIMETypeArray

The MIMETypeArray method returns a safe array of VT_BSTRs containing the names of all of the MIME types defined at the time that the Logon method was executed.

Parameters: None

Returns: VT_VARIANT (safe array of VT_BSTR)

MIMETypeList

The MIMETypeList method returns a list of MIME types defined at the time that the Logon method was executed. The MIME types are separated by the string separator argument.

Parameters: VT_BSTR Separator

Returns: VT_BSTR

OpenBasicSearch

The OpenBasicSearch method displays the Basic Search dialog.

Parameters: None

Returns: VT_EMPTY

OpenScan

The OpenScan method displays the Scan dialog.

Parameters: None

Returns: VT_EMPTY

OpenWorklist

The OpenWorklist method displays the list of worklists.

Parameters: None

Returns: VT_EMPTY

PrintSetup

The PrintSetup method allows the external application to modify the default printing behavior for the client. Values specified with this method are saved as the default print settings, not only for Automation printing, but also for user-initiated printing.

Parameters:

Printer as VT_BSTR

PaperSize as VT_I2

Portrait as VT_BOOL

Copies as VT_I2

Scaling as VT_BOOL

Returns: VT_EMPTY

- *Printer* specifies the name of the printer to which you want to print
- *PaperSize* defines the paper type. Specify the type you want by assigning the number that corresponds to it (1 through 41) in the following list:
 1. Letter 8 1/2 x 11 inches
 2. Letter Small 8 1/2 x 11 inches
 3. Tabloid 11 x 17 inches
 4. Ledger 17 x 11 inches
 5. Legal 8 1/2 x 14 inches

6. Statement 5 1/2 x 8 1/2 inches
 7. Execute 7 1/4 x 10 1/2 inches
 8. A3 297 x 420 mm
 9. A4 210 x 297 mm
 10. A4 Small 210 x 297 mm
 11. A5 148 x 210 mm
 12. B4 (JIS) 250 x 354
 13. B5 (JIS) 182 x 257 mm
 14. Folio 8 1/2 x 13 inches
 15. Quarto 215 x 275 mm
 16. 10x14 inches
 17. 11x17 inches
 18. Note 8 1/2 x 11 inches
 19. Envelope #9 3 7/8 x 8 7/8
 20. Envelope #10 4 1/8 x 9 1/2
 21. Envelope #11 4 1/2 x 10 3/8
 22. Envelope #12 4 1/2 x 11
 23. Envelope #14 5 x 11 1/2
 24. C size sheet
 25. D size sheet
 26. E size sheet
 27. Envelope DL 110 x 220 mm
 28. Envelope C5 162 x 229 mm
 29. Envelope C3 324 x 458 mm
 30. Envelope C4 229 x 324 mm
 31. Envelope C6 114 x 162 mm
 32. Envelope C65 114 x 229 mm
 33. Envelope B4 250 x 353 mm
 34. Envelope B5 176 x 250 mm
 35. Envelope B6 176 x 125 mm
 36. Envelope 110 x 230 mm
 37. Envelope Monarch 3.875 x 7.5 inches
 38. 6 3/4 Envelope 3 5/8 x 6 1/2 inches
 39. US Std Fanfold 14 7/8 x 11 inches
 40. German Std Fanfold 8 1/2 x 12 inches
 41. German Legal Fanfold 8 1/2 x 13 inches
- *Portrait* defines the orientation (true = Portrait, false = Landscape)
 - *Copies* specifies the number of copies to print
 - *Scaling* specifies how to print to page. Set the scaling to true to shrink a large document to fit on a smaller paper size, or, set the scaling to false to keep the format of the document.

ProcessArray

The ProcessArray method returns a safe array of VT_BSTRs containing all of the process names defined at the time that the Logon method was executed.

Parameters: None

Returns: VT_VARIANT (safe array of VT_BSTR)

ProcessList

The ProcessList method returns a string with a list of all of the process names defined at the time that the Logon method was executed. The process names are separated by the string separator argument.

Parameters: Separator as VT_BSTR

Returns: VT_BSTR

QueryPrivilege

The QueryPrivilege method allows an external application to determine the privilege for the logged on user based on the context and the context type specified.

Parameters:

PrivilegeName as VT_BSTR

Context as VT_VARIANT(VT_BSTR)

Type as VT_I2

Returns: VT_BOOL

Returns true, if authorized, or else it is false. Context is a string value that is based on the context type. *Type* has the following values:

- OLEAPI_USER(108): Context is ignored, returns the user general privilege.
- OLEAPI_ITEMTYPEVIEW(103): Context is item type view name, returns the user privilege to access the item type view name.
- OLEAPI_PROCESS(104): Context is the process name, returns the user privilege to access the process.
- OLEAPI_ITEM(107): Context is the item's PID string, returns the user privilege to access the item.

Quit The Quit method ends the Client for Windows application. All open documents, including item lists, any image viewer sessions, and all outstanding Item and Items collection objects are closed.

Parameters: None

Returns: VT_EMPTY

Search

The Search method returns search results in a temporary folder. You can customize what search results return by modifying the parameters of your search. **Important:** When you specify an item type name, an attribute name, or a child component name, be sure to use the internal name, rather than the description or display name.

Parameters:

ItemTypeName as VT_BSTR

QuerySubString as VT_VARIANT(VT_BSTR)

AllVersions as VT_BOOL

TypeFilter as VT_I2

SuspendFilter as VT_I2

ProcessName as VT_VARIANT(VT_BSTR)

StepName as VT_VARIANT(VT_BSTR)

Returns: VT_DISPATCH (Item)

- *ItemTypeName* is the name of the item type to which the item belongs. You can search all of the item types if you use the asterisk (*) in place of a specific item type name for this parameter.
- *QuerySubString* contains that part of the search criteria which is based on attributes of the item type.

Examples of QuerySubString are:

```
"@SOURCE LIKE "IMPO%"
```

```
"@SOURCE = "SCANNER"
```

```
"@SOURCE = "SCANNER" AND @USER_ID = "SMITH"
```

To search on child component attributes, use the following QuerySubString syntax:

```
"ChildComponentName/@AttributeName CONDITION "AttributeValue"
```

A list of operators that you can use as conditions in the QuerySubString are:

Operator	Description
=	Equal to
!=	Not equal to
>	Greater than
>=	Greater than equal to
<	Less than
<=	Less than equal to
LIKE	Like
NOT LIKE	Not like
IS NULL	Is null
IS NOT NULL	Is not null
AND	And
OR	Or
BETWEEN	Between
%	Wild character for multiple character match
_	Wild character for single character match
(...)	Parentheses for scope resolution

More complex QuerySubString examples are:

```
"(@SOURCE = "SCANNER" OR @SOURCE = "OLE") AND (@SOURCE =  
"CUSTOM_IMPORT")"
```

```
"(@MIDDLE_INITIAL IS NULL AND @FIRST_NAME = "G__") OR  
(@LAST_NAME = "D%")"
```

```
"(@SALARY BETWEEN "50000" AND "70000") AND (@EXPERIENCE <=  
"2")"
```

- *AllVersions* can be TRUE or FALSE. If set to TRUE, the Search method returns all versions of the items that match the search criteria. Also, if

AllVersions is set to TRUE, then a process aware search is not possible, meaning that SuspendFilter should be set to 0, ProcessName should be set to Null, and StepName should be set to Null. If AllVersions is set to FALSE, it returns only the latest version of each item.

- *TypeFilter* has the following values:
 - OLEAPI_DOCUMENTS(111): Only Content Manager document items are returned.
 - OLEAPI_FOLDERS(112): Only Content Manager folders are returned.
 - OLEAPI_DOCS_AND_FLDRS(113): Both Content Manager documents and Content Manager folders are returned.
- *SuspendFilter* has the following values:
 - OLEAPI_ALL_STATUS(0): All items, irrespective of status, is returned.
 - OLEAPI_ACTIVE_STATUS(1): Only active items are returned.
 - OLEAPI_SUSPENDED_STATUS(2): Only suspended items are returned.

Important: If the AllVersions flag is set to TRUE, SuspendFilter should be set to 0.

- *ProcessName* is the name of the process that the item is in.

Important: If the AllVersions flag is set to TRUE, ProcessName should be set to Null.

- *StepName* is the work node name that the item is in.

Important: If the AllVersions flag is set to TRUE, StepName should be set to Null.

SetActiveView

The SetActiveView method Sets the active view for the specified item type name.

Parameters:

ItemTypeName as VT_BSTR

ViewName as VT_BSTR

Returns: VT_EMPTY

SetPrintRect

The SetPrintRect method allows you to define a rectangle that contains the images when they are printed on the page. Values specified with this method are saved as the default print settings, not only for Automation printing, but also for user-initiated printing.

Parameters:

RectLeft as VT_I2

RectTop as VT_I2

RectRight as VT_I2

RectBottom as VT_I2

Returns: VT_EMPTY

The four arguments define the distance in millimeters of each box side from the upper left hand corner of the paper. You can reset the print rectangle to none by calling the SetPrintRect method again and setting all arguments to 0.

Important: If you specify a rectangle that does not fit on the paper, some or all of the image will not appear when printed.

VersioningType

The `VersioningType` method returns the Versioning rule type for the item specified by the context.

Parameters:

ItemTypeName as VT_BSTR

Context as VT_I2

Returns: VT_I2

Context can take the following values:

- OLEAPI_ATTRIBUTE (101)
- OLEAPI_BASE (300)
- OLEAPI_BASSETEXT (301)
- OLEAPI_BASESTREAM (302)
- OLEAPI_NOTELOG (303)
- OLEAPI_ANNOTATION (304)

VersionsSupported

The `VersionsSupported` method returns the number of Versions supported for the given item type. A value of 0 indicates that Versioning is not supported for this item type.

Parameters: ItemTypeName as VT_BSTR

Returns: VT_I4

WorklistArray

The `WorklistArray` method returns a safe array of VT_BSTRs containing the names of all the worklists that were defined at the time that the `Logon` method was executed.

Parameters: None

Returns: VT_VARIANT (safe array of VT_BSTR)

WorklistList

The `WorklistList` method returns a string with a list of all of the worklists defined at the time that the `Logon` method was executed. The worklists are separated by the string separator argument.

Parameters: Separator as VT_BSTR

Returns: VT_BSTR

Document object

The *Document* object holds information about a table of contents (TOC) or a Content Manager document. The TOC Document object can represent a folder, worklist, or search result (which is a temporary folder).

Document object properties

Application

The `Application` property returns the Application object.

Data Type: VT_DISPATCH (Application)

This property is read-only.

Count The Count property returns the number of items that are listed in the TOC. This property is valid for TOC Document objects only.

Data Type: VT_I4

This property is read-only.

Item The Item property returns the Item object that is associated with this Document.

Data Type: VT_DISPATCH (Item)

This property is read-only.

Page The Page property contains the selected page number. This property is not valid for TOC Document objects. The default value is 0.

Data Type: VT_I4

This property is read/write.

PageCount

The PageCount property contains the number of pages in a document. This property is not valid for TOC Document objects. The default value is 0.

Data Type: VT_I4

This property is read-only.

Parent The Parent property returns the parent of the Document object (which is the Documents collection object).

Data Type: VT_DISPATCH (Documents)

SelectedCount

The SelectedCount property returns the number of items that are selected in the TOC. This property is valid for TOC Document objects only.

Data Type: VT_I4

This property is read-only.

Type The Type property returns the type of item that is open in the document: a folder, worklist, or a document. The actual values are:

1 Document

2 Folder

3 Worklist

1024 Scan (the basic scan viewer, no other property or method works on this type)

Data Type: VT_I4

This property is read-only.

Document object methods

The Document object supports the following methods.

Activate

The Activate method brings a document window to the foreground.

Parameters: None

Returns: VT_EMPTY

CaretIndex

The **CaretIndex** method returns the index of the caret item (the item that contains the dotted-line rectangle in the grid) in a folder or worklist. This method is valid for TOC Document objects only.

Parameters: None

Returns: VT_I4

ClearSelect

The **ClearSelect** method clears all of the current selections in the TOC. This method is valid for TOC Document objects only.

Parameters: None

Returns: VT_EMPTY

Close

The **Close** method closes the window associated with document and removes the document from the Documents collection. The remaining Document objects in the collection are shifted down to prevent gaps in the collection.

Parameters: VT_VARIANT

The valid values are defined as symbolic constants in ICMClientOLE.h as follows:

- OLEAPI_DONT_SAVE (0): No changes are saved
- OLEAPI_UPDATE_LAST (1): The latest version of the object is updated with the changes
- OLEAPI_CREATE_NEW (2): The changes are saved in a new version of the object

Returns: VT_EMPTY

CloseIt

The **CloseIt** method is the same as the **Close** method. It is implemented solely to support Visual Basic, which uses **Close** as a reserved word. The **CloseIt** method closes the window associated with the associated document (TOC) and removes the document from the Documents collection. The remaining Document objects in the collection will be shifted down to prevent gaps in the collection.

Parameters: VT_VARIANT

The valid values are defined as symbolic constants in ICMClientOLE.h as follows:

- OLEAPI_DONT_SAVE (0): No changes are saved
- OLEAPI_UPDATE_LAST (1): The latest version of the object is updated with the changes
- OLEAPI_CREATE_NEW (2): The changes are saved in a new version of the object

Returns: VT_EMPTY

DisplayPage

The **DisplayPage** method forces the page specified to be displayed in a document. This method is not valid for TOC Document objects.

