

DM Extensions API Reference Guide

Hummingbird Enterprise™ 2004

DM Extensions API Reference Guide

Version: 5.1.0.5

Copyright © 1998-2004 Hummingbird Ltd. All rights reserved.

Electronic Publication Date: April 2004

Hummingbird Enterprise™ 2004, DM Extension, and DM are trademarks of Hummingbird Ltd. and/or its subsidiaries. All other copyrights, trademarks, and trade names are the property of their respective owners.

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

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 Unpublished rights reserved under the copyright laws of the United States and any other appropriate countries.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

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

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

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

Contents

Chapter 1 DM API Referernce Guide

Introduction **2**

Important Notice Regarding Use of the DM Extensions API Toolkit **2**

DM Extensions API Documentation Organization **2**

CMSForm Control and DM Designer Forms **3**

DM Extensions Architecture Overview **4**

Chapter 2 Explorer Core

About Explorer Core **8**

Operation Context **9**

Object Identification **10**

Explorer Core Object Model **11**

Using Explorer Core API **12**

Creating Custom Browser **12**

Columns And Properties **13**

Drag And Drop **14**

Menu And Toolbars **15**

Notifications **17**

Creating Custom Plugins **18**

Explorer Core Objects **19**

BCBrowserFactory **19**

BCCoreCtx **20**

BCObjectArray **20**

BCRegistrar **21**

BCSession **22**

Explorer Core Interfaces **24**

IBCBrowser **25**

IBCBrowser interface IDL definition **29**
IBCBrowser2 **30**
IBCBrowser3 **33**
IBCBrowserCore **34**
IBCBrowserFactory **37**
IBCContextMenuSupport **38**
IBCCoreCtx **46**
IBCDragDropSupport **52**
IBCEnumObjects **57**
IBCEnumProperties **68**
IBCEnumVerbs **74**
IBCObject **80**
IBCObject2 **90**
IBCObjectArray **91**
IBCObjectCaps **102**
IBCObjectCaps2 **107**
IBCObjectChanged **110**
IBCPlugin **111**
IBCProperty **119**
IBCPropertyArray **127**
IBCRegistrar **141**
IBCRegistrarExt **145**
IBCRetrievalSink **151**
IBCSession **153**
IBCVerb **160**
Explorer Core Enumerations **174**
BrowseObjectFlags Enumeration **174**
BrowserInfoFlags Enumeration **174**
CoreEvents Enumeration **175**
CoreFlags Enumeration **177**
GetBrowserFlags Enumeration **177**

KeyFlags Enumeration **177**
LastRowReason Enumeration **178**
PropertyArrayFlags Enumeration **179**
PropertyArraySaveFlags Enumeration **179**
PropertyAttributes Enumeration **180**
RegisterFlags Enumeration **181**
RegistrarCommand Enumeration **181**
VerbFlags Enumeration **182**
ViewMode Enumeration **183**

Chapter 3 DM Object Model

Document Management Objects **186**
Document Management Objects and Services **186**
Document Management Objects Structure **187**
OM Creatable Objects **192**
ActivityInfo **193**
Application **193**
DocProfiles **195**
Profile **195**
Project **197**
Projects **198**
QuickRetrieve **199**
Search **200**
Versions **201**
Object Model Interfaces **202**
IActivityInfo **202**
IAppForm **213**
IAppForms **217**
IAppInfo **220**
IAppInfo2 **240**
IApplication **241**
IApplication2 **256**

IApplication3 **261**
IApplication4 **262**
IApplication5 **264**
IApplications **266**
IColumn **269**
IColumn2 **281**
IColumns **285**
IDocProfiles **292**
IDocProfiles2 **302**
IDOCSForm **306**
IDOCSForms **312**
IDOComponent **315**
IDOComponents **321**
IDocTransferNotify **327**
ILaunchMethod **331**
ILaunchMethods **354**
ILibraries **357**
ILibrary **360**
ILibrary2 **400**
ILibrary3 **403**
IMonitorFile **406**
IMonitorFiles **409**
IOMErrInfo **412**
IOMObjects **414**
IProfile **419**
IProfile2 **470**
IProfileTrustee **474**
IProfileTrustee2 **486**
IProject **497**
IProject2 **521**
IProject3 **526**

IProjects **534**
IProjectTrustee **540**
IProperties **546**
IProperty **551**
IQuickRetrieve **556**
IQuickRetrieveCallback **566**
ISearch **567**
ISearches **591**
ISearchTrustee **595**
ISearchTrustee2 **601**
ITerminology **604**
ITrusteeCollection **609**
IVersion **625**
IVersion2 **663**
IVersion3 **669**
IVersions **673**
Object Model Enumerations **679**
AccessMode Enumeration **679**
AccessType Enumeration **680**
CanChangeVersionStatus Enumeration **680**
CheckInActions Enumeration **681**
ColumnFlags Enumeration **682**
ColumnTypes Enumeration **682**
CreateVersionFromFileFlags Enumeration **683**
DataTypes Enumeration **683**
DisplayMessageTypes Enumeration **684**
FileTypes Enumeration **685**
FormsResults Enumeration **686**
FTypes Enumeration **686**
LaunchMethodChooserFlags Enumeration **687**
LM_INTEGRATIONSCHEME Enumeration **688**

LockProfileFlags Enumeration **688**
MethodsFlags Enumeration **689**
Modes Enumeration **689**
OMActivityConsts Enumeration **690**
OMErrorCodes Enumeration **692**
OMFlagsLookUp Enumeration **693**
OMFlagsQR Enumeration **694**
ProfileDeleteFlags Enumeration **694**
ProfileRights Enumeration **695**
ProfileStatus Enumeration **696**
ProjectChooserFlags Enumeration **697**
ProjectRights Enumeration **697**
QRCReturnFlags Enumeration **698**
QuickSearchesID Enumeration **698**
QuickSearchRights Enumeration **699**
SAOFFlags Enumeration **700**
SelectForEditFlags Enumeration **700**
SelectForEditReturnFlags Enumeration **701**
SelectVersionsFormFlags Enumeration **701**
ShowProfileFlags Enumeration **702**
SPFFlags Enumeration **702**
SSFFlags Enumeration **703**
TransferNotifyCompletedReason Enumeration **703**
UploadFileExFlags Enumeration **704**
VersionChooserFormFlags Enumeration **704**
VersionPropertiesFlags Enumeration **705**
VersionStatus Enumeration **706**

Chapter 4 ActiveX Controls

ActiveX Controls Overview **710**
Hierarchy Control **710**
Hierarchy Control Reference: Hierarchy **713**

Hierarchy List Control **733**
Hierarchy List Control Reference:HierList **735**
FavoritesBar Control **772**
Favorites Bar Control Reference **772**
FavoritesBar **773**
Folder **797**
FolderItem **805**
Folders **811**
FolderItems **816**

Chapter 5 Event Handlers
About Events API **824**
Using Events API **824**
Notifications **824**
How to create a notifications handler **825**
How to install a notifications handler **825**
Using Events API Example #1 **825**
Using Events API Example #2 **827**
Events API Interfaces **829**
IControlMenu **829**
IOMDocumentEventSink **838**
IOMDocumentMenuSink **848**
IOMEventHandler **852**
IOMEnumSink **854**
IOMMenuSink **856**
Events API Enumerations **862**
DEvents Enumeration **862**
DEventTypes Enumeration **865**
HResultCodes Enumeration **865**
HResultCodesEx Enumeration **866**

Chapter 6 Custom Forms
About Custom Forms **870**

Implementing Custom Forms **871**
Registering Custom Forms **872**
Custom Forms API Reference **873**
IChangeDisplayNameForm **874**
ICompatibleAppsForm **875**
IExportForm **877**
IGroupCheckinForm **880**
IGroupCheckoutForm **881**
IHistoryForm **883**
ILaunchMethodChooserForm **885**
ILibrariesForm **886**
ILibraryChooserForm **887**
ILocationForm **889**
ILookupForm **890**
IMailForm **893**
IProfileAccessForm **894**
IProfileAccessForm2 **896**
IProfileForm **904**
IProfileFormChooser **906**
IProjectAccessForm **908**
IProjectAccessForm2 **909**
IProjectChooserForm **917**
IProjectChooserForm2 **919**
IProjectForm **921**
IProjectMembershipForm **922**
IProjectMembershipForm2 **924**
IPropertiesForm **926**
IQuickRetrieveForm **928**
IRelatedItemsForm **929**
ISaveAsOptionsForm **931**
ISearchAccessForm **933**

ISearchAccessForm2 **934**
ISearchCriteriaForm **942**
ISearchFormChooser **944**
ISearchOptionsForm **945**
ISelectVersionsForm **947**
ISelectVersionTypeForm **949**
ITemplateChooserForm **951**
IVersionChooserForm **952**
Custom Forms Enumerations **954**
ExportInFlags Enumeration **954**
ExportOutFlags Enumeration **955**

Chapter 7 Application Integration API
About Application Integration API **958**
Creation of new Integration Scheme **958**
Registration of new Integration Scheme **958**
Active (ODMA) Integration **959**
Application Integration Interfaces **960**
IAppIntegration **960**
ICloseNotify **976**
IIntegrationScheme **978**

Chapter 8 Interceptor
About Interceptor **990**
Architecture **991**
Interceptor Hook Module **991**
Interceptor Core Module **992**
Document-oriented Events **993**
Processing Interceptor Core Events **994**
Wait/Release Functionality **996**

Chapter 9 Active Integration Implementation
ODMA Document ID **1000**

ODMA DMS Interfaces **1000**
ODMGetODMInterface **1001**
IODMDocMan **1001**
IODMDocMan2 **1013**
IODMQuery **1023**
Document Attributes **1025**

Chapter 10 CMS Controls
Using CMS Forms **1028**
How to use the CMSForm control **1028**
How to create a custom form **1028**
How to replace standard form with your custom form **1031**
How to register our sample form **1031**
CMS Enumerations **1032**
ControlTypeEnum Enumeration **1032**
CMS Objects **1033**
CMSBox **1034**
CMSCheckBox **1049**
CMSEdit **1068**
CMSForm **1116**
CMSList **1136**
CMSPush **1153**
CMSRadioGroup **1169**
CMSText **1184**

Chapter 11 Add-ons
About Add-ons **1200**
Using Add-Ons **1200**
Add-on Container Control **1201**
Creation of new Add-on **1202**
Registration of new Add-on **1204**
Add-On Objects **1206**
AddOnCnt Control **1207**

Profile View Add-on **1218**
Viewer OCX Control **1221**
Versions Add-on **1240**
Logger Add-On **1243**
LoggerOCX Control **1243**
LoggerOCXNotificationSink Control **1244**

Chapter 12 E-Mail Integration
About E-Mail Integration **1250**
Using E-Mail Integration **1250**
Using E-Mail Integration Step 1 **1250**
Using E-Mail Integration Step 2 **1252**
Using E-Mail Integration Step 3 **1255**
E-Mail Integration Interfaces **1257**
IEICore **1258**
IEIMessage **1261**
IEIAttachment **1272**
IEIMessages **1277**
IEIAttachments **1282**
Email Integration Enumerations **1288**
EISaveFlags Enumeration **1288**
EISaveFileFormat Enumeration **1289**
AttachmentType Enumeration **1290**

Chapter 13 Cache Manager
About Cache Manager **1292**
Cache Manager Interfaces **1292**
IDownloadDataSink **1292**
IDownloadResourceProvider **1293**
IDownloadSession **1299**
IDownloadSink **1302**
ILocalFileCacheManager **1306**
IUploadResourceProvider **1313**

Chapter 14 Attache Plug-in

About Attache Plug-in **1316**

Attache Interfaces **1317**

IAttacheRoot **1317**

IUndockedProfile **1323**

IUndockedProfiles **1335**

IUndockedVersion **1339**

IUndockedVersions **1344**

Application Integration Settings **1350**

Common Settings **1350**

Active (ODMA) Integration related registry settings **1353**

Interceptor related registry settings **1355**

Passive Integration related registry settings **1361**

Linking Registry Settings **1363**

Attachè related registry settings **1364**

General registry settings **1366**

Custom Forms Registry Settings **1372**

DM plug-in Settings **1373**

DM Server General Registry Settings **1373**

DM Server Login Registry Settings **1385**

DM Server Forms Registry Settings **1386**

DM Server Shadow Documents Registry Settings **1389**

E-Mail Integration related registry settings **1390**

MS Outlook DM Extension Registry Settings **1394**

Installation related registry settings **1395**

User Interface related registry settings **1396**

Index

Chapter

1

DM API Referernce Guide

In This Chapter

The DM Extensions API toolkit provides hundreds of properties and methods that allow you to customize how your DM Extensions client software operates. The following topics are presented:

- "Introduction" on page 2
- "DM Extensions Architecture Overview" on page 4

Introduction

Prior to the formal discussion of the DM Extensions API, several items should be mentioned:

"Important Notice Regarding Use of the DM Extensions API Toolkit" on page 2

"DM Extensions API Documentation Organization" on page 2

"CMSForm Control and DM Designer Forms" on page 3

Important Notice Regarding Use of the DM Extensions API Toolkit

The DM Extensions API toolkit is intended for professional software developers only. Changes you make to your DM Extensions software can cause the product to malfunction. Any new or modified functionality you incorporate into your DM Extensions software should be tested thoroughly before it is deployed to your user community.

DM Extensions API Documentation Organization

This chapter describes the interfaces of DM Extensions components and how to use them. There are the following DM Extensions Components:

- Explorer Core
- DM Object Model
- ActiveX Controls
- Events Handlers
- Custom Forms
- Application Integration
- Interceptor
- Active Integration Implementation
- CMS Controls
- Add-ons
- E-Mail Integration
- Cache Manager
- Attache Plug-in

CMSForm Control and DM Designer Forms

Developers may want to support DM Designer forms in their custom forms. This is especially true for the profile, search, and search results forms, which can be modified in the DM Designer. For that reason developers should use the CMSForm control, which will do most of the work. Using the CMSForm control you can automatically provide backward compatibility with the DM Designer, since it creates forms on the fly. For example, any changes made to the profile form with the DM Designer utility will be immediately available in your profile form.

The CMSForm control can be used to support forms designed in the DM Designer utility. It takes the description of a form object (which can be obtained using a Library object) as a parameter and recreates it at run-time. The developer can provide event procedures that will be executed in response to the CMSForm specific events.

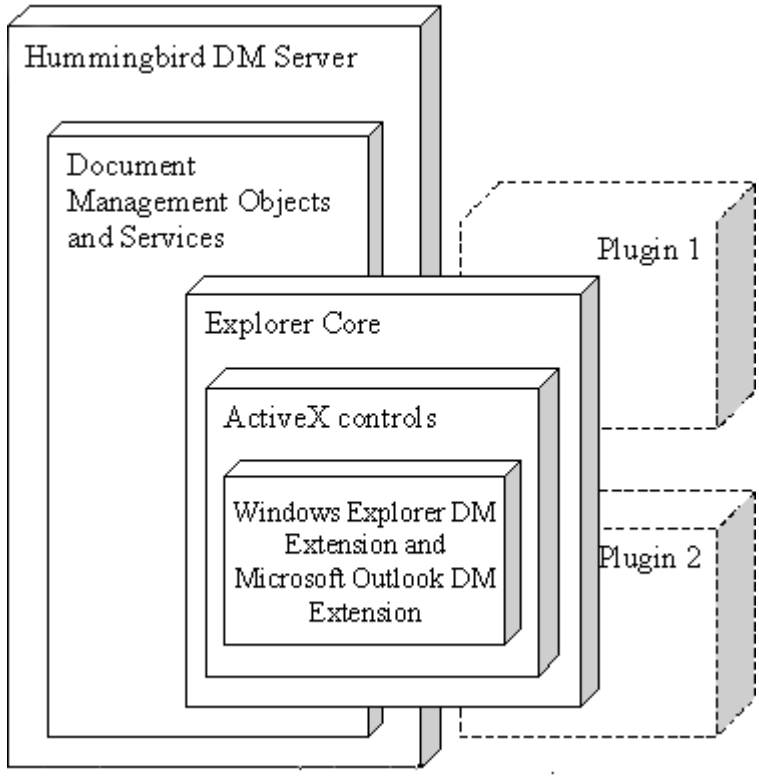
DM Extensions Architecture Overview

DM Extensions is a tightly integrated Windows client for the DM server. Together with the DM Server and the DM database it constitutes a three-tier document management system that allows the user to locate documents in seconds, rather than looking for documents through a tedious directory search.

When designing DM Extensions one of the main goals was to develop a product that could be easily customizable and scalable, so DM Extensions is based on Component Object Model (COM) objects. This allows third-party developers to have access to every product feature and customize or extend almost any part of the product. They can use the same object libraries and services used by DM Extensions developers.

DM Extensions is not a single monolithic product. It consists of many modules, where each one provides some logical piece of the product's functionality. All these modules interact via COM interfaces. This allows the developers to replace any entire module with their own module if a need arises.

The following picture gives the high level overview of the product architecture:



The DM Server forms the basis of the DM document management capabilities.

The Document Management Objects and Services layer makes it easy to use the DM Server functionality by providing high-level document management-oriented COM objects, that represent such fundamental entities as Profiles or Projects. These objects can be used from any modern programming environment including Visual Basic, Delphi, or Power Builder. Thus the developer is allowed to add powerful document management capabilities to almost any application or create own custom solutions integrating workflow, imaging, and document management.

To make it even easier, DM Extensions contains several ActiveX controls that can be used to display and visually manipulate data in the DM libraries. The developer can combine them how he/she likes.