Parameters: Page as VT_I4

Returns: VT_EMPTY

FirstPage

The FirstPage method displays the first page in a document. This method is not valid for TOC Document objects.

Parameters: None

Returns: VT_EMPTY

IndexedItem

The IndexedItem method returns a single item from a TOC Document based on the index specified. This method is valid for TOC Document objects only.

Parameters: Index as VT_I4

Returns: VT_DISPATCH (Item)

LastPage

Displays the last page in a document. This method is not valid for TOC Document objects.

Parameters: None

Returns: VT_EMPTY

Maximize

The Maximize method maximizes the Document object in the main client window, hiding all other Document objects.

Parameters: None

Returns: VT_EMPTY

Minimize

The Minimize method minimizes the Document object in the main client window.

Parameters: None

Returns: VT_EMPTY

NextPage

The NextPage method displays the next page (current page, plus 1) in a document. This method is not valid for TOC Document objects.

Parameters: None

Returns: VT_EMPTY

PreviousPage

The PreviousPage method displays the previous page (current page, minus 1) in a document. This method is not valid for TOC Document objects.

Parameters: None

Returns: VT_EMPTY

Restore

The Restore method restores the Document object in the main client window to its original state (neither minimized or maximized).

Parameters: None

Returns: VT_EMPTY

Selections

The Selections method returns an Items collection object containing all of the Item objects that are selected in the TOC Document. This method is valid for TOC Document objects only.

Parameters: None

Returns: VT_DISPATCH (Items)

SelectRange

The SelectRange method selects a range of items in the TOC Document. The arguments are the first and last items to be selected. This method is valid for TOC Document objects only.

Parameters:

First as VT_I4

Last as VT_I4

Returns: VT_EMPTY

Zoom The Zoom method changes the zoom ratio of the Document object. For example, if you set the zoom ratio to 100, the image is shown at full size, pixel for pixel. If you set the zoom ratio to 50, the image is shown in half height. This method is not valid for TOC Document objects.

Parameters: Percent as VT_I4

Returns: VT_EMPTY

ZoomFit

The ZoomFit method allows you to fit the document image into the viewing rectangle. The Type argument specifies how to fit: 1 means fit height, 0 means fit width. This method is not valid for TOC Document objects.

Parameters: Fit as VT_I4

Returns: VT_EMPTY

ZoomRect

ZoomRect allows you to specify a rectangle to zoom to in the Document object. The left, top, right, and bottom arguments specify the bounding rectangle to display as large as possible in the viewing rectangle (the viewer window). The arguments are specified in pixels. This method is not valid for TOC Document objects.

Parameters:

Left as VT_I4

Top as VT_I4

Right as VT_I4

Bottom as VT_I4

Returns: VT_EMPTY

Documents collection object

The Documents collection object is a collection of all of the open Document objects.

Documents collection object properties

The Documents collection object has the following properties.

Active

The Active property holds the index of the Documents collection object that currently has the focus.

Data Type: VT_I4

This property is read-only.

Application

The Application property returns the Application object.

Data Type: VT_DISPATCH (Application)

This property is read-only.

Count The Count property holds the number of Document objects currently in the collection.

Data Type: VT_I4

This property is read-only.

Parent The Parent property returns the parent of the Documents collection object (which is the Application object).

Data Type: VT_DISPATCH (Application)

This property is read-only.

Documents collection object methods

The Documents collection object supports the following methods.

Cascade

The Cascade method cascades all of the open Document objects that are not minimized.

Parameters: None

Returns: VT_EMPTY

Close The Close method closes all of the windows associated with the Document objects and removes the Document objects from the Documents collection.

Parameters: VT_VARIANT

The valid values are defined as symbolic constants in ICMClientOLE.h as follows:

- OLEAPI_DONT_SAVE (0): No changes are saved
- OLEAPI_UPDATE_LAST (1): The latest version of the object is updated with the changes
- OLEAPI_CREATE_NEW (2): The changes are saved in a new version of the object

Returns: VT_EMPTY

CloseIt

The CloseIt method is the same as the Close method. It is implemented solely to support Visual Basic, which uses Close as a reserved word. The Close method closes all windows associated with the Document objects and removes the Document objects from the Documents collection.

Parameters: VT_VARIANT

The valid values are defined as symbolic constants in ICMClientOLE.h as follows:

- OLEAPI_DONT_SAVE (0): No changes are saved
- OLEAPI_UPDATE_LAST (1): The latest version of the object is updated with the changes
- OLEAPI_CREATE_NEW (2): The changes are saved in a new version of the object

Returns: VT_EMPTY

Item The Item method returns one of the Document objects contained in the collection specified by the Index.

Parameters: Index as VT_I4

Returns: VT_DISPATCH (Document)

OpenDocument

The OpenDocument method creates a Document object for the Content Manager document and adds it to the Documents collection object. If the Browse argument is set to TRUE, the document is opened in the viewer in read-only mode without being locked, allowing other users to open it. The Version argument specifies the Version of the document to open. If Version is greater than or equal to 1, then that Version of the document will be opened. If Version is less than or equal to 0, the latest Version is opened.

Parameters:

Item as VT_DISPATCH (Item)

Browse as VT_BOOL

Version as VT_I4

Returns: VT_DISPATCH (Document)

OpenTOC

The OpenTOC method creates a TOC Document object for the specified Table of Contents (TOC) and adds it to the Documents collection. If the Browse argument is set to TRUE, the folder TOC is opened in read-only mode without being locked, allowing other users to open it. Browse has no affect on TOCs for worklists. The Version argument specifies the Version of the TOC to open. If Version is greater than or equal to 1, then that Version of the TOC will be opened. If Version is less than or equal to 0, the latest Version is opened.

Parameters:

Item as VT_DISPATCH (Item)

Browse as VT_BOOL

Version as VT_I4

Returns: VT_DISPATCH (Document)

Tile The Tile method arranges all of the open Document objects that are not minimized in a tiled manner. The Vertical argument specifies if the objects should be tiled vertically (non-zero) or horizontally (zero).

Parameters: Vertical as VT_I4

Returns: VT_EMPTY

Error object

The Error object describes the details about any error that can have happened while executing a method in Client for Windows. For Content Manager return codes, see *IBM Content Manager for Multiplatforms/IBM Information Integrator for Content Messages and Codes*. For OLE interface returns, see Table 5.

Error object properties

The Error object has the following properties:

ErrorCode

The ErrorCode property returns the EIP error code. For more information about error codes, see *IBM Content Manager for Multiplatforms/IBM Information Integrator for Content Messages and Codes*.

Data Type: VT_I4

This property is read-only.

ErrorID

The ErrorID property contains the error ID. The ErrorID property returns the EIP error ID or the ID described in Table 5. Automation methods now return standardized error codes—either uniform four digit codes described in *Content Manager and Enterprise Information Portal IBM Content Manager for Multiplatforms/IBM Information Integrator for Content Messages and Codes* or values described in the `ICMClientOLE.h` header file, as shown in Table 5.

Data Type: VT_I4

This property is read-only.

Table 5. Standardized OLE interface return codes

OLEAPI_RC_NOT_LOGGED_ON The client is not logged on.	12000
OLEAPI_RC_INSUFFICIENT_MEMORY Insufficient memory.	12001
OLEAPI_RC_NO_ITEMS_FOUND No items found.	12002
OLEAPI_RC_ALREADY_LOGGED_ON The client is already logged on.	12003
OLEAPI_RC_INVALID_ARGUMENT Invalid argument.	12004
OLEAPI_RC_NO_DOC_OPEN No document is open.	12005
OLEAPI_RC_INVALID_ITEM Invalid item.	12006
OLEAPI_RC_INDEX_OUT_OF_RANGE The index array is out of bounds.	12007
OLEAPI_RC_ERROR_PRINTING Printing Error.	12008
OLEAPI_RC_INVALID_MIME_TYPE Invalid MIME type.	12009
OLEAPI_RC_ITEM_NOT_FOLDER The item is not a folder.	12010
OLEAPI_RC_ERROR_GETTING_PART Part not found.	12011
OLEAPI_RC_ERROR_UNLOCKING Error unlocking the item or folder.	12012
OLEAPI_RC_INVALID_DOCUMENT Invalid document.	12013
OLEAPI_RC_NOT_TOC_DOCUMENT Item not a TOC document.	12014
OLEAPI_RC_INSUFFICIENT_PRIVS The client does not have privileges to perform this operation.	12015
OLEAPI_RC_NO_SELECTIONS Zero items selected.	12016
OLEAPI_RC_NOT_DOC_DOCUMENT Item not a document item.	12017

Table 5. Standardized OLE interface return codes (continued)

OLEAPI_RC_ITEM_NOT_TOC Item not a TOC item.	12018
OLEAPI_RC_ITEM_NOT_DOCUMENT Item not a document item.	12019
OLEAPI_RC_TEMP_FOLDER The item is a temporary folder.	12020
OLEAPI_RC_INVALID_MAIN_FRAME Failed to get the handle to main frame.	12021
OLEAPI_RC_INVALID_ITEMTYPE Item type name is invalid.	12022
OLEAPI_RC_INVALID_ATTRIBUTE Attribute name is invalid.	12023
OLEAPI_RC_INVALID_COMPONENT Component name is invalid.	12024
OLEAPI_RC_ITEM_VERSION_NOT_FOUND Item version not found.	12025
OLEAPI_RC_INVALID_WORKLIST Invalid worklist name.	12026
OLEAPI_RC_INVALID_PROCESS Invalid process name.	12027
OLEAPI_RC_INVALID_ITEMTYPE_ID Invalid item type ID.	12028
OLEAPI_RC_ITEM_NOT_PROCESS Item not in process.	12029
OLEAPI_RC_MINMAX_ERROR Multivalue cardinality error.	12030
OLEAPI_RC_REQUIREDVALUE_ERROR Require value error.	12031
OLEAPI_RC_NO_CHILDCOMPONENTS No child components.	12032
OLEAPI_RC_NO_ATTRIBUTES No attributes.	12033
OLEAPI_RC_NOT_SUSPENDED Item is not suspended.	12034
OLEAPI_RC_INVALID_ITEMTYPEVIEW Invalid item type view.	12035
OLEAPI_RC_INVALID_SERVER Invalid server name.	12036
OLEAPI_RC_SERVERLIST_NOT_FOUND Server list not found.	12037
OLEAPI_RC_ZERO_ITEMTYPEVIEWS Zero item type views found.	12038
OLEAPI_RC_INVALID_TYPEFILTER Invalid type filter.	12039
OLEAPI_RC_INVALID_SUSPENDFILTER Validation failure.	12040
OLEAPI_RC_INVALID_DISPATCH Invalid dispatch.	12041
OLEAPI_RC_INVALID_PART_ALREADY_EXISTS Part already exists.	12042
OLEAPI_RC_UNEXPECTED_ERROR Unexpected error.	12999

ErrorMessage

The ErrorMessage property contains a descriptive error message describing what went wrong.

Data Type: VT_BSTR

This property is read-only.

ErrorState

The ErrorState property returns the EIP error state. For more information about error states, see *Content Manager and Enterprise Information Portal IBM Content Manager for Multiplatforms/IBM Information Integrator for Content Messages and Codes*.

Data Type: VT_BSTR

This property is read-only.

Image object

Attention: In place of the Image Object, we recommend using the Document and Documents Objects which allow you to open more than one document at a time.

The Image object holds the currently visible document that has the user's focus.

Image object properties

The Image object supports the following properties:

Application

The Application property returns the Application object.

Data Type: VT_DISPATCH (Application)

This property is read-only.

Item The Item property returns the Item object that is associated with this Image.

Data Type: VT_DISPATCH (Item)

This property is read-only.

Page The Page property contains the selected page number.

Data Type: VT_I4

This property is read/write.

Parent The Parent property returns the parent of the Image object (which is the Application object).

Data Type: VT_DISPATCH (Application)

Image object methods

The Image object supports the following methods:

Close The Close method closes all windows associated with the Image object.

Parameters: Save as VT_VARIANT (usually VT_I4). Save values are:

- OLEAPI_DONT_SAVE (0): No changes are saved
- OLEAPI_UPDATE_LAST (1): The latest version of the object is updated with the changes
- OLEAPI_CREATE_NEW (2): The changes are saved in a new version of the object

If the save value is not specified, a message window asks the user if they want to save the changes or not.

Returns: VT_EMPTY

CloseIt

Attention: The CloseIt method is the same as the Close method. It is implemented solely to support Visual Basic which uses Close as a reserved word.

Parameters: Save as VT_VARIANT (usually VT_I4). Save values are:

- OLEAPI_DONT_SAVE (0): No changes are saved
- OLEAPI_UPDATE_LAST (1): The latest version of the object is updated with the changes

- OLEAPI_CREATE_NEW (2): The changes are saved in a new version of the object

If the save value is not specified, a message window asks the user if they want to save the changes or not.

Returns: VT_EMPTY

DisplayPage

The DisplayPage method forces the page specified to be displayed in the image viewer.

Parameters: Page as VT_I4

Returns: VT_EMPTY

FirstPage

The FirstPage method displays the first page in the viewer.

Parameters: None

Returns: VT_EMPTY

LastPage

The LastPage method displays the last page in the viewer.

Parameters: None

Returns: VT_EMPTY

NextPage

The NextPage method displays the next page (the current page plus one) in the viewer.

Parameters: None

Returns: VT_EMPTY

OpenDocument

The OpenDocument method opens a new Content Manager document in the image viewer. The argument Item is the item that is to be opened. An Item error occurs if the item is a worklist or folder.

Parameters: Item as VT_DISPATCH

Returns: VT_EMPTY

PreviousPage

The PreviousPage method displays the previous page (the current page minus one) in the viewer.

Parameters: None

Returns: VT_EMPTY

Item object

The Item object represents an item that represents a folder, a worklist, a document or a search result. A TOC Item object represents a folder, worklist, or a search result (which is a temporary folder). A document is a Content Manager document item.

Item object properties

The Item object supports the following properties:

Application

The Application property returns the Application object.

Data Type: VT_DISPATCH (Application)

This property is read-only.

CheckoutUserID

The CheckoutUserID property returns the user ID who has the item checked out, if any.

Data Type: VT_BSTR

This property is read-only.

ItemPID

The ItemPID property is a string that uniquely identifies each item.

Data Type: VT_BSTR

This property is read-only.

ItemType

The Item type property is the item type to which the item belongs. The ItemType methods returns the Item type name. The UpdateItem method must be called before any changes to the ItemType property take affect.

Data Type: VT_BSTR

This property is read/write.

Name The Name property returns Content Manager's name for the item. This property is based on the attribute that is selected as the identifier (if any) when the item type was created. If the item is a worklist, the worklist name is returned.

Data Type: VT_BSTR

This property is read-only.

OwnerID

The OwnerID property returns the owner of the item in a process. This property is valid only if the item is process-aware. That is, the item was obtained using the Application object Search method with the SuspendFilter, ProcessName, or StepName parameters specified or the item was obtained from a worklist Item object.

Data Type: VT_BSTR

This property is read-only.

Parent The Parent property returns the parent of the Item object (which can be the Application object or an Items collection object) if the item has a parent. If it does not have a parent, it returns NULL.

Data Type: VT_DISPATCH

This property is read-only.

PartCount

The PartCount property returns the number of base parts stored in a document.

Data Type: VT_I4

This property is read-only.

Priority

The Priority property returns the worklist priority of the item. Valid values are -2,147,483,684 to 2,147,483,647, where -2,147,483,684 is the lowest priority. The default value is zero (0). This property is valid only if the item is process-aware. That is, the item was obtained using the Application object Search method with the SuspendFilter, ProcessName, or StepName parameters specified or the item was obtained from a worklist Item object.

Data Type: VT_I4

This property is read/write.

Process

The Process property returns the process name. This property is valid only if the item is process-aware. That is, the item was obtained using the Application object Search method with the SuspendFilter, ProcessName, or StepName parameters specified or the item was obtained from a worklist Item object.

Data Type: VT_BSTR

This property is read-only.

ReturnLimit

Returns -1 if it is a user defined worklist. Returns -2 if the worklist is a system-assigned worklist. Valid only if the item is a worklist.

Data Type: VT_I4

This property is read-only.

Step

The Step property returns the work node the item is in. This property is valid only if the item is process-aware. That is, the item was obtained using the Application object Search method with the SuspendFilter, ProcessName, or StepName parameters specified or the item was obtained from a worklist Item object.

Data Type: VT_BSTR

This property is read-only.

Suspended

The Suspended property indicates whether or not an item is suspended. This property is valid only if the item is process-aware. That is, the item was obtained using the Application object Search method with the SuspendFilter, ProcessName, or StepName parameters specified or the item was obtained from a worklist Item object.

Data Type: VT_BOOL

This property is read-only.

TOCCount

The TOCCount property returns the number of items contained by this item. Valid only if the item is a TOC Item object.

Data Type: VT_I4

This property is read-only.

Type

The Type property returns an integer value that associates to the following values:

The value is 0

A value of 0 means unknown.

The value is 1

A value of 1 means a document.

The value is 2

A value of 2 means folder.

The value is 3

A value of 3 means worklist.

Data Type: VT_I4

This property is read-only.

Version

The Version property returns the Version of the item. Valid for folders and Content Manager documents only.

Returns: VT_BSTR

This property is read-only.

Item object methods

The Item object supports the following methods.

Activate

The Activate method resumes the suspended item on the process. This method is valid only if the item is a process-aware Content Manager document or folder item.

Parameters: None

Returns: VT_EMPTY

AddAnnotationPart

The AddAnnotationPart method adds an annotation to an existing document. The Path argument must be a full path to the new annotation. If an annotation already exists, an exception is thrown. Note that an extension of .T_L is assumed and will be used even if a different extension is provided. This method is valid for Content Manager document items only.

Parameters: Path as VT_BSTR

Returns: VT_EMPTY

AddPart

The AddPart method adds a specified file as a part to the item. Set the SMSOption parameter to 0 (default resource manager). This method is valid for Content Manager document items only.

Parameters:

Path as VT_BSTR

MIMETYPE as VT_BSTR

PartType as VT_I2

SMSOption as VT_I4

TextSearchable as VT_BOOL

Returns: VT_EMPTY

PartType values are:

- OLEAPI_BASE (300)
- OLEAPI_BASSETEXT (301)
- OLEAPI_BASESTREAM (302)

AddToFolder

The AddToFolder method adds the item to the folder which is specified as an Item object. This method is valid for Content Manager documents and folder items only.

Parameters: Folder as VT_DISPATCH (Item)

Returns: VT_EMPTY

AddToResumeList

Adds an entry into resume list criteria for process activation. This method should be called each time you want to build resume criteria in the resume list. This method is valid for Content Manager documents and folder items only.

Parameters:

ItemTypeName as VT_BSTR

NumItems as VT_I4

Returns: VT_EMPTY

ChangeNotes

The ChangeNotes method saves the argument value as the note log. The existing note log is overwritten. This method is valid for Content Manager documents and folder items only.

Parameters:

Notes as VT_BSTR

Version as VT_BOOL

Returns: VT_EMPTY

ChangeProcess

The ChangeProcess method allows you to specify a new process to send the item through. The new process is specified by the process argument.

Parameters:

ProcessName as VT_BSTR

Priority as VT_I4

Returns: VT_EMPTY

CheckIn

The CheckIn method checks the item in, allowing anyone to modify it. This method is valid for Content Manager documents and folder items only.

Parameters: None

Returns: VT_EMPTY

CheckOut

The CheckOut method checks the item out to the current user, disabling anyone else from modifying it. This method is valid for Content Manager documents and folder items only.

Parameters: None

Returns: VT_EMPTY

Close The Close method unlocks the item previously locked.

Parameters: None

Returns: VT_EMPTY

CloseIt

Attention: The CloseIt method is the same as the Close method. It is implemented solely to support Visual Basic, which uses Close as a reserved word. The CloseIt method unlocks the item previously locked.

Parameters: None

Returns: VT_EMPTY

CloseNotes

The CloseNotes method closes the open note log without saving any changes. This method is valid for Content Manager documents and folder items only.

Parameters: None

Returns: VT_EMPTY

CloseParts

The CloseParts method closes all of the open part files (pages) without saving any changes. It closes the open note log without saving any changes. This method is valid for Content Manager document items only. When set to true, Delete removes part files from the work area. Otherwise, it closes the part files without deleting anything.

Parameters: Delete as VT_BOOL

Returns: VT_EMPTY

Delete The Delete method removes the item from the Content Manager server. This is a non-recoverable operation, so use this method with care. This method is valid for Content Manager documents and folder items only.

Parameters: None

Returns: VT_EMPTY

DeletePart

The DeletePart method deletes the specified base part from the item. This method is valid for Content Manager documents items only.

Parameters: Index as VT_I4

Returns: VT_EMPTY

GetAnnotationFile

The GetAnnotationFile method retrieves the annotation file for the item. The GetAnnotationFile method retrieves an annotation file from Content Manager, stores it on the local workstation, and returns the full path to the temporary file. This method is valid for Content Manager document items only.

Parameters: None

Returns: VT_BSTR

GetChildCompAttributeValueArray

The GetChildCompAttributeValueArray method returns the attribute value

array for the given attribute of a component. This method is valid for Content Manager documents and folder items only.

Parameters:

 ComponentName as VT_BSTR

 AttributeName as VT_BSTR

Returns: VT_VARIANT (safe array of VT_BSTR)

GetChildCompAttributeValueList

The GetChildCompAttributeValueList method returns the list of values for the given attribute of a component. Separated by separator argument. This method is valid for Content Manager documents and folder items only.

Parameters:

 ComponentName as VT_BSTR

 AttributeName as VT_BSTR

 Separator as VT_BSTR

Returns: VT_BSTR

GetHistorylog

The GetHistorylog method returns the history log.

Parameters: None

Returns: VT_BSTR

GetItemPID

The GetItemPID method returns the item PID.

Parameters: None

Returns: VT_BSTR

GetNotes

The GetNotes method returns the note log. This method is valid for Content Manager documents and folder items only.

Parameters: None

Returns: VT_BSTR

GetPartFile

The GetPartFile method retrieves a part file from Content Manager, stores it on the local workstation, and returns the full path to the temporary file. This method is valid for Content Manager document items only.

Parameters: Index as VT_I4

Returns: VT_BSTR

GetParentFolders

The GetParentFolders method returns an Items collection of folders to which this item belongs. This method is valid for Content Manager documents and folder items only.

Parameters: None

Returns: VT_DISPATCH (Items)

GetPartMIMETYPE

The GetPartMIMETYPE method returns the MIME type specified by the part index. This method is valid for Content Manager document items only.

Parameters: Index as VT_I4

Returns: VT_BSTR

GetRootCompAttributeValue

The GetRootCompAttributeValue method returns the root component attribute value specified by the attribute name. This method is valid for Content Manager documents and folder items only.

Parameters: AttributeName as VT_BSTR

Returns: VT_BSTR

GetVersionedItem

The GetVersionedItem method returns the specified version of the item as an Item object. This method is valid for Content Manager document items and folders.

Parameters: Version as VT_I4

Returns: VT_DISPATCH (Item)

GetTOCItem

The GetTOCItem method returns the Item object specified from the TOC. Valid for TOC Items object only.

Parameters: Index as VT_I4

Returns: VT_DISPATCH (Item)

IsValueRequired

The IsValueRequired method returns TRUE if a attribute value is required and FALSE if it is not. This method is valid for Content Manager document and folder items only.

Parameters:

 ComponentName as VT_BSTR

 AttributeName as VT_BSTR

Returns: VT_BOOL

Lock The Lock method locks the item. No other user can modify item type information or modify parts when the item is locked. You must use the Close or CloseIt methods to unlock the item.

Parameters: None

Returns: VT_EMPTY

NextWorklistItem

The NextWorklistItem method returns the next available item in a worklist. The NextWorklistItem method returns a dispatch pointer to Item object which is the next worklist item. This method is valid for worklist Item objects only.

Parameters: None

Returns: VT_DISPATCH (Item)

PrintItem

The PrintItem method prints the item to the currently selected printer using the current print options. If ShowDialog is true, the print window displays where the user can select a different printer, modify options, or cancel printing.

Parameters:

ShowDialog as VT_BOOL
PrintImage as VT_BOOL
StartPage as VT_I4
EndPage as VT_I4
PrintMarkup as VT_BOOL
PrintIndex as VT_BOOL
PrintNoteLog as VT_BOOL
PrintTOC as VT_BOOL
PrintContents as VT_BOOL

Returns: VT_EMPTY

PrintImage specifies whether or not to print the base parts of the document, also known as images.

If you also select *PrintMarkup* any defined annotations are printed on the image.

StartPage and *EndPage* (optional) specify the desired base part page ranges to print. The pages are numbered starting from 1.

For example, to print the middle three pages of a five-page document:

1. Set *StartPage* to 2.
2. Set *EndPage* to 4.

To print an entire document:

1. Set *StartPage* to 1.
2. Set *EndPage* to 10000 or some other sufficiently large number.

PrintIndex and *PrintNotelog* allow you to specify whether the indexing and note log information prints for documents and folders. Worklists ignore these arguments however, you can print the note logs and index information for the documents and folders contained in the worklist by setting *PrintIndex* and *PrintNotelog* to true in worklists.

PrintTOC and *PrintContents* specify how to print worklists and folders. If *PrintTOC* is true, the list of items contained in the folder or worklist prints. If *PrintContents* is true, the folders and documents contained in the table of contents prints as well.

RefreshTOC

The RefreshTOC method re-samples the TOC of a worklist or folder. If you did not call this method, any changes to a worklist or folder's TOC is not recognized by methods in the Item class. Valid for TOC Items object only.

Parameters: None

Returns: VT_EMPTY

RemoveFromFolder

The RemoveFromFolder method removes the item from the folder specified as an argument. This method is valid for Content Manager documents and folder items only.

Parameters: Folder as VT_DISPATCH (Item)

Returns: VT_EMPTY

RemoveFromProcess

The RemoveFromProcess method removes the item from the process. This method is valid only if the item is process-aware. That is, the item was obtained using the Application object Search method with the SuspendFilter, ProcessName, or StepName parameters specified or the item was obtained from a worklist Item object.

Parameters: None

Returns: VT_EMPTY

SaveAnnotation

SaveAnnotation method saves any updates to the annotation file. This method is valid for Content Manager document items only.