These controls are based on the Explorer Core technology that makes it possible to create the user interface regardless of the business logic of underlying objects. It separates content providers ("plug-ins" in the Explorer Core terminology, modules that extend its functionality) and their UI presentations.

The Explorer Core technology frees the developer from supporting numerous user interface features, such as drag-and-drop and context menu. These features (and many others) are supported automatically by the Explorer Core.

The Explorer Core guarantees (and it's even more important) that if following some simple rules while writing an application, the developer will benefit from the extensibility features made possible by the Explorer Core. For example, all new functionality provided by the plug-ins will be immediately available in developer's application. Plug-in objects will be shown in the Explorer Core ActiveX controls without changing a single line in its code and even without recompilation.

Chapter

2

Explorer Core

In This Chapter

This chapter describes the Explorer Core.

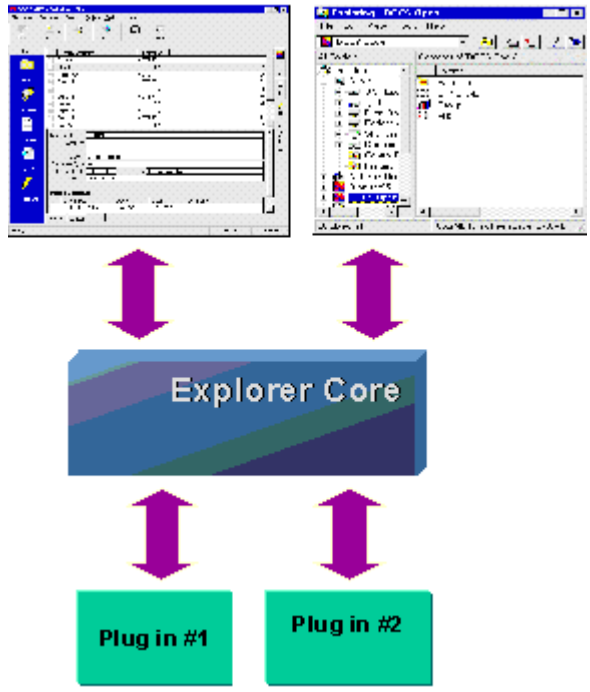
- "About Explorer Core" on page 8
- "Explorer Core Objects" on page 19
- "Explorer Core Interfaces" on page 24
- "Explorer Core Enumerations" on page 174

About Explorer Core

The Explorer Core is a general integration technology that allows the developer to extend the product functionality and greatly reduce development time. The Explorer Core makes it possible to create the user interface regardless of the business logic of underlying objects. It clearly separates the user interface and logic layers in applications. In other words, it separates presentation from content.

These two layers, content providers (plug-ins) and their UI presentations, operate via COM interfaces and so can be created independently of each other. Because these interfaces are not changed, the developer can create new plug-ins and plug them into an Explorer Core-based system. On the one hand it's not necessary to change the user interface, since developer's objects use the same interfaces as any other plug-ins. The developer can reuse entire UI applications. On the other hand, any new UI application will see all the registered plug-ins, since it use the same interfaces as any other Explorer Core-based user interface.

The following picture illustrates how the Explorer Core glues the Microsoft Outlook DM Extension and Windows Explorer DM Extension UI and plug-ins:



The Explorer Core conceals differences between plug-ins for the UI applications. It also conceals differences between UIs for plug-ins. Thus, every plug-in will work with every UI and every UI will display objects from all plug-ins.

Operation Context

Objects can have different visual presentation and support different sets of operations for different contexts. For example, when the user navigates through a tree that shows documents, each document is presented by its name and an icon of its native application. But the same document in another context may be depicted by an icon, which displays the document's status (edited, checked-out, etc.) instead of associated application's icon.

A context should be passed to an object during each request. The object can then decide how it should be presented depending on the supplied context.

The following contexts are now used:

- hierarchy
- contents

Developers are free to define additional contexts. If an object doesn't understand a context passed as an argument, it should use some reasonable defaults for its properties and behavior.

The client can find out whether an object supports a particular context using the UIContextSupported method on its IBCObjectCaps .

See Also

"IBCObjectCaps::UIContextSupported" on page 106

Object Identification

All core and plug-in objects are identified by two parameters: their class and internal identifiers as follows: *Class*!*InternalIdentifier*, where *Class* is the name (class) of the plug-in that supplies objects of given type and *InternalIdentifier* is a string that uniquely identifies an object inside a plug-in. The symbols "/" and "!" are prohibited and serve as separators.

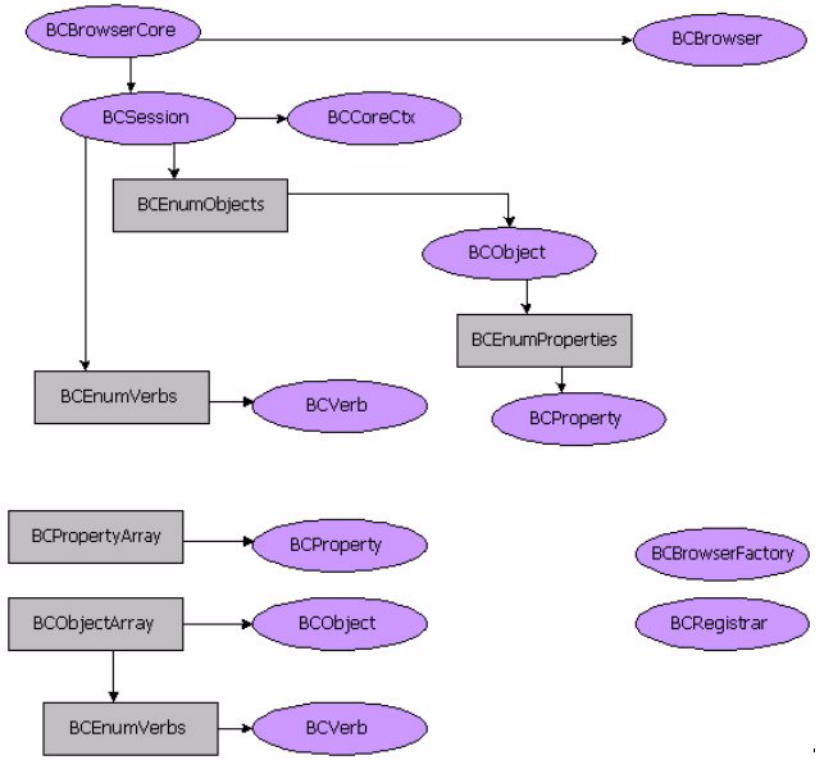
Identifiers can be saved by the client and used in future for retrieving objects.

Examples

```
DOCSOPEN!Pj\SINGLE\123 – folder 123 from library SINGLE.
   DOCSOPEN!Prof\SINGLE\1234 – document 1234 from library SINGLE.
```

Explorer Core Object Model

The following table shows how objects and collections are represented in the diagrams.



The following table shows how objects and collections are represented in the diagrams.

Object



Collection



Using Explorer Core API

Creating Custom Browser

Programmer should implement a COM object BCBrowser that implement an interface IBCBrowser for every view available in browser. For DM version 3.9 and later CBCBrowser should contain IBCBrowser2 and IBCBrowser3 interfaces. Program should logon BCBrowser into BCSession object using method Logon. In VB programs you should use Vhelper library and object VBBrowser.

For example:

```
Dim Session As New DECoreLib.BCSession
Dim Browser As New BCBrowser
Sub Logon
Dim HelperLogon As New VHELPERLib.VBBrowser
HelperLogon.ConnectToSession CoreSession, Browser
Set HelperLogon = Nothing
End Sub
```

Programmer should implement a COM object BCObjectChanged that implements an interface IBCObjectChanged for every type of objects available in browser. BCObjectChanged will be called by objects IBCObject which subscribe to notifications. The following notification can be received:

- bcevNone - Indicates that nothing has been changed
- bcevAdd - Indicates that a new object has been added
- bcevDelete - Indicates that an object has been deleted
- bcevReorder - Indicates that the object's contents have been reordered
- bcevChange - Indicates that an object's property has changed
- bcevRefresh - Indicates that the object's contents have changed

Program should advise BCObjectChanged for notifications of IBCObject using IConnectionPoint interface. In VB programs you should use Vhelper library and object VBObjectChanged.

For example:

```
Sub AddObjectChanged(Object As DECoreLib.BCObject, Collection As
DECoreLib.BCObjectArray)
```

```
Dim ObjectChanged As New BCObjectChanged
Dim VBObjectChanged As New VHELPERLib.VBObjectChanged
VBObjectChanged.Connect Object, ObjectChanged
Collection.Add VBObjectChanged
Set VBObjectChanged = Nothing
Set ObjectChanged = Nothing
End Sub_
```

See Also

"IBCBrowser" on page 25

"IBCBrowser2" on page 30

"IBCBrowser3" on page 33

"IBCObjectChanged" on page 110

"IBCSession::Logon" on page 158

"IBCObject" on page 80

Columns And Properties

1. Object selection (cursor in the right pane was set to a container object)
   - — Query interface IBCObjectCaps from IBCObject of selected container
   - — Use method IBCObjectCaps::GetDetailColumns to retrieve an array of properties to show IBCPropertyArray
   - — Fill the right pane following scenario #4
2. Object deselection (cursor in the right pane was moved from a container object)
   - — Use method IBCObjectCaps::GetDetailColumns to retrieve an array of properties to show IBCPropertyArray
   - — Save property array with method IBCPropertyArray::Save if it changed
3. Customization of columns

— Get list of available properties (for Field Chooser) using IBCEnumProperties

— Use property IBCProperty::DisplayName as a column name in Field Chooser

— For each property customized by user check IBCProperty::Flags:

— bcpaRequired – column is mandatory and cannot be removed if TRUE

— bcpaLinkedWithNext – column should be added or removed together with the next one if TRUE

— Use method IBCObjectCaps::GetDetailColumns to retrieve an array of properties to show IBCPropertyArray

— Update property array with methods IBCPropertyArray::SetColumnProperty, IBCPropertyArray::InsertAt and IBCPropertyArray::RemoveAt.

4 Filling the contents of results pane

— Use method IBCObjectCaps::GetDetailColumns to retrieve an array of properties to show IBCPropertyArray

— Get container children with method IBCObject::EnumChildObjects

— If IBCPropertyArray==NULL show list of objects

— Else for each property in IBCPropertyArray:

— If column width was not changed by user get default attribute “Width” with IBCProperty->GetDefault() and set column width according to it

— Get property value using method IBCObject::GetProperty and show it

— Sort objects if any sorting was enabled by user

Drag And Drop

1 Drag object

— For each selected object (if multiple were selected):

— Ask does it support DD functionality using IBCDragDropSupport-CanDrag()

— Create BCObjectArray with CoCreateInstance()

— Add object(s) that support DD to this array

2 Right drag

— Query interface IBCDragDropSupport from IBCObject of target object

— Use target DragDropSupport->GetMenuItems() to enumerate verbs IBCVerb supported by target object

— Add all verbs to context menu and show it (if the verb has menu ID use it)

— Call IBCDragDropSupport->Drop() passing IDataObject and ID returned by context menu in keyState parameter

3 DropEffect

— Add IDs of dragged objects to it

— Call IBCDragDropSupport->DropEffect() passing IDataObject and keyboard state to it

4 Drop

— Add IDs of dragged objects to it

— Call IBCDragDropSupport->Drop() passing keyboard state and IDataObject to it

Menu And Toolbars

1 Initialization

— Enumerate all available verbs IBCVerb from IBCSession

— For each verb check flag VF_ON_TOOLBAR_BY_DEFAULT

— Add verb to toolbar if mentioned flag is set. Get an icon to use on toolbar from IBCVerb::Icon,IBCVerb::IconActive or IBCVerb::IconDown depending on toolbar button state

- Add verbs without parents to the top level of pulldown menu
- Add all other verbs to menu making sure that:
- Parents are added first, then children
- Verbs are added in appropriate order (use parameter IBCVerb-InsertBefore)
- There is a set of predefined browser menu items that may be referenced to as parents (i.e. BROWSER.FILE). These constants should be mapped to standard menu items (menu “File” in this case).
- If verb claims to have a parent but one does not exist verb should be inserted under other menu item (developers define which one).

2 Selection change

- Create an array of objects IBCObjectArray with CoCreateInstance()
- Fill array with object(s) selected by user
- Call IBCObjectArray::SetAsDefault for current session (parameter bDefault=TRUE)
- Change menu item state with IBCBrowser::ChangeMenuItem (see section Interfaces that describes implementation of IBCBrowser)

3 Execution of commands

- Execute selected verb using IBCObjectArray::DoVerb

4 Context menu

- An interface IBCObjectArray contains methods to operate with context menus containing commands available for an array of objects and execute this commands.
- A method IBCObjectArray::EnumVerbs may be called to receive a list of context menu commands available for the object array.
- The result of this call is a pointer to an enumerator of verbs IBCEnumVerbs that is a standard read-only VB-compatible

collection of objects CBCVerb. This enumeration starts from 1.

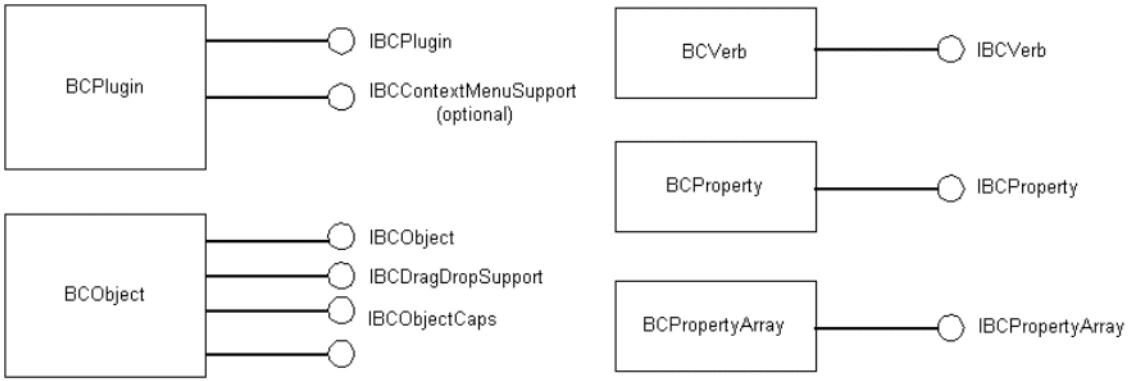
— Add verbs to context menu

— Execute selected verb using IBCObjectArray::DoVerb

Notifications

- The browser should call method Advise() on CBCObject before any object is shown in any pane to ensure notifications on changes are supported.
- When something happens with DM Extensions object client receives notification through method IBCObjectChanged::OnDataChange
- Checks the reason the notification was sent for (parameter whatHappened). The following codes should be supported:
  - — ODC_ADDOBJECT – if it were sent by container that is listed in the right pane shows a new object (pData) that was added
  - — ODC_REMOVEOBJECT - if it were sent by container that is listed in the right pane removes an object (pData) that were deleted
  - — ODC_CHANGEPROPERTY – updates a property (pData) in the right pane if it is shown on screen
  - — ODC_DELETED – removes an object from both panes
- Call method UnAdvise() on CBCObject if an object will be no longer shown anywhere in browser

Plug-in includes following objects:



BCPlugin is the main object of Plug-in. It must be the singleton object. IBCBrowserCore object creates it on its construction using the CoCreateInstanse(...) function. BCBrowserCore creates all Plug-ins registered in the system. You can register Plug-in using the method RegisterExtObject of the object IBCRegistrarExt.

See Also

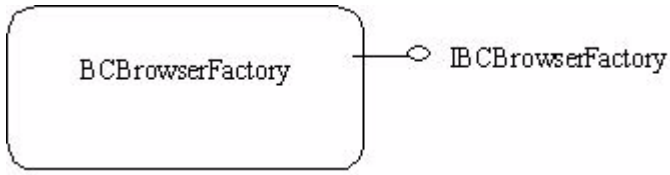
"IBCBrowserCore" on page 34

Explorer Core Objects

- "BCBrowserFactory" on page 19
- "BCCoreCtx" on page 20
- "BCObjectArray" on page 20
- "BCRegistrar" on page 21
- "BCSession" on page 22

BCBrowserFactory

The BCBrowserFactory object can be used to create a default browser.



C++ Syntax

```
const CLSID CLSID_BCBrowserFactory =
{0x13C52145,0xC799,0x11D1,{0x93,0xFE,0x00,0xA0,0x24,0x74,0x62,0x93
}};
IUnknown* pUnk;
HRESULT hr = ::CoCreateInstance(&CLSID_BCBrowserFactory, NULL,
CLSCTX_ALL, IID_IUnknown, (void**)&pUnk);
```

Visual Basic Syntax

```
Dim BrowserFactory
Set BrowserFactory = CreateObject("PwDUI.BCBrowserFactory")
```

When to Implement

Explorer Core extensions may implement such an object if they need a quick and easy way to create a browser.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

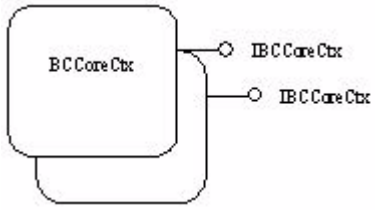
Header: import "DECore.idl"

See Also

"IBCBrowserFactory" on page 37

BCCoreCtx

The BCCoreCtx object maps unique keys to values. Once having inserted a key-value pair (element) into the map, you can efficiently retrieve or delete the pair using the key to access it.