Parameters: Version as VT_BOOL

Returns: VT_EMPTY

SavePart

The SavePart method saves any changes that occurred to the base part file specified by the index. This method is valid for Content Manager documents items only.

Parameters:

Index as VT_I4

Version as VT_BOOL

Returns: VT_EMPTY

SetChildCompAttributeValue

SetChildCompAttributeValue method sets the child component attribute value specified by the attribute name and MultiValueIndex. For example, if a child component has three values, you would set:

MultiValueIndex=0

MultiValueIndex=1

MultiValueIndex=2

This method is valid for Content Manager documents and folder items only.

Parameters:

ComponentName as VT_BSTR

AttributeName as VT_BSTR

MultiValueIndex as VT_I2

AttributeValue as VT_BSTR

Returns: VT_EMPTY

Important: Update method must be called in order for this method to take effect.

SetChildCompMultiValueCount

The SetChildCompMultiValueCount method sets how many values each attribute has for the specified component. This method is valid for Content Manager documents and folder items only.

Parameters:

ComponentName as VT_BSTR

MultiValueIndex as VT_I2

Returns: VT_EMPTY

Important: Update method must be called in order for this method to take effect.

SetRootCompAttributeValue

Sets the root component attribute value specified by the AttributeName. This method is valid for Content Manager documents and folder items only.

Parameters:

AttributeName as VT_BSTR

AttributeValue as VT_BSTR

Returns: VT_EMPTY

Important: Update method must be called in order for this method to take effect.

StartOnProcess

The StartOnProcess method adds the item into the specified process. This method is valid for Content Manager documents and folder items only.

Parameters:

ProcessName as VT_BSTR

Priority as VT_I4

Returns: VT_EMPTY

Suspend

The Suspend method causes the item to be suspended, pending some future events. The events can be a specified time interval or the addition of items to a folder item. The resume list criteria for the folder item can be set using AddToResumeList method.

Parameters: Interval as VT_I4

Returns: VT_EMPTY

UpdateItem

The UpdateItem method updates the item. Call this method to update the item with values set by SetChildCompAttributeValue(...), SetChildCompMultiValueCount(...), SetRootCompAttributeValue(...) and ItemType property. This method is valid for Content Manager documents and folder items only.

Parameters: NewVersion as VT_BOOL

Returns: VT_EMPTY

VersionsArray

The VersionsArray method returns a safe array of VT_BSTR containing the numbers of all the Versions of this item. This method is valid for Content Manager documents and folder items only.

Parameters: None

Returns: VT_VARIANT (safe array of VT_BSTR)

VersionsList

The VersionsList method returns a string with a list of all Version numbers that this item has. The Versions are separated by the string separator argument.

Parameters: Separator as VT_BSTR

Returns: VT_BSTR

Items collection object

The Items collection object holds a list of Item objects, allowing you to access the contained objects. The contained objects are item objects.

Items collection object properties

Application

The Application property returns the Application object.

Data Type: VT_DISPATCH (Application)

This property is read-only.

Count The Count property returns the number of Item objects referenced in the Items collection.

Data Type: VT_I4

This property is read-only.

Parent The Parent property returns the parent of the Items collection (which is usually a Document object).

Data Type: VT_DISPATCH (Document)

This property is read-only.

Items collection object methods

Close The Close method deletes all of the Items in the collection.

Parameters: None.

Returns: VT_EMPTY

CloseIt

Attention: The CloseIt method is the same as the Close method. It is implemented solely to support Visual Basic which uses Close as a reserved word. The CloseIt method closes the Items collection.

Parameters: None

Returns: VT_EMPTY

Item The Item method returns an Item object specified by the Index.

Parameters: Index as VT_I4

Returns: VT_DISPATCH (Item)

Chapter 3. Client for Windows user exit routines

The Content Manager Client for Windows user exit routines are specific points in the program where the client user can specify their own processing routines to enhance or replace the default Content Manager Client for Windows functions. This section describes how to use and implement user exit routines.

For more information about DK constants and datastores, see the Application Programming Guide.

Alternate search

Format
SHORT _cdecl AlternateSearchUserExit (<i>PVOID pDs, HWND hWnd, PSZ pszUserID, USHORT usNumCriteria, PSEARCHCRITERIASTRUCT pCriteria, PVOID pCollectionSearchResult</i>)

Alternate search purpose

Use the Alternate Search user exit to replace the search function of the Client for Windows program with your own search routine. The exit routine returns a dkCollection of DKDDOs. See the Application Programming Guide for more information about DK constants and datastores.

Alternate search parameters

pDs PVOID - Input

Input Session handle (void* to DKDatastoreICM) returned by DKDatastoreICM::connect.

This variable initialized during client logon.

hWnd HWND - Input

The handle to the window. You can use this parameter to display messages and associate them with the application window.

pszUserID

PSZ - Input

The 0-terminated character string containing the user ID of the user who receives the search results. This parameter is not case-sensitive.

usNumCriteria

USHORT - Input

The number of elements in pCriteria array.

pCriteria

PSEARCHCRITERIASTRUCT - Input

The pointer to a SEARCHCRITERIASTRUCT array.

pCollectionSearchResult

PVOID - Output

The pointer to dkCollection of DKDDO from the search results.

Internal representation

SEARCHCRITERIASTRUCT:

```
typedef struct _SEARCHCRITERIASTRUCT
{
    ULONG        ulStruct;
    ULONG        ulReturnLimit;
    CHAR         szSearchString[MAX_SEARCH_STRING_LEN];
    USHORT       usViewID;
    BOOL         bRetrieveLatestVersion;
} SEARCHCRITERIASTRUCT, *PSEARCHCRITERIASTRUCT;
```

Where

ulStruct

Size of the structure including this field

ulReturnLimit

Maximum return size of the view, or each view.

szSearchString

Search String

usViewID

ItemTypeView ID or zero (0) in case of all view search.

bRetrieveLatestVersion:

Flag to indicate if the latest or all versions needs to be retrieved.

Return values

The exit routine should return zero (0) to indicate that the search operation completed normally. All other return values indicate an abnormal ending and are logged as errors.

Comments

The Alternate Search user exit routine works at the view level. When running a basic search, if the search is against a particular view, the Client for Windows program loads the exit routine for that view. If performing an Advanced Search or a search against all views, the Content Manager Client for Windows program loads the exit routine for the base view of the NOINDEX.

For an example of this user exit, go to the SAMPLES directory and see the AlternateSearch() method in the ICMClientUEAltSrch.cpp file.

Attention: Use the system administration client to specify the function and DLL name for this user exit routine. Enter this information in the **Search** field of the User Exits page in the Item Type Definition window.

Query sort

Format
USHORT _cdecl QuerySortUserExit(PVOID pDs, HWND hWnd, PUSERSORTSTRUCT pSortList, USHORT usItemCount, PSZ pszUserId, USHORT usSortObject)

Query sort purpose

This user exit routine is called when a folder or worklist is opened that contains an item for which a exit is defined on the respective item type.

You can program this exit routine to sort the contents of the folder or worklist before displaying. You can define a specific sort order other than the default ascending or descending order provided by Content Manager. You can also use the exit routine to filter out selected documents and folders to prevent display and user access to those objects.

You can assign this user exit routine on an item type basis using the Content Manager system. Each item represented in a folder or worklist table of contents is sorted according to the user exit routine specified for that item type. If more than one item type is in the folder or worklist then each exit routine is called and completed sequentially prior to the displaying the item list.

Query sort parameters

pDs PVOID - Input

Input Session handle (void* to DKDatastoreICM) returned by DKDatastoreICM::connect.

This variable initialized during client logon.

hWnd HWND - Input

The handle to a window. You can use this parameter to display messages and associate them with the application window.

pSortList

PUSERSORTSTRUCT - Input/Output

Pointer to an array of documents and folders to be sorted. Each document or folder is represented by a USERSORTSTRUCT data structure.

usItemCount

USHORT - Input

The number of items count that is passed in pSortList.

pszUserID

PSZ - Input

User ID name of the user that is logged on.

usSortObject

USHORT - Input

Type of object type of the sort list. The valid values are:

- ICMCLIENT_FOLDER (2)
- ICMCLIENT_WORKLIST (4)

Internal representation

ATTRIBUTE:

This structure used to expose the attribute values to the user exit routine.

```
typedef struct _ATTRIBUTE
{
    KEYINFO    attrKeyInfo;
    USHORT     usDataType;
    USHORT     usMaxLength;
    USHORT     usMinLength;
    USHORT     usValueCount;
    PSZ        *ppszValues;
}ATTRIBUTE, *PATATTRIBUTE;
```

Where:

attrKeyInfo

KEYINFO structure containing attribute key information.

usDataType

Attribute data type. The data type can be any of the following constants:

DK_CM_UNDEFINED
DK_CM_VARCHAR
DK_CM_CHAR
DK_CM_SHORT
DK_CM_LONG
DK_CM_FLOAT
DK_CM_DATE
DK_CM_TIME
DK_CM_TIMESTAMP
DK_CM_DOUBLE
DK_CM_BLOB
DK_CM_CLOB

usMaxLength

MaxValue/MaxLength condition based on data type.

usMinLength

MinValue/MinLength condition based on data type.

usValueCount

Count of values in the ppszValues array.

ppszValues

Pointer to an array of null terminated string values for an attribute. For example, if *usValueCount* is 3, *ppszValues* will have three null terminated strings.

COMPONENTSTRUCT:

This structure used to expose the component hierarchy to the user exit routine.

```
typedef struct _COMPONENTSTRUCT
{
    KEYINFO      compViewKeyInfo;
    USHORT       usAttrCount;
    PATTRIBUTE   pAttrs;
    USHORT       usChildCompCount;
    struct _COMPONENTSTRUCT *pChildComps;
}COMPONENTSTRUCT, *PCOMPONENTSTRUCT;
```

Where:

compViewKeyInfo

KEYINFO structure containing component view key information.

usAttrCount

The number of attributes that the component has.

pAttrs ATTRIBUTE structure array containing the attribute values.

usChildCompCount

Number of child components.

pChildComps

COMPONENTSTRUCT structure array containing child components.

DOCROUTINGSTRUCT:

This structure used to expose document routing information associated to the item.

```
typedef struct _DOCROUTINGSTRUCT
{
    USHORT    usSuspended;
    CHAR      szProcess[MAX_DISPLAY_NAME_LEN];
    CHAR      szLastMovedTimeStamp[MAX_TIMESTAMP_LEN];
    CHAR      szWorkpackagePid[MAX_WPID_LEN];
    CHAR      szStep[MAX_DISPLAY_NAME_LEN];
    ULONG     ulReserved;
    CHAR      szOwner[MAX_DISPLAY_NAME_LEN];
    long      lPriority;
}DOCROUTINGSTRUCT, *PDOCROUTINGSTRUCT;
```

Where:

usSuspended

Specified if the item is in suspended state or not. 1=Suspended, 0=Active.

szProcess

Name of the process that the item is in.

szLastMovedTimeStamp

Item last moved time stamp.

szWorkpackagePid

Work package id of the item.

szStep The location name of the work node or collection point in which the item is in.

szOwner

The owner of the item.

lPriority

Item priority in the process.

KEYINFO:

```
typedef struct _KEYINFO
{
    int      id;
    CHAR      szName[MAX_NAME_LEN];
    CHAR      szDisplayName[MAX_DISPLAY_NAME_LEN];
}KEYINFO, *PKEYINFO;
```

Where:

id ID of an attribute or a component view.

szName

Internal name of the attribute or component view.

szDisplayName

Display name of the attribute or component view.

USERSORTSTRUCT:

This structure used to expose an item to the user exit routine.

```
typedef struct _USERSORTSTRUCT
{
    CHAR                szPIDString[PID_STRING_LEN];
    USHORT              usItemType;
    DOCROUTINGSTRUCT    docRoutAttrs;
    PCOMPONENTSTRUCT    pRootComponent;
    USHORT              usFlags;
}USERSORTSTRUCT, *PUSERSORTSTRUCT;
```

Where:

szPIDString
PID representing the item.

usItemType
The item type name to which the item belongs.

docRoutAttrs
DOCROUTINGSTRUCT structure containing document routing information.

pRootComponent
COMPONENTSTRUCT structure pointer to the root component.

usFlag Flag used by the user exit routine to hide or display the item. The item is not displayed if it is set to ICMCLIENT_HIDE or else it is displayed by default or explicitly specify to display by setting it to ICMCLIENT_SHOW.

Return values

The function returns a value of type SHORT. It should return a value of zero (0) for successful completion of the user exit routine. If the user exit completes successfully, the items appear in the order in which they are sorted in the USERSORTSTRUCT array (pSortList [0], pSortList [1]). If the user exit fails with a nonzero return value, then the items are ordered as if the sort user exit had not been called. If the usFlags parameter is set to ICMCLIENT_HIDE, the document or folder does not display in the TOC.

Comments

The user exit routine must not free the buffers that are passed in. It cannot change the layout of the item list, nor should it modify attribute values. Items that are given the ICMCLIENT_HIDE flag do not appear. System-assigned worklists do not use this user exit. This function is processed prior to the display of the list. If the user re-sorts the view by clicking on the attribute name in the TOC view, the user's sort request overrides the results returned by this user exit.

For an example of this user exit, go to the SAMPLES directory and see the QuerySort() method in the ICMClientUEQrySort.cpp file.

Attention: Use the system administration client to specify the function and DLL name for this user exit routine. Enter this information in the **Sort** field of the User Exits page in the Item Type Definition window.

Save record

Format
USHORT _cdecl SaveRecordUserExit(HWND hWnd, USEREXITSTRUCT* pPreSaveStruct)

Save record purpose

This user exit routine is called when a user chooses to save changes to the user-defined attributes of a document or folder. The item attribute fields are passed to the exit routine for processing. For example, the new attribute data entered in the Client for Windows main window can be validated by matching the information in your existing files.

This exit routine also allows changes to the user-defined attribute fields. If the fields are modified by the exit routine, they are audited by the Content Manager system before the record is written to the database. The audits compare the data returned using the following guidelines:

Data type

The format and content of the data must conform to the requirements of the data type for the attribute.

Minimum length

The minimum length requirement of the data string, or minimum numeric value for certain data types, must be met if specified for the attribute.

Maximum length

The maximum length requirement of the data string, or maximum numeric value for certain data types, must be met if specified for the attribute.

Save record parameters

hWnd HWND - Input

The handle to a window. You can use this parameter to display messages and associate them with the application window.

pPreSaveStruct

USEREXITSTRUCTURE* - Input/Output

User-defined attributes for the document or folder and other relevant information are passed in the *pPreSaveStruct* parameter.

Internal representation

ATTRIBUTE:

This structure used to expose the attribute values to the user exit routine.

```
typedef struct _ATTRIBUTE
{
    KEYINFO      attrKeyInfo;
    USHORT       usDataType;
    USHORT       usMaxLength;
    USHORT       usMinLength;
    USHORT       usValueCount;
    PSZ          *ppszValues;
}ATTRIBUTE, *PATATTRIBUTE;
```

Where:

attrKeyInfo

KEYINFO structure containing attribute key information.

usDataType

Attribute data type. The data type can be any of the following constants:

DK_CM_UNDEFINED

DK_CM_VARCHAR

DK_CM_CHAR
 DK_CM_SHORT
 DK_CM_LONG
 DK_CM_FLOAT
 DK_CM_DATE
 DK_CM_TIME
 DK_CM_TIMESTAMP
 DK_CM_DOUBLE
 DK_CM_BLOB
 DK_CM_CLOB

usMaxLength

MaxValue/MaxLength condition based on data type.

usMinLength

MinValue/MinLength condition based on data type.

usValueCount

Count of values in the ppszValues array.

ppszValues

Pointer to an array of null terminated string values for an attribute. For example, if *usValueCount* is 3, *ppszValues* will have three null terminated strings.

COMPONENTSTRUCT:

This structure used to expose the component hierarchy to the user exit routine.

```
typedef struct _COMPONENTSTRUCT
{
    KEYINFO      compViewKeyInfo;
    USHORT       usAttrCount;
    PATTRIBUTE   pAttrs;
    USHORT       usChildCompCount;
    struct _COMPONENTSTRUCT *pChildComps;
}COMPONENTSTRUCT, *PCOMPONENTSTRUCT;
```

Where:

compViewKeyInfo

KEYINFO structure containing component view key information.

usAttrCount

The number of attributes that the component has.

pAttrs ATTRIBUTE structure array containing the attribute values.

usChildCompCount

Number of child components.

pChildComps

COMPONENTSTRUCT structure array containing child components.

KEYINFO:

```
typedef struct _KEYINFO
{
    int      id;
    CHAR     szName[MAX_NAME_LEN];
    CHAR     szDisplayName[MAX_DISPLAY_NAME_LEN];
}KEYINFO, *PKEYINFO;
```

Where:

id ID of an attribute or a component view.

szName
Internal name of the attribute or component view.

szDisplayName
Display name of the attribute or component view.

USEREXITSTRUCT:

```
typedef struct _USEREXITSTRUCT
{
    PVOID                pDs;
    CHAR                 szPIDString[PID_STRING_LEN];
    CHAR                 szProcessName[MAX_DISPLAY_NAME_LEN];
    KEYINFO              compOrigViewKeyInfo;
    CHAR                 szUserId[MAX_NAME_LEN];
    USHORT               usAccessLevel;
    PCOMPONENTSTRUCT     pRootComponent;
} USEREXITSTRUCT, *PUSEREXITSTRUCT;
```

Where:

pDs pDs Session handle returned by DKDatastoreICM::connect

szPIDString
szPIDString is the item's PID String of the current document or folder to be saved.

szProcessName
Process name of currently opened item. This parameter might be NULL for a process unaware item.

compOrigViewKeyInfo
Pointer to the KEYINFO structure describing the attribute information of the item or folder to be saved.

szUserId
Identifies the user ID of the user saving the document or folder.

usAccessLevel
States the access privilege the user has for this document or folder. The value is NULL if the object is not opened by this user. The valid values are:

UX_PRIV_READ when the user opens this object in browse mode.

UX_PRIV_WRITE when the user opens this object in update mode.

pRootComponent
Pointer to the Root Component of the Item or folder. The valid value for this user exit routine is UX_PRIV_WRITE when the user opens this object in UPDATE mode.

Return values

The function returns a value of type SHORT. It should return a value of zero (0) for successful completion of the user exit routine. If any other value is returned, the Save operation is ended and an error message appears.

Comments

The exit routine must not free the buffers that are passed in. All items sent to the exit are read-only copies except the user-defined attributes in the ATTRIBUTE data

structures. This user exit is called when a document or folder is saved after modifying the user-defined attributes. It is called prior to the Change SMS user exit routine if both are specified for the current item type. Validation of the user-defined attribute fields is performed after the user exit routine is completed. This user exit is not processed if the user has not changed the item type or attributes of the object.

For an example of this user exit, go to the SAMPLES directory and see the SaveRecord() method in the ICMClientUESaveRec.cpp file.

Attention: Use the system administration client to specify the function and DLL name for this user exit routine. Enter this information in the **Save** field of the User Exits page in the Item Type Definition window.

Change SMS

Format
USHORT _cdecl ChangeSMSUserExit (HWND hWnd, USEREXITSTRUCT* pExitStruct, PBOOL pfContinue)

Change SMS purpose

This user exit routine is called whenever the item type is changed for an item. The exit routine is passed information about the item and returns a flag indicating whether default processing should continue. The default processing would change the System-Managed Storage (SMS) information for each of the item's parts using the object server and collection information defined in the item's new item type.

Change SMS parameters

hWnd HWND - Input

The handle to a window. You can use this parameter to display messages and associate them with the application window.

pExitStruct

PUSEREXITSTRUCT - Input

User-defined attribute fields and other relevant information for the open document are passed in the pExitStruct parameter.

pfContinue

PBOOL - Input

This is a pointer to the continue flag. Set to value to TRUE to continue with default processing.

Internal representation

ATTRIBUTE:

This structure used to expose the attribute values to the user exit routine.

```
typedef struct _ATTRIBUTE
{
    KEYINFO    attrKeyInfo;
    USHORT     usDataType;
    USHORT     usMaxLength;
```

```

USHORT    usMinLength;
USHORT    usValueCount;
PSZ       *ppszValues;
}ATTRIBUTE, *PATATTRIBUTE;

```

Where:

attrKeyInfo

KEYINFO structure containing attribute key information.

usDataType

Attribute data type. The data type can be any of the following constants:

```

DK_CM_UNDEFINED
DK_CM_VARCHAR
DK_CM_CHAR
DK_CM_SHORT
DK_CM_LONG
DK_CM_FLOAT
DK_CM_DATE
DK_CM_TIME
DK_CM_TIMESTAMP
DK_CM_DOUBLE
DK_CM_BLOB
DK_CM_CLOB

```

usMaxLength

MaxValue/MaxLength condition based on data type.

usMinLength

MinValue/MinLength condition based on data type.

usValueCount

Count of values in the ppszValues array.

ppszValues

Pointer to an array of null terminated string values for an attribute. For example, if *usValueCount* is 3, *ppszValues* will have three null terminated strings.

COMPONENTSTRUCT:

This structure used to expose the component hierarchy to the user exit routine.

```

typedef struct _COMPONENTSTRUCT
{
    KEYINFO    compViewKeyInfo;
    USHORT     usAttrCount;
    PATATTRIBUTE pAttrs;
    USHORT     usChildCompCount;
    struct _COMPONENTSTRUCT *pChildComps;
}COMPONENTSTRUCT, *PCOMPONENTSTRUCT;

```

Where:

compViewKeyInfo

KEYINFO structure containing component view key information.

usAttrCount

The number of attributes that the component has.

pAttrs ATTRIBUTE structure array containing the attribute values.

usChildCompCount
Number of child components.

pChildComps
COMPONENTSTRUCT structure array containing child components.

KEYINFO:

```
typedef struct _KEYINFO
{
    int      id;
    CHAR     szName[MAX_NAME_LEN];
    CHAR     szDisplayName[MAX_DISPLAY_NAME_LEN];
}KEYINFO, *PKEYINFO;
```

Where:

id ID of an attribute or a component view.

szName
Internal name of the attribute or component view.

szDisplayName
Display name of the attribute or component view.

USEREXITSTRUCT:

```
typedef struct _USEREXITSTRUCT
{
    PVOID          pDs;
    CHAR           szPIDString[PID_STRING_LEN];
    CHAR           szProcessName[MAX_DISPLAY_NAME_LEN];
    KEYINFO        compOrigViewKeyInfo;
    CHAR           szUserId[MAX_NAME_LEN];
    USHORT         usAccessLevel;
    PCOMPONENTSTRUCT pRootComponent;
}USEREXITSTRUCT, *PUSEREXITSTRUCT;
```

Where:

pDs pDs Session handle returned by DKDatastoreICM::connect

szPIDString
szPIDString is the item's PID String of the current document or folder to be saved.

szProcessName
Process name of currently opened item. This parameter might be NULL for a process unaware item.

compOrigViewKeyInfo
Pointer to the KEYINFO structure describing the attribute information of the item or folder to be saved.

szUserId
Identifies the user ID of the user saving the document or folder.

usAccessLevel
States the access privilege the user has for this document or folder. The value is NULL if the object is not opened by this user. The valid values are:

UX_PRIV_READ when the user opens this object in browse mode.

UX_PRIV_WRITE when the user opens this object in update mode.

pRootComponent

Pointer to the Root Component of the Item or folder. The valid value for this user exit routine is UX_PRIV_WRITE when the user opens this object in UPDATE mode.

Return values

The function returns a value of type USHORT. It should return a value of zero for successful completion of the user exit routine. If any value other than zero is returned, default processing occurs. If the call is successful, the value returned in the pfContinue parameter is checked.

Comments

The user exit routine must not free the buffers that are passed in. All items sent to the exit routine are read-only copies. This exit routine must not modify these data structures. The index form is closing when this user exit routine is called. If a class has both the Save Record and Change SMS user exit routines specified, the Save Record user exit routine is called first.

For an example of this user exit, go to the SAMPLES directory and see the ChangeSMS() method in the ICMClientUEChgSMS.cpp file.

Attention: Use the system administration client to specify the function and DLL name for this user exit routine. Enter this information in the **Resource manager storage** field of the User Exits page in the Item Type Definition window.

Get attribute value list

Format

short _cdecl GetAttributeValueList (PVOID pDs, PSZ pszViewName, PSZ pszAttrName, int nDialogType, int *pControlType, int *pSortOption, PSZ *ppszListValues, int *Values, int nMaxValueLen)

Get attribute value list purpose

This user exit routine allows you to extend the Attributes and Search windows to include a combination list box. The actual values to be listed are returned by this user exit routine.

This user exit routine must be contained in a Dynamic Library Link (DLL), named ICMClientUEAttrVals.dll. This user exit routine is called for each attribute ID in an item type (root and child). This routine returns the list of values for the combination list box.

A sample of this user exit is located in the SAMPLES directory under the product install directory.

Get attribute value list parameters

pDs PVOID - Input

Input Session handle (void* to DKDatastoreICM) returned by DKDatastoreICM::connect.

This variable initialized during client logon.

pszViewName

PSZ - Input

The internal name of the current item type view being displayed.

pszAttrName

PSZ - Input

AttributeName string

nDialogType

int - Input

Dialog type. It returns one of the following dialogs:

- 0 for the Attribute dialog
- 1 for the Search dialog

pControlType

int * - Output

Returns one of the following:

- 0 for a standard entry field
- 1 for a list box that allows text entry
- 2 for a static list box

Important: For root attributes, control type 1 produces an editable combination list box, and control type 2 produces a static combination list box. For child attributes, both control type 1 and control type 2 produces a static combination list box.

pSortOption

int * - Output

Returns one of the following:

- 0 to leave the combo values in the order returned
- 1 to have the list sorted alphabetically

ppszListValues

PSZ * - Output

An array of character pointers, each pointing to a zero-terminated string representing one of the values that will be displayed in the list box. If the field is a standard edit field (*pControlType=0), this array should have the size of 1. This also means that GetValueListLength, below, should return 1 in *pNumValues.

Restriction: Do not fill in more values than are specified in pNumValues, below.

pNumValues

int * - Input and Output

On input, this parameter contains the number of values that was returned in pNumValues from the GetValueListLength() function. This value can be left alone or decreased if fewer values are actually used. This value must not be increased.

nMaxValueLen

int - Input

The value that is returned through pMaxValueLen in GetValueListLength().