C++ Syntax

```
const CLSID CLSID_BCCoreCtx =
{0x701B619C,0x59D5,0x11D1,{0xBE,0x36,0x00,0xA0,0x24,0x79,0xBD,0x73
}};

IUnknown* pUnk;

HRESULT hr = ::CoCreateInstance(&CLSID_BCCoreCtx, NULL,
CLSCTX_ALL, IID_IUnknown, (void**)&pUnk);
```

Visual Basic Syntax

```
Dim CoreCtx

Set CoreCtx = CreateObject("ExplorerCore.BCCoreCtx")
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

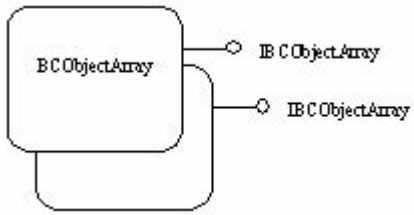
See Also

"IBCCoreCtx" on page 46

BCObjectArray

The BCObjectArray object represents an array of Explorer Core objects that can dynamically shrink and grow as necessary. Once having inserted an object into the array, you can efficiently retrieve or remove

the object from the array using the index to access it. Array indexes always start at position 0.



C++ Syntax

```
const CLSID CLSID_BCObjectArray =
{0x701B61A4,0x59D5,0x11D1,{0xBE,0x36,0x00,0xA0,0x24,0x79,0xBD,0x73
}};
IUnknown* pUnk;
HRESULT hr = ::CoCreateInstance(&CLSID_BCObjectArray, NULL,
CLSCTX_ALL, IID_IUnknown, (void**)&pUnk);
```

Visual Basic Syntax

```
Dim ObjectArray
Set ObjectArray = CreateObject("ExplorerCore.BCObjectArray")
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

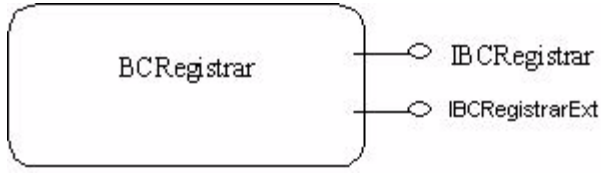
Header: import "DECore.idl"

See Also

"IBCObjectArray" on page 91

BCRegistrar

The BCRegistrar object implements the interface IBCRegistrar to register programmable objects.



C++ Syntax

```
const CLSID CLSID_BCRegistrar =
{0x6BD74205,0x14C8,0x14C8,{0xA2,0x38,0x00,0xA0,0x24,0x74,0x62,0x93
```

```
}};

IUnknown* pUnk;

HRESULT hr = ::CoCreateInstance(&CLSID_BCRegistrar, NULL,
CLSCTX_ALL, IID_IUnknown, (void**)&pUnk);
```

Visual Basic Syntax

```
Dim Registrar

Set Registrar = CreateObject("ExplorerCore.BCRegistrar")
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

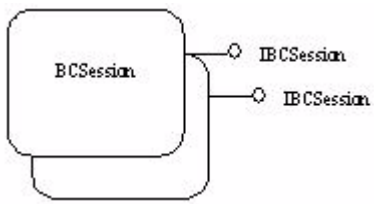
Header: import "DECore.idl"

See Also

"IBCRegistrar" on page 141

BCSession

The BCSession object represents an Explorer Core session. Every client connected to the core must have at least one session.



C++ Syntax

```
const CLSID CLSID_BCSession =
{0x5C8B2042,0x55C0,0x11D1,{0xBE,0x36,0x00,0xA0,0x24,0x79,0xBD,0x73
}};

IUnknown* pUnk;

HRESULT hr = ::CoCreateInstance(&CLSID_BCSession, NULL,
CLSCTX_ALL, IID_IUnknown, (void**)&pUnk);
```

Visual Basic Syntax

```
Dim Session

Set Session = CreateObject("ExplorerCore.BCSession")
```

When to Implement

The session object is implemented by the core and manages data (or plug-ins) that should be visually presented by a browser associated with the session. A programmer should not implement it.

When to Use

It can be used to identify a particular client that calls on some object.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

See Also

"IBCSession" on page 153

Explorer Core Interfaces

- "IBCBrowser" on page 25
- "IBCBrowser2" on page 30
- "IBCBrowser3" on page 33
- "IBCBrowserCore" on page 34
- "IBCBrowserFactory" on page 37
- "IBCContextMenuSupport" on page 38
- "IBCCoreCtx" on page 46
- "IBCDragDropSupport" on page 52
- "IBCEnumObjects" on page 57
- "IBCEnumProperties" on page 68
- "IBCEnumVerbs" on page 74
- "IBCObject" on page 80
- "IBCObject2" on page 90
- "IBCObjectArray" on page 91
- "IBCObjectCaps" on page 102
- "IBCObjectCaps2" on page 107
- "IBCObjectChanged" on page 110
- "IBCPlugin" on page 111
- "IBCProperty" on page 119
- "IBCPropertyArray" on page 127
- "IBCRegistrar" on page 141
- "IBCRegistrarExt" on page 145
- "IBCRetrievalSink" on page 151
- "IBCSession" on page 153
- "IBCVerb" on page 160

IBCBrowser

The BCBrowser object represents a browser that visually presents any Explorer Core objects.

When to Implement

Any client that wants to visualize Explorer Core objects should implement this object. DM Extensions implements IBCBrowser objects for the Windows Explorer DM Extension, the Microsoft Outlook DM Extension

Methods

IBCBrowser Methods	Description
IBCBrowser::ChangeMenuItem	The method changes the state of a command.
IBCBrowser::BrowseObject	The method tells the browser to browse to another object.
IBCBrowser::GetParentWindow	The method returns the handle of the window that is to be used as the parent window when displaying modal dialogs.
IBCBrowser::GetBrowserInfo	The method returns browser attributes.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

IBCBrowser::BrowseObject

The method tells the browser to browse to another object.

C++ Syntax

```
HRESULT BrowseObject(
  BSTR bstrPath,
  short sFlags
);
```

Visual Basic Syntax

```
BCBrowser.BrowseObject (bstrPath, sFlags)
```

Parameters

bstrPath	[in] A string that specifies an object's location. This value depends on the flag or flags set in the flags parameter.
sFlags	[in] An integer specifying the object to be browsed. The parameter can be zero, one or more of the following values. This flag specifies the bstrPath parameter's category. For possible values see BrowseObjectFlags Enumeration enumeration.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

HERE

Remarks

Clients can use this method to force a browser to browse to a specific object on plug-ins. Typically, these are objects contained in the list.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCBrowser" on page 25

"BrowseObjectFlags Enumeration" on page 174

IBCBrowser::ChangeMenuItem

The method changes the state of a command.

C++ Syntax

```
HRESULT ChangeMenuItem(
  BSTR bstrVerbID,
  long lState
);
```

Visual Basic Syntax

```
BCBrowser.ChangeMenuItem ( bstrVerbID,  lState)
```

Parameters

bstrVerbID	[in] A string that uniquely identifies the command.
lState	[in] A long that specifies the command state. For possible values see VerbFlags Enumeration.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCBrowser" on page 25

"VerbFlags Enumeration" on page 182

IBCBrowser::GetBrowserInfo

The method returns browser attributes.

C++ Syntax

```
HRESULT GetBrowserInfo(
  long* plBrowserInfo
);
```

Visual Basic Syntax

```
[ plBrowserInfo  = ] BCBrowser.GetBrowserInfo
```

Parameters

plBrowserInfo	[out, retval] A long that specifies browser attributes. It can be zero or a combination of the values from BrowserInfoFlags enumeration.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCBrowser" on page 25

"BrowserInfoFlags Enumeration" on page 174

IBCBrowser::GetParentWindow

The method returns the handle of the window that is to be used as the parent window when displaying modal dialogs.

C++ Syntax

```
HRESULT GetParentWindow(
  long* plHandle
);
```

Visual Basic Syntax

```
[ plHandle  = ] BCBrowser.GetParentWindow
```

Parameters

plHandle	[out, retval] A long that specifies the window handle.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

Clients can use this method to force a browser to view a specific object on plug-ins. Typically, these are objects contained in the list.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCBrowser" on page 25

IBCBrowser interface IDL definition

```
[
   object,
   uuid(2E2EF695-AD1D-11D1-8AF4-00609775D351),
   helpstring( "An object that provides visual representation of
data." ),
   oleautomation,
   pointer_default(unique)
]
interface IBCBrowser : IUnknown
{
   HRESULT ChangeMenuItem( [in] BSTR bstrVerbID, [in] long
lState);
   HRESULT BrowseObject( [in] BSTR bstrPath, [in] short sFlags);
   HRESULT GetParentWindow([out, retval] long* plHandle);
   HRESULT GetBrowserInfo(([out, retval] long* plBrowserInfo);
}
```

IBCBrowser2

The IBCBrowser2 interface extends functionality of BCBrowser object

Methods

IBCBrowser2 Methods	Description
IBCBrowser2::Refresh	The method refreshes browser contents.
IBCBrowser2::AddMenuItem	The method adds new item to main menu and/or toolbar.
IBCBrowser2::RemoveMenuItem	The method removes menu item from main menu and/or toolbar.
IBCBrowser2::UpdateMenuItem	The method updates menu item and/or toolbar button.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: import "DECore.idl"

See Also

"IBCBrowser" on page 25

IBCBrowser2::AddMenuItem

The method adds a new item to the main menu and/or toolbar.

C++ Syntax

```
HRESULT AddMenuItem(
  IBCVerb* pNewItem
);
```

Visual Basic Syntax

```
BCBrowser2.AddMenuItem (pNewItem)
```

Parameters

pNewItem	[in]A new item that should be added.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "DECore.idl"

See Also

"IBCBrowser2" on page 30

"IBCVerb" on page 160

IBCBrowser2::Refresh

The method refreshes browser's contents.

C++ Syntax

```
HRESULT Refresh(
  long lReserved
);
```

Visual Basic Syntax

```
BCBrowser2.Refresh (lReserved)
```

Parameters

lReserved	[in] Reserved value. Should be zero.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

Clients can use this method to force a browser to refresh its contents.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "DECore.idl"

See Also

IBCBrowser2

IBCBrowser2::RemoveMenuItem

The method removes a menu item from the main menu and/or toolbar.

C++ Syntax

```
HRESULT RemoveMenuItem(
  BSTR bstrVerbID
);
```

Visual Basic Syntax

```
BCBrowser2.RemoveMenuItem (bstrVerbID)
```

Parameters

bstrVerbID	[in] A menu item that should be removed from the main menu and/or toolbar.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "DECore.idl"

See Also

"IBCBrowser2" on page 30

IBCBrowser2::UpdateMenuItem

The method updates a menu item and/or toolbar button.

C++ Syntax

```
HRESULT UpdateMenuItem(
  IBCVerb* pItem
);
```

Visual Basic Syntax

```
BCBrowser2.UpdateMenuItem (pItem)
```

Parameters

pItem	[in] A menu item that should be updated.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "DECore.idl"

See Also

"IBCBrowser2" on page 30

"IBCVerb" on page 160

IBCBrowser3

The IBCBrowser3 interface extends functionality of BCBrowser object

Methods

IBCBrowser3 Methods	Description
IBCBrowser3::GetBrowserInfoEx	The method returns extended information about browser.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: import "DECore.idl"

See Also

“IBCBrowser” on page 25

IBCBrowser3::GetBrowserInfoEx

The method returns extended information about a browser.

C++ Syntax

```
HRESULT GetBrowserInfoEx(  IBCCoreCtx**  ppCtx);
```

Visual Basic Syntax

```
[ Set  ppCtx  = ] BCBrowser3.GetBrowserInfoEx
```

Parameters

ppCtx	[out, retval] BCCoreCtx object that contains additional browser information. This parameter can contain the following values: BrowserCommandLine: Command line that has been used to run application.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

Clients can use this method to force a browser to view a specific object on plug-ins. Typically, these are objects contained in the list.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "DECore.idl"

See Also

“IBCBrowser3” on page 33

“IBCCoreCtx” on page 46

IBCBrowserCore

The BCBrowserCore object is the main Explorer Core object. There's only one BCBrowserCore object.

When to Implement

The object is passed to any plug-in while initializing the plug-in. It is not recommended for Plug-ins to store a reference to the IBCBrowserCore to avoid cross references. Instead of this Plug-ins should implement IConnectionPointContainer interface and IBCBrowserCore connection point entry. During loading plug-ins BCBrowserCore object will try to connect to this connection point.

Methods

IBCBrowserCore Methods	Description
IBCBrowserCore::GetBrowser	The method retrieves the browser associated with a given session.
IBCBrowserCore::GetSession	Returns a session by its identifier.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

IBCBrowserCore::GetBrowser

The method retrieves the browser associated with a given session.

C++ Syntax

```
HRESULT GetBrowser(
  BSTR bstrSID,
  short sFlags,
  IBCBrowser**  ppBrowser
);
```

Visual Basic Syntax

```
[ Set ppBrowser  = ] BCBrowserCore.GetBrowser (bstrSID, sFlags)
```

Parameters

bstrSID	[in] A string that uniquely identifies the session object.
sFlags	[in] An integer that specifies whether or not a browser should be started and registered for the session, if there's no browser associated with the session. This parameter can be one of the values from GetBrowserFlags enumeration.
ppBrowser	[out, retval] Returns BCBrowser object that specifies the browser associated with the session.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCBrowserCore" on page 34

"IBCBrowser" on page 25

"GetBrowserFlags Enumeration" on page 177

IBCBrowserCore::GetSession

The method retrieves a session by its identifier.

C++ Syntax

```
HRESULT GetSession(
  BSTR bstrSID,
  IBCSession** ppSession);
```

Visual Basic Syntax

```
[ Set ppSession  = ] BCBrowserCore.GetSession (bstrSID)_
```

Parameters

bstrSID	[in] A string that uniquely identifies the session object.
ppSession	[out, retval] Returns BCSession object with the supplied identifier.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCBrowserCore" on page 34

"IBCSession" on page 153

IBCBrowserFactory

The BCBrowserFactory object can be used to create a default browser.

When to Implement

Explorer Core extensions may implement such an object if they needs a quick and easy way to create a browser.

Methods

IBCBrowserFactory Methods	Description
IBCBrowserFactory::CreateBrowser	The method creates a default browser for a specified session.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

IBCBrowserFactory::CreateBrowser

The method creates a default browser for a specified session.

C++ Syntax

```
HRESULT CreateBrowser(
  IBCSession* pSession,
  IBCBrowser** ppBrowser
);
```

Visual Basic Syntax

```
[ Set ppBrowser  = ] BCBrowserFactory.CreateBrowser (pSession)
```

Parameters

fff	[in] A BCSession object that identifies the session the browser is to be created for.
ppBrowser	[out, retval] Returns BCBrowser object that specifies the created browser.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCBrowser" on page 25

"IBCSession" on page 153

IBCContextMenuSupport

The IBCContextMenuSupport interface makes it possible to manage the commands (menu and toolbar items) the plug-in provides for its objects.

When to Implement

The BCContextMenuSupport object is implemented by a plug-in.

Methods

IBCContextMenuSupport Methods	Description
IBCContextMenuSupport::DoVerb	The method executes a specified command for all the plug-in objects in a given array.
IBCContextMenuSupport::EnumVerbs	The method enumerates all commands available to the plug-in objects in a given array.
IBCContextMenuSupport::EnumAllVerbs	The method enumerates all commands available for plug-in objects.
IBCContextMenuSupport::SetAsDefault	The method sets or clears the plug-in objects in a given array as default.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

IBCContextMenuSupport::DoVerb

The method executes a specified command for all the plug-in objects in the given array.

C++ Syntax

```
HRESULT DoVerb(
  BSTR bstrSID,
  BSTR bstrUIContext,
  BSTR bstrCmdID,
  long grfKeyState,
  IBCObject* pParent,
  IBCObjectArray* pObjArray);
```

Visual Basic Syntax

```
BCContextMenuSupport.DoVerb (bstrSID, bstrUIContext, bstrCmdID,
grfKeyState, pParent, pObjArray)
```

Parameters

bstrSID	[in] A string that uniquely identifies the session.
bstrUIContext	[in] A string that specifies the operation context that is to be used.
bstrCmdID	[in] A string that uniquely identifies the command to execute. The following predefined values can be used: DEFAULT: Default command. Issued when user double clicks on item. BROWSER.DELETE: Delete command. Issued when user presses “Delete“ key. BROWSER.PRINT: Print command. Issued when user selects Print command. BROWSER.CONTEXTHELP: Context help command. Issued when user presses F1 key.
grfKeyState	[in] A long that specifies the control keys being pressed. This parameter can be zero or a combination of the KeyFlags enumerator values.
pParent	[in] A BCObject object that specifies the parent for all the objects in the array, e.g. the root object of the list where the array entries were selected. The parameter may be Nothing if there is no such object.
pObjArray	[in] A BCObjectArray object that contains objects to which the command relates.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCContextMenuSupport" on page 38

"IBCObject" on page 80

"IBCObjectArray" on page 91

"KeyFlags Enumeration" on page 177

IBCContextMenuSupport::EnumAllVerbs

The method enumerates all commands available for plug-in objects.

C++ Syntax

```
HRESULT EnumAllVerbs(
  BSTR bstrSID,
  IBCEnumVerbs** ppEnumVerbs
);
```

Visual Basic Syntax