Return values

Zero (0) for success; non-zero for error.

Comments

You should make sure that this user exit routine runs quickly because it is called for every field in the Attributes and Basic Search dialogs.

Get value list length

Format
short _cdecl GetValueListLength (PVOID <i>pDs</i> , PSZ <i>pszViewName</i> , PSZ <i>pszAttrName</i> , int <i>nDialogType</i> , int * <i>pNumValues</i> , int * <i>pMaxValueLen</i>)

Get value list length purpose

This user exit routine returns the number of values and the maximum value length for the specified attribute ID, from the specified item type. The client calls this function for every field of every item type (root and child) to determine if there is a list of values for the attribute and if there is, how much space to allocate for it. As a result, the user exit routine returns the number of values and the maximum value length for the specified attribute ID in the specified item type or item type view.

This user exit routine must be contained in a DLL named ICMClientUEAttrVals.dll.

A sample of this user exit is located in the SAMPLES directory under the product install directory.

Get value list length parameters

pDs PVOID - Input

Input Session handle (void* to DKDatastoreICM) returned by DKDatastoreICM::connect.

This variable initialized during client logon.

pszViewName

PSZ - Input

The internal name of the current item type view being displayed.

pszAttrName

PSZ - Input

AttributeName string

nDialogType

int - Input

Dialog type. It returns one of the following dialogs:

- 0 for the Attribute dialog
- 1 for the Search dialog

pNumValues

int * - Output

The default is zero (0) (*pNumValues=0). If you provide values for this attribute, then you must set pNumValues to the amount of values that you expect the list to have.

pMaxValueLen

int* - Output

The default is zero (0) (*pMaxValueLen=0). If you provide values for this attribute, then you must set pMaxValueLen to the maximum length of the longest value in the list.

Return values

Zero (0) for success; non-zero for error.

Comments

You should make sure that this user exit routine runs quickly because it is called for every field in the Attributes and Basic Search dialogs.

Add items to actions menu

Format
INT _cdecl UserOptionList(PUSEROPTIONSTRUCT pUO, INT *pNumUO)

Add items to actions purpose

This user exit routine enables you to add items to the **Actions** menu. This user exit routine works in conjunction with the UpdateFunc (Enable or disable menu items) and the UserOptFunc (User option function).

This user exit routine provides a list of all the user options supported in the environment. This user exit routine must be contained in a DLL named ICMClientUEUserOpts.dll. A sample of this user exit routine is located in the SAMPLES directory under the product install directory.

Add items to actions parameters

pUO PUSEROPTIONSTRUCT - Output

The client passes an array of USEROPTIONSTRUCTs in this parameter. The user exit routine provides the desired number of menu items up to a maximum of 10.

pNumUO

INT * - Output

Initially, the default is zero (0) (*pNumUO=0). Set this to the number of USEROPTIONSTRUCTs that were specified in the pUO parameter.

Internal representation

USEROPTIONSTRUCT:

This structure is defined in ICMClientUEDefs.h.

```
typedef struct _USEROPTIONSTRUCT
{
    CHAR    szUserOptDLL[USEROPTIONLENGTH];
```

```

CHAR    szUserOptFunc[USEROPTIONLENGTH];
CHAR    szUpdateFunc[USEROPTIONLENGTH];
CHAR    szMenuName[USEROPTIONLENGTH];
} USEROPTIONSTRUCT, *PUSEROPTIONSTRUCT;

```

Where:

szUserOptDLL

The DLL containing the UserOptFunc (User option function) and UpdateFunc (Enable or disable menu items) user exit routines.

szUserOptFunc

The name of the UserOptFunc (see “User option function” on page 61).

szUpdateFunc

The name of the UpdateFunc (see “Enable or disable menu items”).

szMenuName

The name of the menu item as it is to appear in the Actions menu.

Return values

Zero (0) for success; non-zero for error. If an error is returned, no user option exit routine are supported; errors are logged in the vic.err file.

Comments

This function is called once when the first time the Actions menu is displayed.

Enable or disable menu items

Format
INT _cdecl UpdateFunc (HWND hwnd, PUSEREXITSTRUCT pUES, ITEMID strWLItemID, PSZ szUserPriv, INT *pnStatus)

Enable or disable purpose

Each time the user selects the **Actions** menu this user exit routine is called to determine if a menu item should be enabled or disabled, based on the item with selection focus, worklist itemid and the user’s privilege string. The item with selection focus is the selected item within a highlighted list of items. If the user exit routine results indicate that the menu item is not available, the menu item is disabled. A sample of this function is located in the SAMPLES directory under the product install directory.

Enable or disable parameters

hwnd HWND - Input

Anchor window for message. Use this window to display messages and associate them with the application window.

pUES PUSEREXITSTRUCT - Input

A pointer to the item with selection focus.

strWLItemID

ITEMID - Input

PID String of the current Worklist. This parameter is empty if the TOC shown is not a Worklist.

szUserPriv

PSZ - Input

User privilege string.

pnStatus

INT * - Output

Indicates whether a menu item is enabled (1) or disabled (0).

Constants:

- UX_PRIV_READ 1
- UX_PRIV_WRITE 2
- MAX_NAME_LEN 33
- MAX_DISPLAY_NAME_LEN 256
- MAX_VALUE_LEN 128
- MAX_SEARCH_STRING_LEN 256
- MAX_TIMESTAMP_LEN 128
- MAX_WPID_LEN 128
- ICMCLIENT_HIDE 1
- ICMCLIENT_SHOW 2

Internal representation

COMPONENTSTRUCT:

This structure used to expose the component hierarchy to the user exit routine.

```
typedef struct _COMPONENTSTRUCT
{
    KEYINFO      compViewKeyInfo;
    USHORT       usAttrCount;
    PATTRIBUTE   pAttrs;
    USHORT       usChildCompCount;
    struct _COMPONENTSTRUCT *pChildComps;
}COMPONENTSTRUCT, *PCOMPONENTSTRUCT;
```

Where:

compViewKeyInfo

KEYINFO structure containing component view key information.

usAttrCount

The number of attributes that the component has.

pAttrs ATTRIBUTE structure array containing the attribute values.

usChildCompCount

Number of child components.

pChildComps

COMPONENTSTRUCT structure array containing child components.

USEREXITSTRUCT:

```
typedef struct _USEREXITSTRUCT
{
    PVOID          pDs;
    CHAR           szPIDString[PID_STRING_LEN];
}
```

```

CHAR                szProcessName[MAX_DISPLAY_NAME_LEN];
KEYINFO             compOrigViewKeyInfo;
CHAR                szUserId[MAX_NAME_LEN];
USHORT              usAccessLevel;
PCOMPONENTSTRUCT    pRootComponent;
}USEREXITSTRUCT, *PUSEREXITSTRUCT;

```

Where:

pDs pDs Session handle returned by DKDatastoreICM::connect

szPIDString

szPIDString is the item's PID String of the current document or folder to be saved.

szProcessName

Process name of currently opened item. This parameter might be NULL for a process unaware item.

compOrigViewKeyInfo

Pointer to the KEYINFO structure describing the attribute information of the item or folder to be saved.

szUserId

Identifies the user ID of the user saving the document or folder.

usAccessLevel

States the access privilege the user has for this document or folder. The value is NULL if the object is not opened by this user. The valid values are:

UX_PRIV_READ when the user opens this object in browse mode.

UX_PRIV_WRITE when the user opens this object in update mode.

pRootComponent

Pointer to the Root Component of the Item or folder. The valid value for this user exit routine is UX_PRIV_WRITE when the user opens this object in UPDATE mode.

Return values

Zero (0) for success; non-zero for error.

Comments

A non-zero return code is logged and disables the menu item. This user exit routine is called each time a contextual menu is called so it must work quickly.

User option function

Format
INT _cdecl UserOptFunc(HWND hwnd, PSUSEREXITSTRUCT pUES, INT nUES, ITEMID strWLItemID, PSZ szUserPriv)

User option function purpose

This user exit routine is called when a user selects a menu item specified in the UserOptionList function see “Add items to actions menu” on page 58. An array of all the selected items, a worklist, if any, and user privileges are passed to this function. A sample of this function is located in the SAMPLES directory under the product install directory.

User option function parameters

hwnd HWND - Input

Anchor window for message. Use this window to display messages and associate them with the application window.

pUES

PUSEREXITSTRUCT - Input

pUES is an array of user exit routine structures, one structure for each selected item.

nUES

INT - Input

nUES is the number of USEREXITSTRUCTs in the pUES structure.

strWListItemID

ITEMID - Input

The item PID string for the worklist. This parameter is empty if the containing TOC for the selected item is not a worklist.

szUserPriv

PSZ - Input User privilege string.

Internal representation

COMPONENTSTRUCT:

This structure used to expose the component hierarchy to the user exit routine.

```
typedef struct _COMPONENTSTRUCT
{
    KEYINFO      compViewKeyInfo;
    USHORT       usAttrCount;
    PATTRIBUTE   pAttrs;
    USHORT       usChildCompCount;
    struct _COMPONENTSTRUCT *pChildComps;
} COMPONENTSTRUCT, *PCOMPONENTSTRUCT;
```

Where:

compViewKeyInfo

KEYINFO structure containing component view key information.

usAttrCount

The number of attributes that the component has.

pAttrs ATTRIBUTE structure array containing the attribute values.

usChildCompCount

Number of child components.

pChildComps

COMPONENTSTRUCT structure array containing child components.

USEREXITSTRUCT:

```
typedef struct _USEREXITSTRUCT
{
    PVOID          pDs;
    CHAR            szPIDString[PID_STRING_LEN];
    CHAR            szProcessName[MAX_DISPLAY_NAME_LEN];
}
```



```

KEYINFO          compOrigViewKeyInfo;
CHAR             szUserId[MAX_NAME_LEN];
USHORT           usAccessLevel;
PCOMPONENTSTRUCT pRootComponent;
}USEREXITSTRUCT, *PUSEREXITSTRUCT;

```

Where:

pDs pDs Session handle returned by DKDatastoreICM::connect

szPIDString

szPIDString is the item's PID String of the current document or folder to be saved.

szProcessName

Process name of currently opened item. This parameter might be NULL for a process unaware item.

compOrigViewKeyInfo

Pointer to the KEYINFO structure describing the attribute information of the item or folder to be saved.

szUserId

Identifies the user ID of the user saving the document or folder.

usAccessLevel

States the access privilege the user has for this document or folder. The value is NULL if the object is not opened by this user. The valid values are:

UX_PRIV_READ when the user opens this object in browse mode.

UX_PRIV_WRITE when the user opens this object in update mode.

pRootComponent

Pointer to the Root Component of the Item or folder. The valid value for this user exit routine is UX_PRIV_WRITE when the user opens this object in UPDATE mode.

Return values

Zero (0) for success; non-zero for error.

Comments

A non-zero return code is logged and disables the menu item. This function is called each time the user clicks on the corresponding menu item under the "Actions" menu.

Appendix A. OLE API mappings from Content Manager version 7 to Content Manager version 8

Many OLE APIs that existed in the previous versions of Content Manager have changed. The following tables list all of the object properties and methods and whether or not they have changed from version 7 to version 8.

Application object

Table 6. Application object properties

Content Manager Version 7 properties	Content Manager Version 8 properties	Reason for change
Application (VT_DISPATCH)	(same)	Not applicable
Documents (VT_DISPATCH)	(same)	Not applicable
Error (VT_DISPATCH)	(same)	Not applicable
HWND (VT_I4)	(same)	Not applicable
Image (VT_DISPATCH)	(same)	Not applicable
KeyFieldTranslation (VT_BOOL)	AttributeTranslation (VT_BOOL)	Name change for new terminology
NewPassword (VT_BSTR)	(same)	Not applicable
Password (VT_BSTR)	(same)	Not applicable
Server (VT_BSTR)	(same)	Not applicable
User (VT_BSTR)	(same)	Not applicable
Visible (VT_BOOL)	(same)	Not applicable

Table 7. Application object methods

Content Manager Version 7 methods	Content Manager Version 8 methods	Reason for change
VT_EMPTY Activate	(same)	Not applicable
VT_VARIANT ClassArray	VT_VARIANT ItemTypeArray	Name change for new terminology
VT_VARIANT ClassKeyFieldArray VT_BSTR IndexClass	VT_VARIANT ItemTypeAttributeArray VT_BSTR ItemTypeName	Name change for new terminology
	VT_VARIANT ChildCompArray VT_BSTR ComponentName	New method for new data model
	VT_VARIANT ChildCompAttributeArray VT_BSTR ComponentName	New method for new data model
VT_BSTR ClassKeyFieldList VT_BSTR IndexClass VT_BSTR Separator	VT_BSTR ItemTypeAttributeList VT_BSTR ItemTypeName VT_BSTR Separator	Names change for new terminology
VT_BSTR ClassList VT_BSTR Separator	VT_BSTR ItemTypeList VT_BSTR Separator	Name change for new terminology

Table 7. Application object methods (continued)

Content Manager Version 7 methods	Content Manager Version 8 methods	Reason for change
	VT_BSTR ChildCompList VT_BSTR ComponentName VT_BSTR Separator	New method for new data model
	VT_BSTR ChildCompAttributeList VT_BSTR ComponentName VT_BSTR Separator	New method for new data model
VT_VARIANT ClassSubsetsArray VT_BSTR IndexClass	VT_VARIANT ItemTypeViewsArray VT_BSTR ItemTypeName	Names change for new terminology
VT_BSTR ClassSubsetsList VT_BSTR IndexClass VT_BSTR Separator	VT_BSTR ItemTypeViewsList VT_BSTR ItemTypeName VT_BSTR Separator	Names change for new terminology
VT_VARIANT ContentClassArray	VT_VARIANT MIMETypeArray	Name change for new terminology
VT_BSTR ContentClassList VT_BSTR Separator	VT_BSTR MIMETypeList VT_BSTR Separator	Name change for new terminology
VT_DISPATCH CreateDocument VT_BSTR Source	VT_DISPATCH CreateItem VT_I2 Type	CreateDocument and CreateFolder merged
VT_DISPATCH CreateFolder VT_BSTR Source	VT_DISPATCH CreateItem VT_I2 Type	CreateDocument and CreateFolder merged
VT_EMPTY DisableMenus VT_I4 Flags VT_VARIANT Hide	VT_EMPTY DisableMenus VT_I4 Flags VT_BOOL Hide	Changed data type
VT_I4 ExtendedPrintSetup VT_BOOL Comments VT_BOOL Borders VT_BOOL SinglePage VT_BOOL HorizPages VT_BOOL PageNumbers VT_I2 NumRows VT_I2 NumColumns	VT_EMPTY ExtendedPrintSetup VT_BOOL Comments VT_BOOL Borders VT_BOOL SinglePage VT_BOOL HorizPages VT_BOOL PageNumbers VT_I2 NumRows VT_I2 NumColumns	Eliminate unusable return value
VT_BSTR GetCurrentClassSubset VT_BSTR IndexClass	VT_BSTR GetActiveView VT_BSTR ItemTypeName	Name change for new terminology
VT_DISPATCH GetWorkbasket VT_BSTR Name	VT_DISPATCH GetWorklistFromName VT_BSTR Name	New document routing model
VT_DISPATCH ItemID VT_BSTR Item	VT_DISPATCH GetItemFromPID VT_BSTR ItemPid	Names change for new terminology
VT_VARIANT KeyFieldArray	VT_VARIANT AttributeArray	Name change for new terminology
VT_BSTR KeyFieldList VT_BSTR Separator	VT_BSTR AttributeList VT_BSTR Separator	Name change for new terminology
VT_I4 Logon	VT_EMPTY Logon	Eliminate unusable return value
VT_EMPTY OpenBasicSearch	(same)	Not applicable
VT_EMPTY OpenScan	(same)	Not applicable
VT_EMPTY OpenWorkbasket	VT_EMPTY OpenWorklist	New document routing model

Table 7. Application object methods (continued)

Content Manager Version 7 methods	Content Manager Version 8 methods	Reason for change
VT_I4 PrintSetup VT_BSTR Printer VT_I2 PaperSize VT_BOOL Portrait VT_I2 Copies VT_VARIANT Scaling	VT_EMPTY PrintSetup VT_BSTR Printer VT_I2 PaperSize VT_BOOL Portrait VT_I2 Copies VT_BOOL Scaling	Eliminate unusable return value and changed data type
VT_BOOL QueryPrivilege VT_I4 Authority VT_VARIANT Context	VT_BOOL QueryPrivilege VT_BSTR PrivilegeName VT_VARIANT Context VT_I2 Type	Support new data model
VT_EMPTY Quit	(same)	Not applicable
VT_DISPATCH Search VT_BSTR IndexClass VT_BSTR SearchString VT_VARIANT TypeFilter VT_VARIANT WipFilter VT_VARIANT SuspendFilter	VT_DISPATCH Search VT_BSTR ItemTypeViewName VT_VARIANT QuerySubstring VT_BOOL AllVersions VT_I2 TypeFilter VT_I2 SuspendFilter VT_VARIANT ProcessName VT_VARIANT StepName	Support new data model
VT_EMPTY SetCurrentClassSubset VT_BSTR IndexClass VT_BSTR Subset	VT_EMPTY SetActiveView VT_BSTR ItemTypeName VT_BSTR ViewName	Names change for new terminology
VT_EMPTY SetPrintRect	(same)	Not applicable
VT_I4 VersionsSupported	VT_I4 VersionsSupported VT_BSTR ItemTypeName	Support new data model
VT_VARIANT WorkbasketArray	VT_VARIANT WorklistArray	New document routing model
VT_BSTR WorkbasketList VT_BSTR Separator	VT_BSTR WorklistList VT_BSTR Separator	New document routing model
	VT_VARIANT ProcessArray	New method for new document routing model
	VT_BSTR ProcessList VT_BSTR Separator	New method for new document routing model
	VT_BOOL IsLoggedOn	New method
	VT_I2 VersioningType VT_BSTR ItemTypeName VT_I2 Context	New method for new data model
	VT_BOOL IsValueRequired VT_BSTR ComponentName VT_BSTR AttributeName	New method for new data model
	VT_I4 GetIDFromName VT_BSTR Name VT_I2 Type	New method for new data model
	VT_BSTR GetDisplayNameFromName VT_BSTR Name VT_I2 Type	New method for new data model

Document object

Table 8. Document object properties

Content Manager Version 7 properties	Content Manager Version 8 properties	Reason for change
Application (VT_DISPATCH)	(same)	Not applicable
Count (VT_I4)	(same)	Not applicable
Item (VT_DISPATCH)	(same)	Not applicable
Page (VT_I4)	(same)	Not applicable
PageCount (VT_I4)	(same)	Not applicable
Parent (VT_DISPATCH)	(same)	Not applicable
SelectedCount (VT_I4)	(same)	Not applicable
Type (VT_I4)	(same)	Not applicable

Table 9. Document object methods

Content Manager Version 7 methods	Content Manager Version 8 methods	Reason for change
VT_I4 Activate	VT_EMPTY Activate	Eliminate unusable return value
VT_I4 CaretIndex	(same)	Not applicable
VT_I4 ClearSelect	VT_EMPTY ClearSelect	Eliminate unusable return value
VT_I4 Close & CloseIt VT_VARIANT Save	VT_EMPTY Close & CloseIt VT_VARIANT Save	Eliminate unusable return value
VT_I4 DisplayPage VT_I4 Page	VT_EMPTY DisplayPage VT_I4 Page	Eliminate unusable return value
VT_I4 FirstPage	VT_EMPTY FirstPage	Eliminate unusable return value
VT_DISPATCH IndexedItem VT_I4 Index	(same)	Not applicable
VT_I4 LastPage	VT_EMPTY LastPage	Eliminate unusable return value
VT_I4 Maximize	VT_EMPTY Minimize	Eliminate unusable return value
VT_I4 Minimize	VT_EMPTY Minimize	Eliminate unusable return value
VT_I4 NextPage	VT_EMPTY NextPage	Eliminate unusable return value
VT_I4 PreviousPage	VT_EMPTY PreviousPage	Eliminate unusable return value
VT_I4 Restore	VT_EMPTY Restore	Eliminate unusable return value
VT_DISPATCH Selections	(same)	Not applicable
VT_I4 SelectRange VT_I4 First VT_I4 Last	VT_EMPTY SelectRange VT_I4 First VT_I4 Last	Eliminate unusable return value
VT_I4 Zoom VT_I4 Percent	VT_EMPTY Zoom VT_I4 Percent	Eliminate unusable return value
VT_I4 ZoomFit VT_I4 Type	VT_EMPTY ZoomFit VT_I4 Type	Eliminate unusable return value
VT_I4 ZoomRect VT_I4 Left VT_I4 Top VT_I4 Right VT_I4 Bottom	VT_EMPTY ZoomRect VT_I4 Left VT_I4 Top VT_I4 Right VT_I4 Bottom	Eliminate unusable return value

Documents object

Table 10. Documents object properties

Content Manager Version 7 properties	Content Manager Version 8 properties	Reason for change
Active (VT_I4)	(same)	Not applicable
Application (VT_DISPATCH)	(same)	Not applicable
Count (VT_I4)	(same)	Not applicable
Parent (VT_DISPATCH)	(same)	Not applicable

Table 11. Documents object methods

Content Manager Version 7 methods	Content Manager Version 8 methods	Reason for change
VT_I4 Cascade	VT_EMPTY Cascade	Eliminate unusable return value
VT_EMPTY Close & CloseIt	VT_EMPTY Close & CloseIt VT_VARIANT Save	Support new data model
VT_DISPATCH Item VT_I4 Index	(same)	Not applicable
VT_DISPATCH OpenDocument VT_DISPATCH Item VT_VARIANT Browse VT_VARIANT Version	VT_DISPATCH OpenDocument VT_DISPATCH Item VT_BOOL Browse VT_I4 Version	Support new data model
VT_DISPATCH OpenTOC VT_DISPATCH Item VT_VARIANT Browse	VT_DISPATCH OpenTOC VT_DISPATCH Item VT_BOOL Browse VT_I4 Version	Support new data model
VT_I4 Tile VT_I4 Vertical	VT_EMPTY Tile VT_I4 Vertical	Eliminate unusable return value

Error object

Table 12. Error object properties

Content Manager Version 7 properties	Content Manager Version 8 properties	Reason for change
ErrorMessage (VT_BSTR)	(same)	Not applicable
ExtReturnCode (VT_I4)	(not supported anymore)	Not applicable
ReturnCode (VT_I4)	(not supported anymore)	Not applicable
	ErrorCode(VT_I4)	New property for new data model
	ErrorID(VT_I4)	New property for new data model
	ErrorState(VT_BSTR)	New property for new data model

Image object

Table 13. Image object properties

Content Manager Version 7 properties	Content Manager Version 8 properties	Reason for change
Application (VT_DISPATCH)	(same)	Not applicable

Table 13. Image object properties (continued)

Content Manager Version 7 properties	Content Manager Version 8 properties	Reason for change
Item (VT_DISPATCH)	(same)	Not applicable
Page (VT_I4)	(same)	Not applicable
Parent (VT_DISPATCH)	(same)	Not applicable

Table 14. Image object methods

Content Manager Version 7 methods	Content Manager Version 8 methods	Reason for change
VT_EMPTY Close & CloseIt VT_VARIANT Save	(same)	Not applicable
VT_I4 DisplayPage VT_I4 Page	VT_EMPTY DisplayPage VT_I4 Page	Eliminate unusable return value
VT_I4 FirstPage	VT_EMPTY FirstPage	Eliminate unusable return value
VT_I4 LastPage	VT_EMPTY LastPage	Eliminate unusable return value
VT_I4 NextPage	VT_EMPTY NextPage	Eliminate unusable return value
VT_I4 OpenDocument VT_DISPATCH Index	VT_EMPTY OpenDocument VT_DISPATCH Item	Eliminate unusable return value and name change
VT_EMPTY PreStage VT_I4 Index	(not supported anymore)	Not applicable
VT_I4 PreviousPage	VT_EMPTY PreviousPage	Eliminate unusable return value

Item object

Table 15. Item object properties

Content Manager Version 7 properties	Content Manager Version 8 properties	Reason for change
Application (VT_DISPATCH)	(same)	Not applicable
CheckedStatus (VT_BSTR)	CheckOutUserID (VT_BSTR)	New data model
Class (VT_BSTR)	ItemType (VT_BSTR)	Name change for new terminology
ItemID (VT_BSTR)	ItemPID (VT_BSTR)	Support new data model
Name (VT_BSTR)	(same)	Not applicable
Parent (VT_DISPATCH)	(same)	Not applicable
PartCount (VT_I4)	(same)	Not applicable
Priority (VT_I4)	(same)	Not applicable
SystemAssigned (VT_BOOL)	ReturnLimit (VT_I4)	New document routing model
TOCCCount (VT_I4)	(same)	Not applicable
Type (VT_I4)	(same)	Not applicable
	Version (VT_I4)	New method for new data model
	Suspended(VT_BOOL)	New method for new data model
	Process(VT_BSTR)	New method for new data model
	Step(VT_BSTR)	New method for new data model
	OwnerID(VT_BSTR)	New method for new data model