```
[ Set ppEnumVerbs  = ] BCContextMenuSupport.EnumAllVerbs (bstrSID)
```

Parameters

bstrSID	[in] A string that uniquely identifies the session.
ppEnumVerbs	[out, retval] Returns a BCEnumVerbs object that enumerates the requested commands.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

The method is used by the EnumAllVerbs method on the session when initially filling the menu and toolbar to enumerate all commands available to a plug-in.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCContextMenuSupport" on page 38

"IBCEnumVerbs" on page 74

IBCContextMenuSupport::EnumVerbs

The method enumerates all commands available to the plug-in objects in the given array.

C++ Syntax

```
HRESULT EnumVerbs(
  BSTR bstrSID,
  BSTR bstrUIContext,
  IBCObjectArray* pObjArray,
  IBCObject* pParent,
  IBCEnumVerbs** ppEnumVerbs
);
```

Visual Basic Syntax

```
[ Set ppEnumVerbs  = ] BCContextMenuSupport.EnumVerbs (bstrSID,
bstrUIContext, pObjArray, pParent)
```

Parameters

bstrSID	[in] A string that uniquely identifies the session.
bstrUIContext	[in] A string that specifies the operation context that is to be used.
pObjArray	[in] A BCObjectArray object that contains objects to which the command relates.
grfKeyState	[in] A long that specifies the control keys being pressed. This parameter can be zero or a combination of the KeyFlags enumerator values.
pParent	[in] A BCObject object that specifies the parent for all the objects in the array, e.g. the root object of a list where the array entries were selected. The parameter may be Nothing if there is no such object.
pParent	[out, retval] Returns a BCEnumVerbs object that enumerates the requested commands.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

“IBCContextMenuSupport” on page 38

“IBCObjectArray” on page 91

“IBCObject” on page 80

“IBCEnumVerbs” on page 74

“KeyFlags Enumeration” on page 177

IBCContextMenuSupport::SetAsDefault

The method sets or clears the plug-in objects in the given array as defaults.

C++ Syntax

```
HRESULT SetAsDefault(
  BSTR bstrSID,
  VARIANT_BOOL bDefault,
  IBCObjectArray* pObjArray,
  IBCObject* pParent
);
```

Visual Basic Syntax

```
BCContextMenuSupport.SetAsDefault (bstrSID, bDefault, pObjArray,
pParent)
```

Parameters

bstrSID	[in] A string that uniquely identifies the session.
bDefault	[in] A Boolean value that indicates whether the object array should be set as default or unset. True: Array should be default. False: Array should not be default.
pObjArray	[in] A BCObjectArray object that contains objects that should be set as defaults or cleared.
pParent	[in] A BCObject object that specifies the parent for all the objects in the array, e.g. the root object of a list where the array entries were selected. The parameter may be Nothing if there is no such object.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

The method is called when there needs to set an array as default indicating that its objects were highlighted or, on the contrary, the user cleared selection. UI should respond to this action by changing the desktop menu, toolbar, and other elements.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCContextMenuSupport" on page 38

"IBCObjectArray" on page 91

"IBCObject" on page 80

IBCCoreCtx

The BCCoreCtx object maps unique keys to values. Once you have inserted a key-value pair (element) into the map, you can efficiently retrieve or delete the pair using the key to access it.

A BCCoreCtx object may be created with the CreateObject function or a Dim statement with the New keyword.

Methods

IBCCoreCtx Methods	Description
IBCCoreCtx::RegisterObjectParam	The method inserts an object into the map; replaces an existing element if a matching key is found.
IBCCoreCtx::GetObjectParam	The method looks up the object value mapped to a given key.
IBCCoreCtx::RevokeObjectParam	The method removes an object element specified by a key.
IBCCoreCtx::RegisterVariantParam	The method inserts a variant into the map; replaces an existing element if a matching key is found.
IBCCoreCtx::GetVariantParam	The method looks up the variant value mapped to a given key.
IBCCoreCtx::RevokeVariantParam	The method removes a variant element specified by a key.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

IBCCoreCtx::GetObjectParam

The method looks up the object value mapped to a given key.

C++ Syntax

```
HRESULT GetObjectParam(
  BSTR bstrKey,
  IUnknown** ppUnk
);
```

Visual Basic Syntax

```
[ Set ppUnk  = ] BCCoreCtx.GetObjectParam (bstrKey)
```

Parameters

bstrKey	[in]A string specifies the key that identifies the element to be looked up.
ppUnk	[out, retval] Returns an object that identifies the looked-up value.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCCoreCtx" on page 46

IBCCoreCtx::GetVariantParam

The method looks up the variant value mapped to a given key.

C++ Syntax

```
HRESULT GetVariantParam(  BSTR bstrKey,  VARIANT* pVar);
```

Visual Basic Syntax

```
[ pVar  = ] BCCoreCtx.GetVariantParam (bstrKey)
```

Parameters

bstrKey	[in]A string specifies the key that identifies the element to be looked up.
pVar	[out, retval] Returns a variant that identifies the looked-up value.

Example

The following example creates a map, inserts a value into it, and then displays the value.

```
Dim Ctx As New BCCoreCtx
```

```
Ctx.RegisterVariantParam "Key", 7
MsgBox Ctx.GetVariantParam("Key")
```

Remarks

The following predefined keys can be used to retrieve all registered variant parameters and object parameters:

Key	Value
%AllVariantKeys	An array (SAFEARRAY) that contains keys for all registered variant parameters.
%AllObjectKeys	An array (SAFEARRAY) that contains keys for all registered object parameters.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCCoreCtx" on page 46

IBCCoreCtx::RegisterObjectParam

The method inserts an object into the map and replaces an existing element if a matching key is found.

C++ Syntax

```
HRESULT RegisterObjectParam(
  BSTR bstrKey,
  IUnknown* pUnk
);
```

Visual Basic Syntax

```
BCCoreCtx.RegisterObjectParam (bstrKey, pUnk)
```

Parameters

bstrKey	[in] A string that specifies the key of the new element.
pUnk	[in] An object that specifies the value of the new element.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

The method is used to insert an object element in a map. First, the key is looked up. If the key is found, then the corresponding value is changed; otherwise a new key-value pair is created.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCCoreCtx" on page 46

IBCCoreCtx::RegisterVariantParam

The method inserts a variant into the map; replaces an existing element if a matching key is found.

C++ Syntax

```
HRESULT RegisterVariantParam(
  BSTR bstrKey,
  VARIANT var,
);
```

Visual Basic Syntax

```
BCCoreCtx.RegisterVariantParam (bstrKey, var)
```

Parameters

bstrKey	[in] A string that specifies the key of the new element.
var	[in] A variant that specifies the value of the new element.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

The method is used to insert a variant element in a map. First, the key is looked up. If the key is found, then the corresponding value is changed; otherwise a new key-value pair is created.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCCoreCtx" on page 46

IBCCoreCtx::RevokeObjectParam

The method removes an object element specified by a key.

C++ Syntax

```
HRESULT RevokeObjectParam(
  BSTR bstrKey
);
```

Visual Basic Syntax

```
BCCoreCtx.RevokeObjectParam (bstrKey)
```

Parameters

bstrKey	[in] A string that specifies the key for the element to be removed.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

The method looks up the map entry corresponding to the supplied key; then, if the key is found, removes the entry.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCCoreCtx" on page 46

IBCCoreCtx::RevokeVariantParam

The method removes a variant element specified by a key.

C++ Syntax

```
HRESULT RevokeVariantParam(
  BSTR bstrKey
);
```

Visual Basic Syntax

```
BCCoreCtx.RevokeVariantParam (bstrKey)
```

Parameters

bstrKey	[in] A string that specifies the key for the element to be removed.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

The method looks up the map entry corresponding to the supplied key; then, if the key is found, removes the entry.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCCoreCtx" on page 46

IBCDragDropSupport

The IBCDragDropSupport makes it possible to manage drag and drop operations the plug-in provides for its objects.

When to Implement

The IBCDragDropSupport object is implemented by a plug-in.

When to Use

The IBCDragDropSupport is used by Explorer Core to manage drag and drop operations.

Methods

IBCDragDropSupport Methods	Description
IBCDragDropSupport::CanDrag	The method checks whether the object can be dragged.
IBCDragDropSupport::CanDropBetweenObjects	The method checks whether an object allows reordering.

IBCDragDropSupport Methods	Description
IBCDragDropSupport::DropEffect	The method returns drop effect for a specified source data object.
IBCDragDropSupport::Drop	The method performs a drop operation for a specified source data object.
IBCDragDropSupport::GetMenuItems	The method returns actions that can be performed when dragging by the right mouse button.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

IBCDragDropSupport::CanDrag

The method checks whether the object can be dragged.

C++ Syntax

```
HRESULT CanDrag(  VARIANT_BOOL* pbDrag);
```

Visual Basic Syntax

```
[ pbDrag = ] BCDragDropSupport.CanDrag
```

Parameters

pbDrag	[out, retval] A Boolean value that indicates whether the object can be dragged. VARIANT_TRUE: The object can be dragged. VARIANT_FALSE: The object cannot be dragged

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IBCDragDropSupport" on page 52

IBCDragDropSupport::CanDropBetweenObjects

The method checks whether an object allows reordering.

C++ Syntax

```
HRESULT CanDropBetweenObjects(  IBCCoreCtx* pCoreCtx ,
VARIANT_BOOL* pbDropBetween );
```

Visual Basic Syntax

```
[ pbDropBetween ] = BCDragDropSupport.Drop ( pCoreCtx )
```

Parameters

pCoreCtx	[in] BCCoreCtx parameter that contains current context.
pbDropBetween	[out, retval] A Boolean value that indicates whether the object allows reordering. VARIANT_TRUE: The object allows reordering. VARIANT_FALSE: The object does not allow reordering.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IBCDragDropSupport" on page 52

IBCDragDropSupport::Drop

The method performs a drop operation for a specified source data object.

C++ Syntax

```
HRESULT Drop(  IBCCoreCtx* pCoreCtx ,  VARIANT vKeyState ,
IDataObject* pDataObject);
```

Visual Basic Syntax

```
BCDragDropSupport.Drop pCoreCtx , vKeyState , pDataObject
```

Parameters

pCoreCtx	[in] BCCoreCtx parameter that contains current context.
vKeyState	[in] Keyboard status from KeyFlags.
pDataObject	[in] Source data object.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IBCDragDropSupport" on page 52

IBCDragDropSupport::DropEffect

The method returns drop effect for a specified source data object.

C++ Syntax

```
HRESULT DropEffect(  IBCCoreCtx* pCoreCtx ,  long lKeyState ,
IDataObject* pDataObject  long* plEffect);
```

Visual Basic Syntax

```
[ plEffect ] = BCDragDropSupport.DropEffect ( pCoreCtx , lKeyState
, pDataObject )
```

Parameters

pCoreCtx	[in] BCCoreCtx parameter that contains current context.
lKeyState	[in] Keyboard status from KeyFlags.
pDataObject	[in] Source data object.
plEffect	[out, retval] Value that indicates drop effect. DROPEFFECT_NONE (0): Drop target cannot accept the data.Drop target cannot accept the data. DROPEFFECT_COPY (1): Drop results in a copy. The original data is untouched by the drag source. DROPEFFECT_MOVE (2): Drag source should remove the data. DROPEFFECT_LINK (4): Drag source should create a link to the original data. False: The object does not allow reordering.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IBCDragDropSupport" on page 52

IBCDragDropSupport::GetMenuItems

The method returns actions that can be performed when dragging by the right mouse button.

C++ Syntax

```
HRESULT GetMenuItems(  IBCCoreCtx* pCoreCtx ,  long lKeyState ,
IDataObject* pDataObject  IBCEnumVerbs** ppEnum);
```

Visual Basic Syntax

```
[ Set ppEnum ] = BCDragDropSupport.DropEffect ( pCoreCtx ,
lKeyState , pDataObject )
```

Parameters

pCoreCtx	[in] BCCoreCtx parameter that contains current context.
lKeyState	[in] Keyboard status from KeyFlags.
pDataObject	[in] Source data object.
ppEnum	[out, retval] BCEnumVerbs object that contains context menu verbs.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IBCDragDropSupport" on page 52

IBCEnumObjects

IBCEnumObjects interface can be used to enumerate through the root objects of a plug-in or the child objects of a BCObject.

When to Implement

The BCEnumObjects object is implemented by plug-ins to enumerate BCObject objects.

Methods

IBCEnumObjects Methods	Description
IBCEnumObjects::Next	The method retrieves a specified number of items in the enumeration sequence.
IBCEnumObjects::Skip	The method skips over a specified number of items in the enumeration sequence.
IBCEnumObjects::Reset	The method resets the enumeration sequence to the beginning.
IBCEnumObjects::Clone	The method creates another enumerator that contains the same enumeration state as the current one.
IBCEnumObjects::get_Size	The method returns the number of items in the enumerator.
IBCEnumObjects::Prev	The method retrieves a specified number of previous items in the enumeration sequence.
IBCEnumObjects::Find	The method searches the enumerator sequentially to find the first item matching specified criteria.
IBCEnumObjects::ScrollToEnd	The method scrolls the enumeration sequence to the end.
IBCEnumObjects::Cancel	The method cancels background fetching items in enumerator.
IBCEnumObjects::Retrieve	The method makes the enumerator to start background fetching items.
IBCEnumObjects::OrderBy	The method resorts the enumeration sequence by specified criteria.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

See Also

"IBCObject" on page 80

IBCEnumObjects::Cancel

The method cancels background fetching items in enumerator.

C++ Syntax

```
HRESULT Cancel();
```

Visual Basic Syntax

```
BCEnumObjects.Cancel
```

Parameters

None

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCEnumObjects" on page 57

IBCEnumObjects::Clone

The method creates another enumerator that contains the same enumeration state as the current one.

C++ Syntax

```
HRESULT Clone(
  IBCEnumObjects**  ppEnum
);
```

Visual Basic Syntax

```
[ Set  ppEnum  = ] BCEnumObjects.Clone
```

Parameters

ppEnum	[in] A string that specifies an object's location. This value depends on the flag or flags set in the flags parameter.
sFlags	[out,retval] Returns a BCEnumObjects object as a copy of this enumerator.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

Using this method, a client can record a particular point in the enumeration sequence, and then return to that point later.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCEnumObjects" on page 57

IBCEnumObjects::Find

The method sequentially searches the enumerator to find the first item matching specified criteria.

C++ Syntax

```
HRESULT Find(
  BSTR bstrCriteria,
  long* plStartAt,
);
```

Visual Basic Syntax

```
[ plStartAt  = ] BCEnumObjects.Find (bstrCriteria)
```

Parameters

bstrCriteria	[in]A string that specifies the search criteria.
plStartAt	[out,retval] An array of size celt (or larger) with the BCObject elements of your interest.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

There are no rules what criteria should be understood by an enumerator but usually it's the DisplayName object's property. The Document Management plug-in understands the names and numbers of documents when retrieving them or DisplayName of other objects.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCEnumObjects" on page 57

"IBCObject" on page 80

IBCEnumObjects::get_Size

The method returns the number of items in the enumerator.

C++ Syntax

```
HRESULT get_Size(
  long* plCount
);
```

Visual Basic Syntax

```
[ plCount  = ] BCEnumObjects.Size
```

Parameters

plCount	[out,retval] Returns a long that specifies the number of elements in the enumerator or –1, if the enumerator supports background fetching.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCEnumObjects" on page 57

IBCEnumObjects::Next

The method retrieves a specified number of items in the enumeration sequence.

C++ Syntax

```
HRESULT Next(
  long celt,
  IBCObject**  rgelt,
  long*  pceltFetched
);
```

Visual Basic Syntax

```
[ Set  pceltFetched  = ] BCEnumObjects.Next (celt,  rgelt)
```

Parameters

pceltFetched	[out, retval] Returns the number of elements actually supplied in rgelt.
celt	[in] A long that specifies the number of elements being requested.
rgelt	[out] An array of size celt (or larger) with the BCObject elements of your interest.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

The method retrieves the next count items in the enumeration sequence. If there are fewer than the requested number of elements left in the sequence, it retrieves the remaining elements. The number of elements actually retrieved is returned.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCEnumObjects" on page 57

"IBCObject" on page 80

IBCEnumObjects::OrderBy

The method resorts the enumeration sequence by specified criteria.

C++ Syntax

```
HRESULT OrderBy(
  BSTR bstrCriteria
);
```

Visual Basic Syntax

```
BCEnumObjects.OrderBy (bstrCriteria)
```

Parameters

bstrCriteria	[in] A string that specifies the sort criteria. It has the following format: ColumnID1[:SortOrder1][;ColumnID 2[:SortOrder2]...] where ColumnID1, ColumnID2 are column identifiers, SortOrder1, SortOrder2 are “A” (ascending sort order) or “D” (descending). The default value is “A”.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

A client may call this method if it's necessary to use existing enumerator but with different sort order.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

“IBCEnumObjects” on page 57

IBCEnumObjects::Prev

The method retrieves a specified number of previous items in the enumeration sequence.

C++ Syntax

```
HRESULT Prev(  long celt,  IBCObject**  rgelt,  long*
pceltFetched );
```

Visual Basic Syntax

```
[ pceltFetched  = ] BCEnumObjects.Prev (celt,  rgelt)
```

Parameters

celt	[in] A long that specifies the number of elements being requested.
rgelt	[out] An array of size celt (or larger) with the BCObject elements of your interest.
pceltFetched	[out, retval] Returns the number of elements actually supplied in objects.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

The method retrieves the previous count items in the enumeration sequence. If there are fewer than the requested number of elements left in the beginning of the sequence, it retrieves the remaining elements. The number of elements actually retrieved is returned.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCEnumObjects" on page 57

"IBCObject" on page 80

IBCEnumObjects::Reset

The method resets the enumeration sequence to the beginning.

C++ Syntax

```
HRESULT Reset();
```

Visual Basic Syntax

```
BCEnumObjects.Reset
```

Parameters

None

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCEnumObjects" on page 57

IBCEnumObjects::Retrieve

The method makes the enumerator to start background fetching items.

C++ Syntax

```
HRESULT Retrieve(
  long lEnumID
);
```

Visual Basic Syntax

```
BCEnumObjects.Retrieve ( lEnumID)
```

Parameters

lEnumID	[in] An identifier that is used in notification.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

A client should call this method if an enumerator supports background enumeration (get_size() method returns –1) after client was connected to enumerator using the IRetrievalSink connection point entry. A client should specify the unique lEnumID parameter if using the same

IRetrievalSink implementation to receive notification from multiple enumerators.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCEnumObjects" on page 57

IBCEnumObjects::ScrollToEnd

The method scrolls the enumeration sequence to the end.

C++ Syntax

```
HRESULT ScrollToEnd();
```

Visual Basic Syntax

```
BCEnumObjects.ScrollToEnd
```

Parameters

None

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCEnumObjects" on page 57

IBCEnumObjects::Skip

The method skips over a specified number of items in the enumeration sequence.

C++ Syntax

```
HRESULT Skip(
  long celt
);
```

Visual Basic Syntax

```
BCEnumObjects.Skip (celt)
```

Parameters

celt	[in] A long that specifies the number of elements to be skipped.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCEnumObjects" on page 57

IBCEnumProperties

The BCEnumProperties object represents a collection of object's properties. The collection may be retrieved with the EnumContentProperties method on a BCObject object.

When to Implement

The BCEnumProperties object is implemented by plug-ins.

Methods

IBCEnumProperties Methods	Description
IBCEnumProperties::Add	The method adds a new item to a collection.
IBCEnumProperties::Remove	The method removes a specific item from a collection.
IBCEnumProperties::Clear	The method removes all items from a collection.

Properties

IBCEnumProperties Properties	Description
IBCEnumProperties::Item	The property returns a specific member of a collection either by position or by key.
IBCEnumProperties::Count	The property returns the number of items in a collection.
IBCEnumProperties::_NewEnum	The standard property for collection. Allows to use this collection to cycle of Visual Basic.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

See Also

"IBCObject" on page 80

IBCEnumProperties::_NewEnum

The standard property for collection. Allows you to use this collection to cycle of Visual Basic.

C++ Syntax

```
HRESULT _NewEnum(  LPUNKNOWN* ppEnum);
```

Parameters

ppEnum	[out, retval] Returns the collection.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Access:

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCEnumProperties" on page 68

IBCEnumProperties::Add

The method adds a new item to a collection.

C++ Syntax

```
HRESULT Add(
  IBCProperty* pProp
);
```

Visual Basic Syntax

```
BCEnumProperties.Add (pProp)
```

Parameters

pProp	[in] A BCProperty object that specifies the item to add to the collection.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCEnumProperties" on page 68

"IBCProperty" on page 119

IBCEnumProperties::Clear

The method removes all items from a collection.

C++ Syntax

```
HRESULT Clear();
```

Visual Basic Syntax

```
BCEnumProperties.Clear
```

Parameters

None

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCEnumProperties" on page 68

IBCEnumProperties::Count

The property returns the number of items in a collection.

C++ Syntax

```
HRESULT Count(
  long* plCount
);
```

Visual Basic Syntax

```
[ plCount = ] BCEnumProperties.Count
```

Parameters

plCount	[out, retval] Returns a long that specifies the number of items in the collection.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Access:

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCEnumProperties" on page 68

IBCEnumProperties::Item

The property returns a specific member of a collection either by position or by key.

C++ Syntax

```
HRESULT Item(  VARIANT vIndex,  IBCProperty** ppProp);
```

Visual Basic Syntax

```
[ Set ppProp = ] BCEnumProperties.Item (vIndex)
```

Parameters

vIndex	[in] A numeric expression that specifies the position of a member of the collection. An index must be a number from 1 to the value of the collection's Count property.
ppProp	[out, retval] Returns a BCProperty object that specifies the requested item.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Access:

Read Only

Remarks

If the value provided as index does not match any existing member of the collection, an error occurs. Item is the default property for a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCEnumProperties" on page 68

"IBCProperty" on page 119

IBCEnumProperties::Remove

The method removes a specific item from a collection.

C++ Syntax

```
HRESULT Remove(
  VARIANT vIndex
);
```

Visual Basic Syntax

```
BCEnumProperties.Remove (vIndex)
```

Parameters

vIndex	[in] A numeric expression specifying the index in the collection of the object to be removed.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCEnumProperties" on page 68

IBCEnumVerbs

The BCEnumVerbs object represents a collection of menu or toolbar commands (verbs).

Methods

IBCEnumVerbs Methods	Description
IBCEnumVerbs::Add	The method adds a new item to a collection.
IBCEnumVerbs::Remove	The method removes a specific item from a collection.
IBCEnumVerbs::Clear	The method removes all items from a collection.

Properties

IBCEnumVerbs Properties	Description
IBCEnumVerbs::Item	The property returns a specific member of a collection either by position or by key.
IBCEnumVerbs::Count	The property returns the number of items in a collection.
IBCEnumVerbs::_NewEnum	The standard property for collection. Allows to use this collection to cycle of Visual Basic.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

IBCEnumVerbs::_NewEnum

The standard property for a collection. Allows you to use this collection to cycle of Visual Basic.

C++ Syntax

```
HRESULT _NewEnum(  LPUNKNOWN* ppEnum);
```

Parameters

ppEnum	[out, retval] Returns the collection.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Access

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCEnumVerbs" on page 74

IBCEnumVerbs::Add

The method adds a new item to a collection.

C++ Syntax

```
HRESULT Add(
  IBCVerb* pVerb
);
```

Visual Basic Syntax

```
BCEnumVerbs.Add (pVerb)
```

Parameters

pVerb	[in] A BCVerb object that specifies the item to add to the collection.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCEnumVerbs" on page 74

"IBCVerb" on page 160

IBCEnumVerbs::Clear

The method removes all items from a collection.

C++ Syntax

```
HRESULT Clear();
```

Visual Basic Syntax

```
BCEnumVerbs.Clear
```

Parameters

None

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCEnumVerbs" on page 74

IBCEnumVerbs::Count

The property returns the number of items in a collection.

C++ Syntax

```
HRESULT Count(
  long* plCount
);
```

Visual Basic Syntax

```
[ plCount = ] BCEnumVerbs.Count
```

Parameters

plCount	[out, retval] Returns a long that specifies the number of items in the collection.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Access

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCEnumVerbs" on page 74

IBCEnumVerbs::Item

The property returns a specific member of a collection either by position or by key.

C++ Syntax

```
HRESULT Item(  VARIANT vIndex,  IBCVerb** ppVerb);
```

Visual Basic Syntax

```
[ Set ppVerb = ] BCEnumVerbs.Item (vIndex)
```

Parameters

vIndex	[in] A numeric expression that specifies the position of a member of the collection. Index must be a number from 1 to the value of the collection's Count property.
ppVerb	[out, retval] Returns a BCVerb object that specifies the requested item.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Access

Read Only

Remarks

If the value provided as index does not match any existing member of the collection, an error occurs. Item is the default property for a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCEnumVerbs" on page 74

"IBCVerb" on page 160

IBCEnumVerbs::Remove

The method removes a specific item from a collection.

C++ Syntax

```
HRESULT Remove(
  VARIANT vIndex
);
```

Visual Basic Syntax

```
BCEnumVerbs.Remove (vIndex)
```

Parameters

vIndex	[in] A numeric expression specifying the index in the collection of the object to be removed.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCEnumVerbs" on page 74

"IBCObject" on page 80

IBCObject

The BCObject object represents an Explorer Core object provided by that plug-in.

When to Implement

The BCObject object is implemented by a plug-in.

Methods

IBCObject Methods	Description
IBCObject::EnumChildObjects	The method enumerates all child objects for this Explorer Core object.
IBCObject::GetProperty	The method returns the value of an object's property.
IBCObject::EnumContentProperties	The method returns a collection of custom properties that are the children of this object support.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

IBCObject::EnumChildObjects

The method enumerates all child objects for this Explorer Core object.

C++ Syntax

```
HRESULT EnumChildObjects(  IBCCoreCtx* pCoreCtx,  IBCEnumObjects**
ppEnum);
```

Visual Basic Syntax

```
[ Set ppEnum  = ] BCObject.EnumChildObjects (pCoreCtx)
```

Parameters

ppEnum	[out, retval] Returns a BCEnumObjects object that enumerates the child objects.
pCoreCtx	[in] A BCCoreCtx object that specifies enumeration parameters as follows:

Key	Value
SID	A string that uniquely identifies the session.

Key	Value
UIContext	A string that specifies the operation context that is to be used. If it is absent or an object does not understand the passed value, the default context is used.
Columns	A string with the identifiers of properties that will be shown in the Detail Mode. The identifiers are separated by a semicolon. Plug-ins can use the list in composing a request that will retrieve all the columns at once to increase performance.
OrderBy	A string that identifies the property the list is to be ordered by. It has the following format: ColumnID1[:SortOrder1][;ColumnID2[:SortOrder2]...] Where ColumnID1, ColumnID2 are column identifiers, SortOrder1, SortOrder2 are "A" (ascending sort order) or "D" (descending). The default value is "A".
Refresh	A short integer that requesting refresh. 0 - no refresh need 2 - need refresh

Key	Value
NotifyInterval	A string that specifies how often notification should be send. It has the following format: `ObjectLimit1:NumberOfObjects1;[ObjectLimit2]:NumberOfObjects2…` where: • ObjectLimit1,ObjectLimit2, … Apply this rule when number of read objects less or equal than this value • NumberOfObjects1,NumberOfObjects2, … Number of objects for sending notification. **Example:** "5:1;20:4;100:10;:20": It will be treated as: • notify after reading each object for first five objects • notify after reading each 4 objects when count of readed objects less or equal 20 • notify after each 10 objects when count of readed objects less or equal 100 • notify after each 20 objects for all remained objects

Key	Value
NotifyDelay	A string that specifies delay between notification. It has the following format: `NumberOfObject1:Delay1;[NumberOfObjects2]:Delay2d` where: • NumberOfObject1, NumberOfObject2 Apply this rule when number of read objects less or equal than this value • Delay1, Delay2 Delay in milliseconds between notifications. **Example:** "5:0;20:100;:500" • no delay when sending notifications for first five objects • delay 100 msec between notification when count of readed objects less or equal 100 • delay 500 msec between notifications for all remained objects

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCObject" on page 80

"IBCCoreCtx" on page 46

"IBCEnumObjects" on page 57

IBCObject::EnumContentProperties

The method returns a collection of custom properties that are the children of this object support.

C++ Syntax

```
HRESULT EnumContentProperties(  BSTR bstrSID,  BSTR bstrUIContext,
IBCEnumProperties** ppEnum);
```

Visual Basic Syntax

```
[ Set ppEnum  = ] BCObject.EnumContentProperties (bstrSID,
bstrUIContext)
```

Parameters

bstrSID	[in] A string that uniquely identifies the session object.
bstrUIContext	[in] A string that specifies the operation context that is to be used.
ppEnum	[out, retval] Returns a BCEnumProperties object that specifies the collection of custom properties objects or Nothing if the object's Container property is False.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

The retrieved collection doesn't include the standard properties all objects should support such as DisplayName (see IBCObject::GetProperty).

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCObject" on page 80

"IBCEnumProperties" on page 68

IBCObject::GetProperty

The method returns the value of an object's property.

C++ Syntax

```
HRESULT GetProperty(  BSTR bstrSID,  BSTR bstrUIContext,  BSTR
bstrPropertyID,  VARIANT* pvValue);
```

Visual Basic Syntax

```
[ pvValue  = ] BCObject.GetProperty (bstrSID, bstrUIContext,
bstrPropertyID)
```

Parameters

bstrSID	[in] A string that uniquely identifies the session object.
bstrUIContext	[in] A string that specifies the operation context that is to be used.
pvValue	[out, retval] Returns a variant that specifies the property value. If the specified property is not supported, the return value is Empty.
bstrPropertyID	[in] A string that specifies the property's name. It may be any of the properties in the EnumContentProperties collection or one of the following standard properties:

Name	Type	Description
Class	String	(Mandatory) Object's class. It uniquely identifies all objects of the given type.

Name	Type	Description
Container	Boolean	(Optional) Indicates whether or not the object can have children. Unlike HasNoChildren this property specifies the potential to contain sub-objects. True: Object can have children. False: Object cannot have children. The default value is False.
Description	String	(Optional) Object's description.
DisplayName	String	(Mandatory) Object name to display. It's a mandatory property.
HasNoChildren	Boolean	(Mandatory) Object name to display. It's a mandatory property. True: Object has children. False: Object does not have children. The default value is False.
ID	String	(Mandatory) Object's identifier.
Icon	Long	(Optional) Small icon handle. If not supported, a standard icon is used when displaying the object.
IconID	String	(Optional) Small icon identifier. The identifier format is defined by the plug-in. If not supported, a standard icon is used when displaying the object.
LargeIcon	Long	(Optional) Large icon handle. If not supported, a standard icon is used when displaying the object.
LargeIconID	String	(Optional) Large icon identifier. The identifier format is defined by the plug-in. If not supported, a standard icon is used when displaying the object.
NoChildrenText	String	(Optional) Text to be displayed in the right panel when the object has no children.
ShortName	String	(Optional) Object's short name having not more than 10 characters. If not supported, the object's DisplayName is truncated.

Name	Type	Description
SlowRetrieve	Boolean	(Optional) Indicates whether or not retrieving the object's children is time-consuming. True: Retrieving is time-consuming. The user should call the appropriate request explicitly rather than by highlighting the object in a tree. False: Retrieving is not time-consuming. The default value is False.
SlowRetrieveText	String	(Optional) Prompt text to be displayed in the right panel to retrieve the object's children.
Style	String	(Optional) Styles which the Explorer Core proposes to apply when displaying the object. There are two types of styles: standard and defined by plug-ins. Here are standard styles (browsers can (it's strongly recommended) take into account these styles when displaying objects): Important: Important object demanding user's attention. NotAvailable: Object that is not available at the moment. Overdue: Some overdue object (that is a variety of Important). Plain: Standard way of displaying objects. Unknown: This style is used when it is impossible to display the object with the style it demands. For example, if the object's font is not installed.
Transient	Boolean	(Optional) Indicates whether or not the object is persistent. True: Object is not persistent. False: Object is persistent. The default value is False. If an object is not persistent, it cannot be (or it's undesirable) retrieved by the ObjectByID method on the plug-in or session.
TransientWithCheck	Boolean	(Optional) The same as the Transient property but UI is allowed during this query (a user may be asked and the object may be saved and became persistent.

Name	Type	Description
FullPathID	String	(Optional) Object full path identifiers. This string should consist from Object IDs beginning with root object in hierarchy to current object. '/' as delimiter for Object ID should be used.
DefaultFullPathID	String	(Optional) Object default full path identifiers.
FileExtension	String	(Optional) File extension for item. This property is used by Microsoft Outlook DM Extension to display appropriate icon. Returned extension should be registered in the system.
CanCut	Boolean	(Optional) Indicates whether or not the object can be cut from current parent object. True: Object can be cut. False: Object cannot be cut. The default value is False.
CanDelete	Boolean	(Optional) Indicates whether or not the object can be deleted. True: Object can be deleted. False: Object cannot be deleted. The default value is False.
CanRename	Boolean	(Optional) Indicates whether or not the object can be renamed. True: Object can be renamed. False: Object cannot be renamed. The default value is False.
CanExportFile	Boolean	(Optional) Indicates whether or not the object can be exported. True: Object can be exported as a file. False: Object cannot be exported as a file. The default value is False.
ExportFileName	String	(Optional) Filename that should be used when exporting an object.
ExportFileContents	IUnknown	(Optional) IStream object that contains file contents for an object.

Name	Type	Description
CanUseEmailIntegration	Boolean	Indicates whether the e-mail integration can be used when message is dropped into this object in Microsoft Outlook DM Extension.

Return values

S_OK	Successful operation description.
S_FALSE	A property is not supported or the operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCObject" on page 80

IBCObject2

This interface extends BCObject BCObject functionality.

Methods

Methods	Description
IBCObject2::SetProperty	The method sets property for this Explorer Core object.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: import "DECore.idl"

See Also

"IBCObject" on page 80

IBCObject2::SetProperty

The method sets a property for the Explorer Core object.

C++ Syntax

```
HRESULT SetProperty(
  BSTR bstrSID,
  BSTR bstrUIContext,
  BSTR bstrPropertyID,
  VARIANT vValue
);
```

Visual Basic Syntax