Table 16. Item object methods

Content Manager Version 7 methods	Content Manager Version 8 methods	Reason for change
VT_I4 Activate	VT_EMPTY Activate	Eliminate unusable return value
VT_I4 AddAnnotationPart VT_BSTR Path	VT_EMPTY AddAnnotationPart VT_BSTR Path	Eliminate unusable return value
VT_I4 AddPart VT_BSTR Path VT_BSTR ContentClass VT_VARIANT SMSOption	VT_EMPTY AddPart VT_BSTR Path VT_BSTR MIMETYPE VT_I2 PartType VT_I4 SMSOption VT_BOOL TextSearchable	Support new data model
VT_I4 AddToFolder VT_DISPATCH Folder	VT_EMPTY AddToFolder VT_DISPATCH Folder	Eliminate unusable return value
VT_I4 ChangeNotes VT_BSTR Value	VT_EMPTY ChangeNotes VT_BSTR Notes VT_BOOL Version	Eliminate unusable return value and support new data model
VT_I4 ChangeWorkflow VT_BSTR WorkFlow	VT_EMPTY ChangeProcess VT_BSTR ProcessName VT_I4 Priority	Eliminate unusable return value and support new document routing data model
VT_I4 CheckIn	VT_EMPTY CheckIn	Eliminate unusable return value
VT_I4 CheckOut	VT_EMPTY CheckOut	Eliminate unusable return value
VT_I4 Close & CloseIt	VT_EMPTY Close & CloseIt	Eliminate unusable return value
VT_I4 CloseNotes	VT_EMPTY CloseNotes	Eliminate unusable return value
VT_I4 CloseParts	VT_EMPTY CloseParts VT_BOOL Delete	Eliminate unusable return value and support new data model
VT_I4 CompleteWorkflow	(not supported anymore)	Not applicable
VT_I4 Delete	VT_EMPTY Delete	Eliminate unusable return value
VT_I4 DeletePart VT_I4 Index	VT_EMPTY DeletePart VT_I4 Index	Eliminate unusable return value
VT_I4 FaxItem VT_VARIANT withSubFolderContents	(not supported anymore)	Not applicable
VT_BSTR GetAnnotationFile VT_VARIANT Version	VT_BSTR GetAnnotationFile	Support new data model
VT_BSTR GetKeyFields VT_BSTR Name	VT_BSTR GetRootCompAttributeValue VT_BSTR AttributeName	Support new data model
	VT_VARIANT GetChildCompAttributeValueArray VT_BSTR ComponentName VT_BSTR AttributeName	New method for new data model
	VT_BSTR GetChildCompAttributeValueList VT_BSTR ComponentName VT_BSTR AttributeName VT_BSTR Separator	New method for new data model
VT_BSTR GetHistorylog	(same)	Not applicable
VT_BSTR GetNotes	(same)	Not applicable
VT_DISPATCH GetParentFolders	(same)	Not applicable

Table 16. Item object methods (continued)

Content Manager Version 7 methods	Content Manager Version 8 methods	Reason for change
VT_BSTR GetPartContentClass VT_I4 Index	VT_BSTR GetPartMIMEType VT_I4 Index	Name change for new terminology
VT_BSTR GetPartFile VT_I4 Index VT_VARIANT Version	VT_BSTR GetPartFile VT_I4 Index	Support new data model
VT_DISPATCH GetTOCItem VT_I4 Index	(same)	Not applicable
	VT_DISPATCH GetVersionedItem VT_I4 Version	New method for new data model
VT_DISPATCH NextWorkbasketItem	VT_DISPATCH NextWorkistItem	New document routing model
VT_EMPTY Open	VT_EMPTY Lock	Name change for clarity
VT_I4 PrintItem VT_BOOL ShowDialog VT_VARIANT PrintImage VT_VARIANT StartPage VT_VARIANT EndPage VT_VARIANT PrintMarkup VT_VARIANT PrintIndex VT_VARIANT PrintNotelog VT_VARIANT PrintTOC VT_VARIANT PrintContents	VT_EMPTY PrintItem VT_BOOL ShowDialog VT_BOOL PrintImage VT_I4 StartPage VT_I4 EndPage VT_BOOL PrintMarkup VT_BOOL PrintIndex VT_BOOL PrintNotelog VT_BOOL PrintTOC VT_BOOL PrintContents	Eliminate unusable return value and changed data type
VT_I4 RefreshTOC	VT_EMPTY RefreshTOC	Eliminate unusable return value
VT_I4 RemoveFromFolder VT_DISPATCH Folder	VT_EMPTY RemoveFromFolder VT_DISPATCH Folder	Eliminate unusable return value
VT_I4 RemoveFromWorkbasket VT_DISPATCH Workbasket	(not supported anymore)	New document routing model
VT_I4 RemoveFromWorkflow	VT_EMPTY RemoveFromProcess	Name change for new document routing terminology
VT_I4 RouteToWorkbasket VT_DISPATCH Workbasket VT_VARIANT Priority VT_VARIANT Force	(not supported anymore)	New document routing model
VT_I4 SavePart VT_I4 Index VT_VARIANT SaveMode	VT_EMPTY SavePart VT_I4 Index VT_BOOL Version	Eliminate unusable return value and support new data model
VT_EMPTY SetKeyFields VT_BSTR Name VT_BSTR NewValue	VT_EMPTY SetRootCompAttributeValue VT_BSTR AttributeName VT_BSTR AttributeValue	Name change for new terminology
	VT_EMPTY SetChildCompAttributeValue VT_BSTR ComponentName VT_BSTR AttributeName VT_I2 MultiValueIndex VT_BSTR AttributeValue	New method for new data model
	VT_EMPTY SetChildCompMultiValueCount VT_BSTR ComponentName VT_I2 MultiValueCount	New method for new data model

Table 16. Item object methods (continued)

Content Manager Version 7 methods	Content Manager Version 8 methods	Reason for change
VT_I4 StartWorkflow VT_BSTR Workflow VT_VARIANT Workbasket VT_VARIANT Priority	VT_EMPTY StartOnProcess VT_BSTR ProcessName VT_I4 Priority	Name change for new document routing terminology
VT_I4 Suspend VT_VARIANT Timestamp VT_VARIANT TimeoutWorkbasket VT_VARIANT Classes VT_VARIANT Criteria VT_VARIANT ReadyWorkbasket	VT_EMPTY Suspend VT_I4 Interval	New document routing model
VT_I4 UpdateIndex	VT_EMPTY UpdateItem VT_BOOL NewVersion	Support new data model
VT_VARIANT VersionsArray VT_VARIANT PartType	VT_VARIANT VersionsArray	Support new data model
VT_BSTR VersionsList VT_BSTR Separator VT_VARIANT PartType	VT_BSTR VersionsList VT_BSTR Separator	Support new data model
	VT_EMPTY SaveAnnotation VT_BOOL Version	New method for new data model
	VT_EMPTY AddToResumeList VT_BSTR ItemTypeName VT_I4 NumItems	New method for new document routing model

Items object

Table 17. Items object properties

Content Manager Version 7 properties	Content Manager Version 8 properties	Reason for change
Application (VT_DISPATCH)	(same)	Not applicable
Count (VT_I4)	(same)	Not applicable
Parent (VT_DISPATCH)	(same)	Not applicable

Table 18. Items object methods

Content Manager Version 7 methods	Content Manager Version 8 methods	Reason for change
VT_I4 Close & CloseIt	VT_EMPTY Close & CloseIt	Eliminate unusable return value
VT_DISPATCH Item VT_I4 Index	(same)	Not applicable

Appendix B. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give 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:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J74/G4
555 Bailey Avenue
San Jose, CA 95141
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, 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. 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 IBM's application programming interfaces.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

IBM	DisplayWrite	PowerPC
400	e-business	PTX
Advanced Peer-to-Peer Networking	HotMedia	QBIC
AIX	Hummingbird	RS/6000
AIXwindows	ImagePlus	SecureWay
APPN	IMS	SP
AS/400	Micro Channel	VideoCharger
C Set ++	MQSeries	Visual Warehouse
CICS	MVS/ESA	VisualAge
DATABASE 2	NetView	VisualInfo
DataJoiner	OS/2	WebSphere
DB2	OS/390	
DB2 Universal Database	PAL	

Approach, Domino, Lotus, Lotus 1-2-3, Lotus Notes and SmartSuite are trademarks or registered trademarks of the Lotus Development Corporation in the United States, other countries, or both.

Intel and Pentium are trademarks or registered trademarks of Intel Corporation in the United States, other countries, or both.

Microsoft, Windows, and Windows NT are registered 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 Sun Microsystems, Inc. in the United States, other countries, or both.

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.

Index

A

application object 7
application object method 9
application object properties 7

C

client application objects 2
Close 35

D

document object 20
documents object 24

E

error object 2, 27

H

handling errors 4

I

image object 2, 29
item lists 1
item object 2, 30
items collection 41

M

method
 application object 9
 Activate 9
 AttributeArray 9
 AttributeList 9
 ChildCompArray 9
 ChildCompAttributeArray 9
 ChildCompAttributeList 9
 ChildCompList 9
 CreateItem 10
 DisableMenus 10
 ExtendedPrintSetup 11
 GetActiveView 12
 GetDisplayNameFromName 12
 GetIDFromName 13
 GetItemFromPID 13
 GetWorklistFromName 13
 IsLoggedIn 13
 IsValueRequired 13
 ItemTypeArray 13
 ItemTypeAttributeArray 13
 ItemTypeAttributeList 14
 ItemTypeList 14
 ItemTypeViewsArray 14
 ItemTypeViewsList 14
 Logon 14

method (*continued*)

 application object (*continued*)

 MIMETypeArray 15
 MIMETypeList 15
 OpenBasicSearch 15
 OpenScan 15
 OpenWorklist 15
 PrintSetup 15
 ProcessArray 16
 ProcessList 17
 QueryPrivilege 17
 Quit 17
 Search 17
 SetActiveView 19
 SetPrintRect 19
 VersioningType 20
 VersionsSupported 20
 WorklistArray 20
 WorklistList 20

 document object

 Activate 21
 CaretIndex 22
 ClearSelect 22
 Close 22
 CloseIt 22
 DisplayPage 22
 FirstPage 23
 IndexedItem 23
 LastPage 23
 Maximize 23
 Minimize 23
 NextPage 23
 PreviousPage 23
 Restore 23
 Selections 24
 SelectRange 24
 Zoom 24
 ZoomFit 24
 ZoomRect 24

 documents collection object

 Cascade 25
 Close 25
 CloseIt 25
 Item 26
 OpenDocument 26
 OpenTOC 26
 Tile 26

 image object

 Close 29
 CloseIt 29
 DisplayPage 30
 FirstPage 30
 LastPage 30
 NextPage 30
 OpenDocument 30
 PreviousPage 30

 item object

 Activate 33
 AddAnnotationPart 33
 AddPart 33
 AddToFolder 34

method (*continued*)

 item object (*continued*)

 AddToResumeList 34
 ChangeNotes 34
 ChangeProcess 34
 CheckIn 34
 CheckOut 34
 CloseIt 35
 CloseNotes 35
 CloseParts 35
 Delete 35
 DeletePart 35
 GetAnnotationFile 35
 GetChildCompAttributeValueArray 35
 GetChildCompAttributeValueList 36
 GetHistorylog 36
 GetItemPid 36
 GetNotes 36
 GetParentFolders 36
 GetPartFile 36
 GetPartMIMEType 37
 GetRootCompAttributeValue 37
 GetTOCItem 37
 GetVersionedItem 37
 IsValueRequired 37
 Lock 37
 NextWorklistItem 37
 PrintItem 38
 RefreshTOC 38
 RemoveFromFolder 39
 RemoveFromProcess 39
 SaveAnnotation 39
 SavePart 39
 SetChildCompAttributeValue 39
 SetChildCompMultiValueCount 40
 SetRootCompAttributeValue 40
 StartOnProcess 40
 Suspend 40
 UpdateItem 40
 VersionsArray 41
 VersionsList 41

 items collection object

 Close 41
 CloseIt 41
 Item 41

methods 2

O

object

 application 2, 7, 9
 document 20
 document object 2
 documents 24
 documents collection 3
 error 27
 image 29
 image object 3
 item 30
 item object 3
 items collection 3

P

- programming tips 3
- Programming with OLE automation 1
- properties 1
 - application object 7
 - application 7
 - AttributeTranslation 7
 - Documents 7
 - Error 8
 - hWnd 8
 - Image 8
 - NewPassword 8
 - Password 8
 - Server 8
 - User 8
 - Visible 8
 - document object
 - Application 20
 - Count 21
 - Item 21
 - Page 21
 - PageCount 21
 - Parent 21
 - SelectedCount 21
 - Type 21
 - documents collection object
 - Active 25
 - Application 25
 - Count 25
 - Parent 25
 - error object
 - ErrorCode 27
 - ErrorID 27
 - ErrorMessage 28
 - ErrorState 28
 - image object
 - Application 29
 - Item 29
 - Page 29
 - Parent 29
 - item object
 - Application 31
 - CheckOutUserID 31
 - ItemPID 31
 - ItemType 31
 - Name 31
 - OwnerID 31
 - Parent 31
 - PartCount 31
 - Priority 32
 - Process 32
 - ReturnLimit 32
 - Step 32
 - TOCCount 32
 - Type 32
 - Version 33
 - items collection object
 - Application 41
 - Count 41
 - Parent 41
- property and argument types 4

R

- read-only 1
- read/write 1

- releasing objects 4

S

- sample Visual Basic program 5

T

- table of contents (TOC) 1

U

- user exit
 - add items to actions menu 58
 - add items to actions parameters 58
 - alternate search 43
 - alternate search parameters 43
 - alternate search purpose 43
 - AlternateSearchUserExit 43
 - change SMS parameters 52
 - change SMS user type 52
 - ChangeSMSUserExitType 52
 - enable or disable menu items 59
 - enable or disable menu
 - parameters 59
 - enable or disable menu purpose 59
 - get attribute value list 55
 - get value list length 57
 - get value list length parameters 57
 - get value list length purpose 57
 - GetAttributeValueList 55
 - GetValueListLength 57
 - query sort 44
 - query sort parameters 45
 - query sort purpose 44
 - QuerySortUserExit 44
 - save record 48
 - save record parameters 49
 - save record purpose 49
 - SaveRecordUserExit 48
 - UpdateFunc 59
 - user option function 61
 - user option function parameters 62
 - user option function purpose 61
 - UserOptFunc 61
 - UserOptionList 58

W

- write-only 1



Program Number: 5724-B19

Printed in U.S.A.

SC27-1337-01