```
BCObject2.SetProperty (bstrSID, bstrUIContext, bstrPropertyID,
vValue)
```

Parameters

bstrSID	[in] A string that uniquely identifies the session object.
bstrUIContext	[in] A string that specifies the operation context that is to be used.
bstrPropertyID	[in] A string that specifies the property's name.
vValue	[in] A property value.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "DECore.idl"

See Also

"IBCObject2" on page 90

IBCObjectArray

The BCObjectArray object represents an array of Explorer Core objects that can dynamically shrink and grow as necessary. Once you have inserted an object into the array, you can efficiently retrieve or remove the object from the array using the index to access it. Array indexes always start at position 0.

Methods

IBCObjectArray Methods	Description
IBCObjectArray::Size	The method returns the number of elements in this array.
IBCObjectArray::Add	The method adds an object to the end of the array.
IBCObjectArray::Remove	The method removes a given object from the array.
IBCObjectArray::InsertAt	The method inserts an object at a specified index into the array.
IBCObjectArray::RemoveAt	The method inserts an object at a specified index into the array.
IBCObjectArray::GetAt	The method returns the object at a given index.
IBCObjectArray::RemoveAll	The method returns the object at a given index.
IBCObjectArray::DoVerb	The method executes a specified command for all the objects in the array.
IBCObjectArray::EnumVerbs	The method enumerates all commands available to the objects in the array.
IBCObjectArray::SetAsDefault	The method sets or clears the objects in the array as default.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

IBCObjectArray::Add

The method adds an object to the end of the array.

C++ Syntax

```
HRESULT Add(
  IUnknown* pObject
);
```

Visual Basic Syntax

```
BCObjectArray.Add (pObject)
```

Parameters

pObject	[in] An object to be added to the array.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

The method adds a new element to the end of the array, growing the array by 1.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCObjectArray" on page 91

IBCObjectArray::DoVerb

The method executes a specified command for all the objects in the array.

C++ Syntax

```
HRESULT DoVerb(
  BSTR bstrSID,
  BSTR bstrUIContext,
  BSTR bstrCmdID,
  long grfKeyState,
  IBCObject* pParent
);
```

Visual Basic Syntax

```
BCObjectArray.DoVerb (bstrSID, bstrUIContext, bstrCmdID,
grfKeyState, pParent)
```

Parameters

bstrSID	[in] A string that uniquely identifies the session object.
bstrUIContext	[in] A string that specifies the operation context that is to be used.
bstrCmdID	[in] A string that uniquely identifies the command to execute. For possible values see the IBCContextMenuSupport::DoVerb method.
grfKeyState	[in] A long that specifies the control keys being pressed. For possible values see KeyFlags Enumeration values.
pParent	[in] A BCObject object that specifies the parent for all the objects in the array, e.g. the root object of a list where the array entries were selected. The parameter may be Nothing if there is no such object.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCObjectArray" on page 91

"IBCObject" on page 80

"IBCContextMenuSupport" on page 38

"KeyFlags Enumeration" on page 177

IBCObjectArray::EnumVerbs

The method enumerates all commands available to the objects in the array.

C++ Syntax

```
HRESULT EnumVerbs(
  BSTR bstrSID,
  BSTR bstrUIContext,
  IBCObject* pParent
  IBCEnumVerbs** ppEnumVerbs
);
```

Visual Basic Syntax

```
[ Set ppEnumVerbs  = ] BCObjectArray.EnumVerbs (bstrSID,
bstrUIContext, pParent)
```

Parameters

bstrSID	[in] A string that uniquely identifies the session object.
bstrUIContext	[in] A string that specifies the operation context that is to be used.
pParent	[in] A BCObject object that specifies the parent for all the objects in the array, e.g. the root object of a list where the array entries were selected. The parameter may be Nothing if there is no such object.
ppEnumVerbs	[out, retval] Returns a BCEnumVerbs object that enumerates the requested commands.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCObjectArray" on page 91

"IBCObject" on page 80

"IBCEnumVerbs" on page 74

IBCObjectArray::GetAt

The method returns the object at a given index.

C++ Syntax

```
HRESULT GetAt(
  long lPos,
  IUnknown** ppObject
);
```

Visual Basic Syntax

```
[ Set ppObject  = ] BCObjectArray.GetAt (lPos)
```

Parameters

lPos	[in] A long index at which the element is to be retrieved. The value must be greater than or equal to 0 and less than or equal to the value returned by Size.
ppObject	[out, retval] Returns the object at the specified index.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCObjectArray" on page 91

IBCObjectArray::InsertAt

The method removes a given object from the array.

C++ Syntax

```
HRESULT InsertAt(
  long lPos,
  IUnknown* pObject
);
```

Visual Basic Syntax

```
BCObjectArray.InsertAt (lPos, pObject)
```

Parameters

lPos	[in] A long index at which the new element is to be inserted. The value must be greater than or equal to 0 and less than or equal to the value returned by Size.
pObject	[in] An object to be placed in the array.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

The method inserts an object at a specified index in an array. In the process, it shifts up (by incrementing the index) the existing element at this index, and it shifts up all the elements above it.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCObjectArray" on page 91

IBCObjectArray::Remove

The method removes a given object from the array.

C++ Syntax

```
HRESULT Remove(
  IUnknown* pObject
);
```

Visual Basic Syntax

```
BCObjectArray.Remove (pObject)
```

Parameters

pObject	[in] An object to be removed from the array.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

The method looks up the supplied object and then, if the object is found, removes it.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCObjectArray" on page 91

IBCObjectArray::RemoveAll

The method removes all the elements from this array.

C++ Syntax

```
HRESULT RemoveAll();
```

Visual Basic Syntax

```
BCObjectArray.RemoveAll
```

Parameters

None

Remarks

The method looks up the supplied object and then, if the object is found, removes it.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCObjectArray" on page 91

IBCObjectArray::RemoveAt

The method removes an object at a specified index from the array.

C++ Syntax

```
HRESULT RemoveAt(
  long lPos
);
```

Visual Basic Syntax

```
BCObjectArray.RemoveAt (lPos)
```

Parameters

lPos	[in] A long index at which the element is to be removed. The value must be greater than or equal to 0 and less than or equal to the value returned by Size.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

The method removes an element at a specified index in an array. In the process, it shifts down all the elements above the removed element.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCObjectArray" on page 91

IBCObjectArray::SetAsDefault

The method sets or clears the objects in the array as defaults.

C++ Syntax

```
HRESULT SetAsDefault(
  BSTR bstrSID,
  VARIANT_BOOL bDefault,
  IBCObject* pParent
);
```

Visual Basic Syntax

```
BCObjectArray.SetAsDefault (bstrSID, bDefault, pParent)
```

Parameters

bstrSID	[in] A string that uniquely identifies the session object.
bDefault	[in] A Boolean value that indicates whether the object array should be set as default or unset.
pParent	[in] A BCObject object that specifies the parent for all the objects in the array, e.g. the root object of a list where the array entries were selected. The parameter may be Nothing if there is no such object.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

The method is called when there needs to set an array as default indicating that its objects were highlighted or, on the contrary, the user

cleared selection. UI should respond to this action by changing the desktop menu, toolbar, and other elements.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCObjectArray" on page 91

"IBCObject" on page 80

IBCObjectArray::Size

The method sets a property for the Explorer Core object.

C++ Syntax

```
HRESULT Size(
  long* pcelt
);
```

Visual Basic Syntax

```
[ pcelt  = ] BCObjectArray.Size
```

Parameters

pcelt	[out, retval] Returns a long that specifies the array size.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCObjectArray" on page 91

IBCObjectCaps

The BCObjectCaps object indicates what functionality an Explorer Core object supports.

When to Implement

The BCObjectCaps object is implemented by a plug-in.

Methods

IBCObjectCaps Methods	Description
IBCObjectCaps::GetDetailColumns	The method returns an array of columns that should be used to display the object contents in report view.
IBCObjectCaps::GetStructuredViewDefinition	The method returns an array of columns that should be used to display the object contents in structured view.
IBCObjectCaps::SupportedViewModes	The method indicates what views the object supports.
IBCObjectCaps::ContextParamSupported	The method returns whether or not the object supports a specified context key.
IBCObjectCaps::UIContextSupported	The method returns whether or not the object supports a specified operation context.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

IBCObjectCaps::ContextParamSupported

The method returns whether or not the object supports a specified context key.

C++ Syntax

```
HRESULT ContextParamSupported(  BSTR bstrParamName  VARIANT_BOOL*
pbSupported);
```

Visual Basic Syntax

```
[ pbSupported  = ] BCObjectCaps.ContextParamSupported
(bstrParamName)
```

Parameters

bstrParamName	[in] A string that specifies the key to be checked.
pbSupported	[out, retval] Returns a Boolean value that specifies whether or not the key is supported. True: Key is supported. False: Key is not supported.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

If a context key is supported, it can be specified in the pCoreCtx parameter of the EnumChildObjects method on BCObject.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCObjectCaps" on page 102

"IBCObject" on page 80

IBCObjectCaps::GetDetailColumns

The method returns an array of columns that should be used to display the object contents in the report view.

C++ Syntax

```
HRESULT GetDetailColumns(
  IBCPropertyArray** ppColumns
);
```

Visual Basic Syntax

```
[ Set ppColumns  = ] BCObjectCaps.GetDetailColumns
```

Parameters

ppColumns	[out, retval] Returns a BCPropertyArray object that specifies the array or Nothing if the report view is not supported.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCObjectCaps" on page 102

"IBCPropertyArray" on page 127

IBCObjectCaps::GetStructuredViewDefinition

The method returns an array of columns that should be used to display the object contents in the structured view.

C++ Syntax

```
HRESULT GetStructuredViewDefinition(
  IBCPropertyArray** ppColumns
);
```

Visual Basic Syntax

```
[ Set ppColumns  = ] BCObjectCaps.GetStructuredViewDefinition
```

Parameters

ppColumns	[out, retval] Returns a BCPropertyArray object that specifies the array or Nothing if the structured view is not supported.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCObjectCaps" on page 102

"IBCPropertyArray" on page 127

IBCObjectCaps::SupportedViewModes

The method indicates the views the object supports.

C++ Syntax

```
HRESULT SupportedViewModes(
  long* plViewModesSupported
);
```

Visual Basic Syntax

```
[ plViewModesSupported  = ] BCObjectCaps.SupportedViewModes
```

Parameters

plViewModesSupported	[out, retval] Returns a long that specifies the supported views. It can be zero, one or more of the ViewMode enumeration values.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCObjectCaps" on page 102

"ViewMode Enumeration" on page 183

IBCObjectCaps::UIContextSupported

The method returns whether or not the object supports a specified operation context.

C++ Syntax

```
HRESULT UIContextSupported(
  BSTR bstrContextName
  VARIANT_BOOL* pbSupported
);
```

Visual Basic Syntax

```
[ pbSupported  = ] BCObjectCaps.UIContextSupported
(bstrContextName)
```

Parameters

bstrContextName	[in] A string that specifies the operation context that is to be checked.
pbSupported	[out, retval] Returns a Boolean value that specifies whether or not the context is supported. True: Context is supported. False: Context is not supported.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

If an object does not understand a context passed as an argument to the method, it will use some reasonable defaults for its properties and behavior.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCObjectCaps" on page 102

IBCObjectCaps2

The IBCObjectCaps2 interface extends BCObjectCaps object functionality.

Methods

IBCObjectCaps2 Methods	Description
IBCObjectCaps2::GetViewMode	The method returns current view mode for object.
IBCObjectCaps2::SetViewMode	The method sets current view mode for object.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: import "DECore.idl"

See Also

"IBCObjectCaps" on page 102

IBCObjectCaps2::GetViewMode

The method returns the current view mode for object.

C++ Syntax

```
HRESULT GetViewMode(
  long* plViewMode
);
```

Visual Basic Syntax

```
[ plViewMode  = ] BCObjectCaps2.GetViewMode
```

Parameters

plViewMode	[out, retval] Returns the current view mode. For possible values see the IBCObjectCaps::SupportedViewModes method.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "DECore.idl"

See Also

"IBCObjectCaps2" on page 107

"IBCObjectCaps" on page 102

IBCObjectCaps2::SetViewMode

The method sets the current view mode for object.

C++ Syntax

```
HRESULT SetViewMode(
  long lViewMode
);
```

Visual Basic Syntax

```
BCObjectCaps2.SetViewMode (lViewMode)
```

Parameters

plViewMode	[in] The current view mode. For possible values see IBCObjectCaps::SupportedViewModes method.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "DECore.idl"

See Also

"IBCObjectCaps2" on page 107

"IBCObjectChanged" on page 110

IBCObjectChanged

The IBCObjectChanged object is used to receive notification from plug-in object.

When to Implement

The BCObjectChanged object is implemented by a browser.

Methods

IBCObjectChanged Methods	Description
IBCObjectChanged::OnDataChange	The method is called by plug-in when object state or object content has been changed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

IBCObjectChanged::OnDataChange

The method is called by plug-in when an object state or object content has been changed.

C++ Syntax

```
HRESULT OnDataChange(
  IBCObject* pObject,
  short sReason,
  IDispatch* pUnk
);
```

Visual Basic Syntax

```
BCObjectChanged.OnDataChange (pObject, sReason, pUnk)
```

Parameters

pObject	[in] BCObject that has been changed.
sReason	[in] Notification reason. Can be one of the CoreEvents enumeration values.
pUnk	[in] Additional parameter that indicates object(s) that has been changed. It can be NULL if notification related to pObject. This parameter can be BCObject or BCObjectArray. For bcevDelete notification it indicates object(s) that have been deleted from pObject. For bcevAdd notification it indicates object(s) that have been added to pObject.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCObjectChanged" on page 110

"IBCObject" on page 80

"IBCObjectArray" on page 91

"CoreEvents Enumeration" on page 175

IBCPlugin

The BCPlugin object represents an Explorer Core plug-in, a module that extends Explorer Core functionality.

One of the plug-in examples is the Document Management plug-in that provides document management capabilities and turns the

Explorer Core technology into a complete shrink-wrapped product, DM Extensions.

Methods

IBCPlugin Methods	Description
IBCPlugin::Init	The method initializes the plug-in.
IBCPlugin::GetPluginClass	The method returns the plug-in class.
IBCPlugin::GetPluginInfo	The method returns plug-in attributes as an associative map.
IBCPlugin::Logon	The method logs the user into the plug-in.
IBCPlugin::Logoff	The method logs off the plug-in.
IBCPlugin::EnumChildObjects	The method enumerates all root objects provided by the plug-in for a given session.
IBCPlugin::ObjectByID	The method returns a specific plug-in object by its identifier.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

IBCPlugin::GetPluginClass

The method returns the plug-in class.

C++ Syntax

```
HRESULT GetPluginClass(
  BSTR* pbstrClass
);
```

Visual Basic Syntax

```
[ pbstrClass  = ] BCPlugin.GetPluginClass
```

Parameters

pbstrClass	[out, retval] A string that uniquely identifies the plug-in.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

The plug-in class must be unique.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCPlugin" on page 111

IBCPlugin::Logoff

The method logs off the plug-in.

C++ Syntax

```
HRESULT Logoff(
  BSTR bstrSID
);
```

Visual Basic Syntax

```
BCPlugin.Logoff (bstrSID)
```

Parameters

bstrSID	[in] A string that uniquely identifies the session.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

The Explorer Core uses this method when it needs to unload a plug-in from the supplied session. The plug-in can unload all the objects related to the session it has created to free the memory.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header:Header: #import "DECore.idl"

See Also

"IBCPlugin" on page 111

IBCPlugin::Logon

The method logs the user into the plug-in.

C++ Syntax

```
HRESULT Logon(
  BSTR bstrSID
);
```

Visual Basic Syntax

```
BCPlugin.Logon (bstrSID)
```

Parameters

bstrSID	[in] A string that uniquely identifies the session.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

The Explorer Core uses this method for explicit logging into plug-ins.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCPlugin" on page 111

IBCPlugin::Init

The method initializes the plug-in.

C++ Syntax

```
HRESULT Init(
  IBCBrowserCore* pCore
);
```

Visual Basic Syntax

```
BCPlugin.Init (pCore)
```

Parameters

pCore	[in] A BCBrowserCore object identifying the Explorer Core instance.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

It is not recommended for Plug-ins to store a reference to the IBCBrowserCore to avoid cross references. Instead of this Plug-ins should implement the IConnectionPointContainer interface and IBCBrowserCore connection point entry. During loading plug-ins the BCBrowserCore object will try to connect to this connection point.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCPlugin" on page 111

"IBCBrowserCore" on page 34

IBCPlugin::EnumChildObjects

The method enumerates all root objects provided by the plug-in for a given session.

C++ Syntax

```
HRESULT EnumChildObjects(
  BSTR bstrSID,
  IBCCoreCtx* pCoreCtx,
  IBCEnumObjects** ppEnum
);
```

Visual Basic Syntax

```
[ Set ppEnum  = ] BCPlugin.EnumChildObjects (bstrSID, pCoreCtx)
```

Parameters

bstrSID	[in] A string that uniquely identifies the session.
pCoreCtx	[in] A BCCoreCtx object that specifies enumeration parameters. For possible values see the IBCObject::EnumChildObjects method.
ppEnum	[out, retval] Returns a BCEnumObjects object that enumerates the root plug-in objects.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCPlugin" on page 111

"IBCCoreCtx" on page 46

"IBCEnumObjects" on page 57

"IBCObject" on page 80

IBCPlugin::GetPluginInfo

The method returns plug-in attributes as an associative map.

C++ Syntax

```
HRESULT GetPluginInfo(
  IBCCoreCtx** ppCoreCtx
);
```

Visual Basic Syntax

```
[ Set ppCoreCtx  = ] BCPlugin.GetPluginInfo
```

Parameters

ppCoreCtx	[out, retval] Returns a BCCoreCtx object that contains the requested plug-in attributes as follows:

Key	Value
Description	A string describing the plug-in.
VersionsSupported	A Boolean that indicates whether or not the plug-in supports versions (context versions). True: Plug-in supports versions. False: Plug-in does not support versions.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCPlugin" on page 111

"IBCCoreCtx" on page 46

IBCPlugin::ObjectByID

The method returns a specific plug-in object by its identifier.

C++ Syntax

```
HRESULT ObjectByID(
  BSTR bstrSID,
  BSTR bstrFullID,
  IBCObject** ppObject
);
```

Visual Basic Syntax

```
[ Set ppObject  = ] BCPlugin.ObjectByID (bstrSID, bstrFullID)
```

Parameters

bstrSID	[in] A string that uniquely identifies the session.
bstrFullID	[in] A string that uniquely identifies the required object.
ppObject	[out, retval] Returns the BCObject object identified by bstrFullID.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCPlugin" on page 111

"IBCObject" on page 80

IBCProperty

The BCProperty object represents a property of an Explorer Core object.

When to Implement

The BCProperty object is implemented by plug-ins.

Methods

IBCProperty Methods	Description
IBCProperty::GetDefault	The method returns the default value for a specified attribute.

Properties

IBCProperty Properties	Description
IBCProperty::ID	The property returns the property identifier.
IBCProperty::DisplayName	The property returns the name of the property to display in a list.
IBCProperty::Description	The property returns the object property's description.
IBCProperty::Type	The property returns the data type of an object's property.
IBCProperty::Flags	The property returns the object property's attributes.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

IBCProperty::Description

The property returns object property's description.

C++ Syntax

```
HRESULT Description(
  BSTR* pbstrDescription
);
```

Visual Basic Syntax

```
[ pbstrDescription = ] BCProperty.Description
```

Parameters

pbstrDescription	[out, retval] Returns a string that specifies the full description of the property.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Access

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCProperty" on page 119

IBCProperty::DisplayName

The property returns the name of the property to display in the list.

C++ Syntax

```
HRESULT DisplayName(
  BSTR* pbstrDisplayName
);
```

Visual Basic Syntax

```
[ pbstrDisplayName = ] BCProperty.DisplayName
```

Parameters

pbstrDisplayName	[out, retval] Returns a string that specifies the property display name.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Access

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCProperty" on page 119

IBCProperty::Flags

The property returns object property's attributes.

C++ Syntax

```
HRESULT Flags(
  long* plFlags
);
```

Visual Basic Syntax

```
[ plFlags = ] BCProperty.Flags
```

Parameters

plFlags	[out, retval] Returns a long that specifies property's attributes. It can be zero, one or more of the PropertyAttributes enumeration values. These flags specify when property's column should be displayed. If the justify flags are not specified, the default justification will be used: numeric fields will be right justified, all others - left justified.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Access

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCProperty" on page 119

"PropertyAttributes Enumeration" on page 180

IBCProperty::GetDefault

The method returns the default value for a specified attribute.

C++ Syntax

```
HRESULT GetDefault(
  BSTR bstrAttrName,
  VARIANT* pvDefault
);
```

Visual Basic Syntax

```
[ pvDefault  = ] BCProperty.GetDefault (bstrAttrName)
```

Parameters

pvDefault	[out, retval] Returns a variant that specifies the default value of the requested attribute. If the specified attribute is not supported, the return value is Empty.
bstrAttrName	[in] A string that indicates the attribute whose value is being requested as follows:

Value	Type	Description
Width	Long	The width of property's column for report view in pixels.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCProperty" on page 119

IBCProperty::ID

The property returns the property identifier.

C++ Syntax

```
HRESULT ID(
  BSTR* pbstrID
);
```

Visual Basic Syntax

```
[ pbstrID = ] BCProperty.ID
```

Parameters

pbstrID	[out, retval] Returns a string that uniquely identifies the property.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Access

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCProperty" on page 119

IBCProperty::Type

The property returns the data type of object's property.

C++ Syntax

```
HRESULT Type(
  long* plType
);
```

Visual Basic Syntax

```
[ plType = ] BCProperty.Type
```

Parameters

plType	[out, retval] Returns a long that specifies property's type. It can be one of the following values:

Value	Description
VT_I2(2)	Indicates that the object is a short integer.
VT_I4(3)	Indicates that the object is a long integer.
VT_R4(4)	Indicates that the object is of single-precision floating-point type.
VT_R8(5)	Indicates that the object is a short integer.
VT_CY(6)	Indicates that the object is of currency type.
VT_I2(2)	Indicates that the object is a short integer.
VT_DATE(7)	Indicates that the object is a date.
VT_BSTR(8)	Indicates that the object is a string.
VT_BOOL(11)	Indicates that the object is a Boolean value.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Access

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCProperty" on page 119

IBCPropertyArray

The BCPropertyArray object is used to control visual presentation of Explorer Core object's properties in report view.

The array can dynamically shrink and grow as necessary. Once you have inserted an object into the array, you can efficiently retrieve or remove the object from the array using the index to access it. Array indexes always start at position 0.

When to Implement

The BCPropertyArray object is implemented by plug-ins.

Methods

IBCPropertyArray Methods	Description
IBCPropertyArray::Add	Adds an item to an array.Inserts an item into an array at a specified position.
IBCPropertyArray::InsertAt	Inserts an item into an array at a specified position.
IBCPropertyArray::RemoveAt	Removes an item at a specified position from an array.
IBCPropertyArray::RemoveAll	Removes all items from an array.
IBCPropertyArray::GetAt	Returns a specified property of an array by position.
IBCPropertyArray::SetAt	Sets a specified property of an array by position.
IBCPropertyArray::IsEqual	Compares another array with the current array.
IBCPropertyArray::GetColumnProperty	Returns the value of a specified property in an array.
IBCPropertyArray::SetColumnProperty	Sets the value of a specified property in an array.
IBCPropertyArray::RestoreDefaults	Restores an array to its default state.
IBCPropertyArray::Save	Saves the state of properties in an array.

Properties

IBCPropertyArray Properties	Description
IBCPropertyArray::Count	The number of items in the array.
IBCPropertyArray::Flags	The property returns operations supported by the array.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

IBCPropertyArray::Add

The method adds a property to the end of the array.

C++ Syntax

```
HRESULT Add(
  IBCProperty* pProp
);
```

Visual Basic Syntax

```
BCPropertyArray.Add (pProp)
```

Parameters

pProp	[in] A BCProperty object to be added to the array.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

The method adds a new element to the end of the array, growing the array by 1.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCPropertyArray" on page 127

"IBCProperty" on page 119

IBCPropertyArray::Count

The property returns the number of items in the array.

C++ Syntax

```
HRESULT Count(
```

```
  long* pcelt
);
```

Visual Basic Syntax

```
[ pcelt = ] BCPropertyArray.Count
```

Parameters

pcelt	[out, retval] Returns a long that specifies the size of the array. Since indexes are zero-based, the size is 1 greater than the largest index.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Access

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCPropertyArray" on page 127

IBCPropertyArray::Flags

The property returns operations supported by the array.

C++ Syntax

```
HRESULT Flags(
  long* plFlags
);
```

Visual Basic Syntax

```
[ plFlags = ] BCPropertyArray.Flags
```

Parameters

plFlags	[out, retval] Returns a long that specifies the supported operations. It can be zero, one or more of the PropertyArrayFlags enumeration values.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Access

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCPropertyArray" on page 127

"PropertyArrayFlags Enumeration" on page 179

IBCPropertyArray::GetAt

The method returns the property at a given index.

C++ Syntax

```
HRESULT GetAt(  long  lPos,   IBCProperty** pProp);
```

Visual Basic Syntax

```
 [ Set pProp  = ] BCPropertyArray.GetAt (lPos)
```

Parameters

lPos	A long index at which the element is to be retrieved. The value must be greater than or equal to 0 and less than the value returned by Count.
pProp	[out, retval] Returns the BCProperty object at the specified index.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCPropertyArray" on page 127

"IBCProperty" on page 119

IBCPropertyArray::GetColumnProperty

The method returns a specified attribute of the property at a given index.

C++ Syntax

```
 HRESULT GetColumnProperty(  long  lPos,   BSTR bstrName,
VARIANT* pvValue);
```

Visual Basic Syntax

```
 [ pvValue  = ] BCPropertyArray.GetColumnProperty (lPos, bstrName)
```

Parameters

pvValue	[out, retval] Returns a variant that specifies the attribute value. If the specified attribute is not supported, the return value is Empty.
lPos	A long index that specifies the element whose attribute is to be retrieved. The value must be greater than or equal to 0 and less than the value returned by Count.
bstrName	[in] A string that specifies the property's attribute. It may be one of the following values:

Value	Type	Description
SortOrder	String	Sort order. It may be an empty string (not sorted), "A" (ascending sort order), or "D" (descending one).
Width	Long	The width of the property's column for report view in pixels.

Browsers can set (using the SetColumnProperty method) and retrieve any named attribute.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Example

The following example displays the widths of columns that should be used to present the contents of the folder with number 1362 on the CRYSTAL library of the Document Management plug-in in report view.

```
Dim Session As New BCSession
Dim ObjCaps As BCObjectCaps
Dim Props As BCPropertyArray
Session.Logon Nothing
```

```
Set ObjCaps = Session.ObjectByID("DOCSOPEN!Pj\CRYSTAL\1362")
If Not ObjCaps Is Nothing Then
If ObjCaps.SupportedViewModes And VM_DETAIL Then
Set Props = ObjCaps.GetDetailColumns
For i = 1 To Props.Count
MsgBox "Width of " + Props.GetAt(i - 1).ID + _
": " + Str(Props.GetColumnProperty(i - 1, "Width"))
Next
End If
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCPropertyArray" on page 127

IBCPropertyArray::InsertAt

The method inserts a property at a specified index into the array.

C++ Syntax

```
HRESULT InsertAt(
  long  lPos,
  IBCProperty* pProp
);
```

Visual Basic Syntax

```
BCPropertyArray.InsertAt (lPos, pProp)
```

Parameters

lPos	[in] A long index at which the new element is to be inserted. The value must be greater than or equal to 0 and less than or equal to the value returned by Count.
pProp	[in] A BCProperty object to be placed in the array.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

The method inserts a property at a specified index in the array. In the process, it shifts up (by incrementing the index) the existing element at this index and all the elements above it.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCPropertyArray" on page 127

"IBCProperty" on page 119

IBCPropertyArray::IsEqual

The method compares a specified array with this one.

C++ Syntax

```
HRESULT IsEqual(
  IBCPropertyArray* pArray
  VARIANT_BOOL*  pbResult,
);
```

Visual Basic Syntax

```
[ pbResult  = ] BCPropertyArray.IsEqual (pArray)
```

Parameters

pArray	[in] A BCPropertyArray object to be compared.
pbResult	[out, retval] Returns a Boolean value that specifies whether or not the arrays are equal. True: Arrays are equal. False: Arrays are not equal.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCPropertyArray" on page 127

IBCPropertyArray::RemoveAll

The method removes all the elements from the array.

C++ Syntax

```
HRESULT RemoveAll();
```

Visual Basic Syntax

```
BCPropertyArray.RemoveAll
```

Parameters

None

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCPropertyArray" on page 127

IBCPropertyArray::RemoveAt

The method removes a property at a specified index from the array.

C++ Syntax

```
HRESULT RemoveAt(
  long  lPos
);
```

Visual Basic Syntax

```
BCPropertyArray.RemoveAt (lPos)
```

Parameters

lPos	[in] A long index at which the element is to be removed. The value must be greater than or equal to 0 and less than or equal to the value returned by Count.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

The method removes an element at a specified index in the array. In the process, it shifts down all the elements above the removed element.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCPropertyArray" on page 127

IBCPropertyArray::RestoreDefaults

The method restores the array to its previously saved state.

C++ Syntax

```
HRESULT RestoreDefaults();
```

Visual Basic Syntax

```
BCPropertyArray.RestoreDefaults
```

Parameters

None

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCPropertyArray" on page 127

IBCPropertyArray::Save

The method saves the array to restore it later.

C++ Syntax

```
HRESULT Save(
  short sFlags
);
```

Visual Basic Syntax

```
BCPropertyArray.Save (sFlags)
```

Parameters

sFlags	[in] An integer value that specifies how the array should be saved. The parameter can be zero or the PropertyArraySaveFlags enumeration value.

Return Values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCPropertyArray" on page 127

"PropertyArraySaveFlags Enumeration" on page 179

IBCPropertyArray::SetAt

The method sets the value for a given index without growing an array.

C++ Syntax

```
HRESULT SetAt(
  long  lPos,
  IBCProperty* pProp
);
```

Visual Basic Syntax

```
BCPropertyArray.SetAt (lPos, pProp)
```

Parameters

lPos	[in] A long index at which the element is to be replaced. The value must be greater than or equal to 0 and less than or equal to the value returned by Count.
pProp	[in] A BCProperty object to be stored at the specified position.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

The method sets the array element at the specified index. SetAt will not cause the array to grow. You must ensure that your index value represents a valid position in the array. If it is out of bounds, an error will occur.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCPropertyArray" on page 127

"IBCProperty" on page 119

IBCPropertyArray::SetColumnProperty

The method sets the value of a specified attribute of the property at a given index.

C++ Syntax

```
HRESULT SetColumnProperty(
  long   lPos,
  BSTR   bstrName,
  VARIANT vValue
);
```

Visual Basic Syntax

```
BCPropertyArray.SetColumnProperty (lPos, bstrName, vValue)
```

Parameters

lPos	[in] A long index that indicates the element whose attribute is to be set. The value must be greater than or equal to 0 and less than or equal to the value returned by Count.
bstrName	[in] A string that specifies the property's attribute. For possible values see IBCPropertyArray::GetColumnProperty.
vValue	[in] An expression that specifies the attribute value. The type of this parameter must conform to the attribute type.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCPropertyArray" on page 127

IBCRegistrar

This interface can be used to register programmable objects. Its methods RegisterClass and UnregisterClass core extensions should call in their DllRegisterServer and DllUnregisterServer functions, respectively.

The interface inherits its definition from IUnknown, so all implementations must also include three standard methods of `IUnknown`: AddRef, Release, and QueryInterface.

Methods

IBCRegistrar Methods	Description
IBCRegistrar::RegisterClass	Instructs the core to create its registry entries related to the core emulator associated to a specified programmable object's class. If this function fails, the state of the registry for those classes is indeterminate.
IBCRegistrar::UnregisterClass	Instructs the core to remove only those entries created through IBCRegistrar::RegisterClass.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

IBCRegistrar::RegisterClass

The method instructs the core to create its registry entries related to the core emulator associated with a specified programmable object's class. If this function fails, the state of the registry for these classes is indeterminate.

C++ Syntax

```
HRESULT RegisterClass(  BSTR  bstrClsid,  long lFlags);
```

Visual Basic Syntax

```
BCRegistrar.RegisterClass bstrClsid, lFlags
```

Parameters

bstrClsid	[in] Specifies CLSID or ProgID of the programmable object for which the core aggregate is to be registered.
lFlags	[in] Specifies flags used when creating the core object that emulates the specified object. The value may be any combination of the RegisterFlags enumeration.

Return values

S_OK	The registry entries were created successfully.
SELFREG_E_CLASS	The core was unable to complete the registration of the supplied object class.
E_OUTOFMEMORY	Ran out of memory.
E_UNEXPECTED	An internal error occurred.

Remarks

The method allows in-process core extensions to properly register their programmable objects. The extension should call it when processing the DllRegisterServer function just after adding required HKEY_CLASSES_ROOT registry entries for its object. When processing the request IBCRegistrar first checks whether or not the specified CLSID has been linked to a core emulator by looking at the ImplementedObjects registry key. If the link exists, it updates the registry entries for CLSID and the associated core emulator. Otherwise, IBCRegistrar creates a new CLSID for the core emulator, links it to the supplied programmable object's CLSID by adding all required registry entries.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCRegistrar" on page 141

"RegisterFlags Enumeration" on page 181

IBCRegistrar::UnregisterClass

Instructs the core to remove only the entries created through IBCRegistrar::RegisterClass.

C++ Syntax

```
HRESULT UnregisterClass(  BSTR  bstrClsid);
```

Visual Basic Syntax

```
BCRegistrar.UnregisterClass (bstrClsid)
```

Parameters

bstrClsid	[in] Specifies CLSID of the programmable object whose registry entries are to be removed.

Return values

S_OK	The registry entries were created successfully.
S_FALSE	Unregistration of object's known entries was successful, but other entries still exist for this object.
SELFREG_E_CLASS	The core was unable to remove the entries of the supplied object.
E_OUTOFMEMORY	Ran out of memory
E_UNEXPECTED	An internal error occurred.

Remarks

The method allows core in-process extensions to properly remove all registry entries related to their programmable objects. The extension should call it when processing the DllRegisterServer function before removing all entries it added itself when registering its object.

When processing this request IBCRegistrar finds out the registry entry under the ImplementedObjects key that links the specified object with a core emulator. Then it removes the entry and all the registry entries related to the corresponding core emulator's CLSID. If the TreatAs registry subkey under the programmable object's CLSID indicate the

same core emulator's CLSID, it deletes the subkey by calling CoTreatAsClass with CLSID_NULL.

The core should not disturb any entries that it did not create and that currently exist for its classes and the programmable object. Win32 registry functions specifically disallow the deletion of an entire populated tree in the registry. The server can attempt, as the last step, to remove the CLSID key, but if other entries still exist, the key will remain.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCRegistrar" on page 141

IBCRegistrarExt

BCRegistrarExt object register extension objects for DM Extensions.

When to Implement

Explorer Core implements this interface to register extension objects.

When to Use

This interface can be used to register extension objects.

Methods

IBCRegistrar Methods	Description
IBCRegistrarExt::RegisterExtObject	Instructs the core to create its registry entries related to the core emulator associated to a specified programmable object's class. If this function fails, the state of the registry for those classes is indeterminate.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: import "DECore.idl"

See Also

"IBCRegistrar" on page 141

IBCRegistrarExt::RegisterExtObject

Instructs the core to create its registry entries related to the core emulator associated to a specified programmable object's class. If this function fails, the state of the registry for those classes is indeterminate.

C++ Syntax

```
HRESULT RegisterExtObject(  long nCommand,
  BSTR strType,
  BSTR strName,
  BSTR strCLSIDOrProgID,
  IBCCoreCtx* pParams );
```

Visual Basic Syntax

```
BCRegistrarExt.RegisterExtObject nCommand, strType, strName,
strCLSIDOrProgID [, pParams ]
```

Parameters

nCommand	[in] Specifies operation. For possible values see RegistrarCommand Enumeration.
strType	[in] Specifies object type. See remarks below for possible object types.
strName	[in] Specifies object name. This name is used as a key to reference custom object.
strCLSIDorProgID	[in] Specifies CLSID or ProgID of the programmable object.
pParams	[in, optional] BCCoreCtx object that contains additional parameters that may be used with custom object.

Return Values

This method supports the standard return value E_OUTOFMEMORY and E_UNEXPECTED as well as the following:

S_OK	The registry entries were created successfully.
E_INVALIDARG	Unknown object type is specified.

Remarks

The method allow to register/unregister/disable/enable custom objects that extend DM Extensions functionality. Below the list of supported object types and additional parameterss that are supported for the custom object.

Type	Supported parameters		Comment
	Name	Value	
AppIntegrationScheme	InsertAt	Insert position ('A'-'Z')	Application Integration Scheme
Plugin	InsertAt	Insert position ('A'-'Z')	Plug-in

Type	Supported parameters		Comment
	Name	Value	
AddOn	InsertAt	Insert position ('A'-'Z')	Add-on
	Description	Display name	
	WorksWith	Plug-ins delimiter by ';' that are supported by add-on. Empty if all plug-ins are supported.	
	ObjectsSypported	Classes of objects, your add-on will work with. If object with not supported class will be selected in left (hierarchy) pane your add-on will be temporary hidden	
EventHandler	InsertAt	Insert position ('A'-'Z')	Event handler
Registrar			Additional registration object
AccessForm			Access form (IProfileAccessForm, ISearchAccessForm)
CompatibleAppsForm			Compatible application chooser form (ICompatibleAppsForm)
GroupCheckinForm			Check-in form (IGroupCheckinForm)

Type	Supported parameters		Comment
	Name	Value	
GroupCheckoutForm			Check-out form (IGroupCheckoutForm)
HistoryForm			History form (IHistoryForm)
LibrariesForm			Libraries form (ILibrariesForm, ILibraryChooserForm)
LookupForm	FormName	Profile or Search form name. Multiple values should be delimited by ';'.	Lookup form (ILookupForm)
	LookupID	Lookup ID. Multiple values should be delimited by ';'.	
	TargetProperty	Profile or search column name. Multiple values should be delimited by ';'.	
MailForm			Mail form (IMailForm)
ProfileForm	FormName	Profile form name	Profile form (IProfileForm)
ProfileFormChooser			Profile form chooser (IProfileFormChooser)
ProjectChooserForm			Folders chooser form (IProjectChooserForm). Obsolete.
ProjectForm			Edit Folder form (IProjectForm)

Type	Supported parameters		Comment
	Name	Value	
ProjectMembershipForm			Not supported (IProjectMembershipForm)
QuickRetrieveForm	FormName	Quick Retrieve form name.	QuickRetrieve form (IQuickRetrieveForm)
SaveOptionsForm			Save As form (ISaveAsOptionsForm)
SearchCriteriaForm	FormName	QuickSearch form name	QuickSearch criteria form (ISearchCriteriaForm)
SearchFormChooser			Search form chooser form (ISearchFormChooser)
SearchOptionsForm			QuickSearch options form (ISearchOptionsForm)
SelectVersionsForm			Select versions form (ISelectVersionsForm)
TemplatesForm			Templates form (ITemplateChooserForm)
VersionChooserForm			Version chooser form (IVersionChooserForm)
PropertiesForm			Version properties form (IPropertiesForm)
LaunchMethodChooserForm			Launch method chooser form (ILaunchMethodChooserForm)
LocationForm			Select library form (ILocationForm)
ExportForm			Export form (IExportForm)
RelatedItemsForm			Related form (IRelatedItemsForm)
SelectVersionTypeForm			Select version type form (ISelectVersionTypeForm)

Type	Supported parameters		Comment
	Name	Value	
ChangeDisplayNameForm			Change Folder's component display name form (IChangeDisplayName)

Other type of custom objects can be available via implemenation custom Registrar object.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IBCRegistrarExt" on page 145

IBCRetrievalSink

The BCRetrievalSink object is implemented by browser and used for receiving notification from BCEnumObjects if this objects supports background enumeration.

Methods

IBCRetrievalSink Methods	Description
IBCRetrievalSink::LastRowRetrieved	The method is called when enumerator has finished object enumeration.
IBCRetrievalSink::RowsRetrieved	The method is called when next part of objects have been retrieved.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

See Also

"IBCEnumObjects" on page 57

IBCRetrievalSink::LastRowRetrieved

The method is called when enumerator has finished object enumeration.

C++ Syntax

```
HRESULT LastRowRetrieved(  short sReason,  long lTotalRows,  long
lEnumID);
```

Visual Basic Syntax

```
BCRetrievalSink.LastRowRetrieved (sReason, lTotalRows, lEnumID)
```

Parameters

sReason	[in] A Short value that indicates the reason of finishing object enumeration and can be one of the LastRowReason enumeration value.
lTotalRows	[in] Total objects have been retrieved.
lEnumID	[in] Enumerator ID that has been put in the IBCEnumObjects::Retrieve method.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCRetrievalSink" on page 151

"IBCEnumObjects" on page 57

"LastRowReason Enumeration" on page 178

IBCRetrievalSink::RowsRetrieved

The method is called when the next part of objects have been retrieved.

C++ Syntax

```
HRESULT RowsRetrieved(  long lCurrentLine,  long lEnumID);
```

Visual Basic Syntax

```
BCRetrievalSink.RowsRetrieved (lCurrentLine, lEnumID)
```

Parameters

lCurrentLine	[in] A Long value that specifies how many objects have been retrieved.
lEnumID	[in] Enumerator ID that has been put in the IBCEnumObjects::Retrieve method.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCRetrievalSink" on page 151

"IBCEnumObjects" on page 57

IBCSession

The BCSession object represents an Explorer Core session. Every client connected to the core must have at least one session. A BCSession object may be created with the CreateObject function or a Dim statement with the New keyword.

When to Implement

The session object is implemented by the core and manages data (or plug-ins) that should be visually presented by a browser associated with the session. It can be used to identify a particular client that calls on some object.

Methods

IBCSession Methods	Description
IBCSession::Logon	The method logs the user into the session in a specified browser.
IBCSession::GetSessionInfo	The method returns session attributes as an associative map.
IBCSession::EnumChildObjects	The method enumerates all root objects provided by registered plug-ins.
IBCSession::EnumAllVerbs	The method enumerates all commands available to plug-in objects.
IBCSession::ObjectByID	The method returns a specific object by its identifier.

Properties

IBCPropertyArray Properties	Description
IBCSession::SID	The property returns the session identifier.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

IBCSession::EnumAllVerbs

The method enumerates all commands available to plug-in objects.

C++ Syntax

```
HRESULT EnumAllVerbs(
  IBCCoreCtx* pCoreCtx,
  IBCEnumVerbs** ppEnumVerbs
);
```

Visual Basic Syntax

```
[ Set ppEnumVerbs  = ] BCSession.EnumAllVerbs (pCoreCtx)
```

Parameters

pCoreCtx	[in] A BCCoreCtx object. The parameter is not used now.
ppEnumVerbs	[out, retval] The method returns a BCEnumVerbs object that enumerates the requested commands.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

The method is used when initially filling the menu and toolbar. This method composes the commands by aggregating enumerators received from plug-ins. Plug-ins can have dependencies between their verbs. Since the order in which plug-ins are enumerated is not defined, the commands can have references to still undefined identifiers. Therefore, a browser must first collect all verbs and then decide which of them refer to non-existent ones.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCSession" on page 153

"IBCCoreCtx" on page 46

"IBCEnumVerbs" on page 74

IBCSession::EnumChildObjects

The method enumerates all root objects provided by registered plug-ins.

C++ Syntax

```
HRESULT EnumChildObjects(
  IBCCoreCtx* pCoreCtx,
```

```
  IUnknown** ppIEnumObjects
);
```

Visual Basic Syntax

```
[ Set ppIEnumObjects  = ] BCSession.EnumChildObjects (pCoreCtx)
```

Parameters

pCoreCtx	[in] A BCCoreCtx object that specifies enumeration parameters. For possible values see IBCObject::EnumChildObjects.
ppIEnumObjects	[out, retval] Returns a BCEnumObjects object that enumerates the root objects of plug-ins.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

The resulting enumerator is composed of all the enumerators provided by plug-ins in the order that conforms to the registry InsertAt settings. Objects from different plug-ins are not mixed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCSession" on page 153

"IBCCoreCtx" on page 46

"IBCEnumVerbs" on page 74

"IBCObject" on page 80

IBCSession::GetSessionInfo

The method returns session attributes as an associative map.

C++ Syntax

```
HRESULT GetSessionInfo(
  IBCCoreCtx** ppCoreCtx
);
```

Visual Basic Syntax

```
[ Set ppCoreCtx  = ] BCSession.GetSessionInfo
```

Parameters

ppCoreCtx	[out, retval] Returns a BCCoreCtx object that contains the requested session attributes as follows:

Key	Value
SID	A string that uniquely identifies the session.
LoadedPlugins	A string that enumerates plug-in classes separated by a semicolon.
VersionsSupported	A Boolean that indicates whether or not there are plug-ins that support versions (context versions). True: There exists a plug-in that supports versions. False: There exist no plug-ins that support versions.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCSession" on page 153

"IBCCoreCtx" on page 46

IBCSession::Logon

The method logs the user into the session in a specified browser.

C++ Syntax

```
HRESULT Logon(  IBCBrowser* pIBCBrowser);
```

Visual Basic Syntax

```
BCSession.Logon (pIBCBrowser)
```

Parameters

pIBCBrowser	[in] A BCBrowser object that specified a browser that will present the session data.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

The Explorer Core uses this method for explicit logging on. The method calls the Logon methods on plug-ins for (optional) user identification.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCSession" on page 153

"IBCBrowser" on page 25

IBCSession::ObjectByID

The method returns a specific object by its identifier.

C++ Syntax

```
HRESULT ObjectByID(
  BSTR  bstrObjectID,
```

```
  IBCObject** ppObj
);
```

Visual Basic Syntax

```
[ Set ppObj  = ] BCSession.ObjectByID (bstrObjectID)
```

Parameters

bstrObjectID	[in] A string that uniquely identifies the required object.
ppObj	[out, retval] Returns the BCObject object identified by bstrObjectID.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCSession" on page 153

"IBCObject" on page 80

IBCSession::SID

The property returns the session identifier.

C++ Syntax

```
HRESULT SID(
  BSTR* pbstrSID
);
```

Visual Basic Syntax

```
[ pbstrSID = ] BCSession.SID
```

Parameters

pbstrSID	[out, retval] Returns a string that uniquely specifies the session.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Access

Read Only

Remarks

While creating a session, the core defines a session identifier. The developer should use this identifier to uniquely specify a BCSession object. SID is the default property for a BCSession object.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCSession" on page 153

IBCVerb

The IBCVerb object represents a menu or toolbar command (verb).

When to Implement

The BCVerb object is implemented by plug-ins.

Methods

IBCVerb Methods	Description
IBCVerb::Icon	The method returns the default icon for a toolbar button's normal or "up" state.
IBCVerb::IconActive	The method returns the icon for a toolbar button's focused state (when the mouse is over the icon on the toolbar).
IBCVerb::IconDown	The method returns the icon for a toolbar button's selected or "down" state.

Properties

IBCVerb Properties	Description
IBCVerb::ID	The property returns the command identifier.
IBCVerb::DisplayName	The property returns the name of the command to display in the menu.
IBCVerb::Flags	The property returns the command state.
IBCVerb::Parent	The property returns the identifier of the parent menu item for a pop-up menu.
IBCVerb::InsertBefore	The property returns the identifier of the command before which this verb is placed.
IBCVerb::Description	The property returns the command's description.
IBCVerb::ToolTip	The property returns the command tooltip.
IBCVerb::ShortName	The property returns the command short name that will appear on the toolbar.
IBCVerb::AccelKey	The property returns the command shortcut.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

IBCVerb::AccelKey

The property returns the command shortcut.

C++ Syntax

```
HRESULT AccelKey(
  long* plAccelKey
);
```

Visual Basic Syntax

```
[ plAccelKey = ] BCVerb.AccelKey
```

Parameters

plAccelKey	[out, retval] Returns a long that specifies the command shortcut. Its low-order word is either a virtual-key code or an ASCII character code. The high-order word specifies the accelerator flags. This word can be a combination of the following values:

Value	Description
FSHIFT (&H4)	The Shift key must be held down when the accelerator key is pressed.
FCONTROL (&H8)	The Ctrl key must be held down when the accelerator key is pressed.
FALT (&H10)	The Alt key must be held down when the accelerator key is pressed.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Access

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCVerb" on page 160

IBCVerb::Description

The property returns the command's description.

C++ Syntax

```
HRESULT Description(
  BSTR* pbstrDescription);
```

Visual Basic Syntax

```
[ pbstrDescription = ] BCVerb.Description
```

Parameters

pbstrDescription	[out, retval] Returns a string that specifies the full descriptive name of the command to appear in the status bar.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Access

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCVerb" on page 160

IBCVerb::DisplayName

The property returns the name of the command to display in the menu.

C++ Syntax

```
HRESULT DisplayName(  BSTR* pbstrName);
```

Visual Basic Syntax

```
[ pbstrName = ] BCVerb.DisplayName
```

Parameters

pbstrName	[out, retval] Returns a string that specifies the command name.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Access

Read Only

Remarks

If this property is empty a string menu command will not be added (useful for accelerators or toolbar buttons only).

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCVerb" on page 160

IBCVerb::Flags

The property returns the command state.

C++ Syntax

```
HRESULT Flags(
  long* pFlags
);
```

Visual Basic Syntax

```
[ pFlags = ] BCVerb.Flags
```

Parameters

pFlags	[out, retval] Returns a long that specifies the command state. It can be one or more of the VerbFlags enumeration values.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Access

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCVerb" on page 160

"VerbFlags Enumeration" on page 182

IBCVerb::Icon

The method returns the default icon for normal or "up" state of a toolbar button.

C++ Syntax

```
HRESULT Icon(
  VARIANT_BOOL  bSmall,
  long* plIconHandle
);
```

Visual Basic Syntax

```
[ plIconHandle  = ] BCVerb.Icon (bSmall)
```

Parameters

bSmall	[in] A Boolean value that indicates whether the icon being retrieved is small or large as follows: True: Icons are small. False: Icons are large.
plIconHandle	[out, retval] Returns a long that specifies the HICON value of the default icon for normal or "up" state of a toolbar button.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCVerb" on page 160

IBCVerb::IconActive

The method returns the icon for toolbar button's focused state (when the mouse is over a toolbar icon).

C++ Syntax

```
HRESULT IconActive(
  VARIANT_BOOL  bSmall,
  long* plIconHandle
);
```

Visual Basic Syntax

```
[ plIconHandle  = ] BCVerb.IconActive (bSmall)
```

Parameters

bSmall	[in] A Boolean value that indicates whether the icon being retrieved is small or large as follows: True: Icons are small. False: Icons are large.
plIconHandle	[out, retval] Returns a long that specifies the HICON value of the icon for a toolbar button's focused state.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCVerb" on page 160

IBCVerb::IconDown

The method returns the icon for a selected or "down" state of a toolbar button.

C++ Syntax

```
HRESULT IconDown(
  VARIANT_BOOL  bSmall,
  long* plIconHandle
);
```

Visual Basic Syntax

```
[ plIconHandle  = ] BCVerb.IconDown (bSmall)
```

Parameters

bSmall	[in] A Boolean value that indicates whether the icon being retrieved is small or large as follows: True: Icons are small. False: Icons are large.
plIconHandle	[out, retval] Returns a long that specifies the HICON value of the icon for selected or "down" state of a toolbar button.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCVerb" on page 160

IBCVerb::ID

The property returns the command identifier.

C++ Syntax

```
HRESULT ID(
  BSTR* pbstrCmdID
);
```

Visual Basic Syntax

```
[ pbstrCmdID = ] BCVerb.ID
```

Parameters

pbstrCmdID	[out, retval] Returns a string that uniquely specifies the command.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Access

Read Only

Remarks

The verb identifier consists of two parts: the plug-in class and a string that specifies the verb within the plug-in.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCVerb" on page 160

IBCVerb::InsertBefore

The property returns the identifier of the command this verb is placed before.

C++ Syntax

```
HRESULT InsertBefore(
  BSTR* pbstrID
);
```

Visual Basic Syntax

```
[ pbstrID = ] BCVerb.InsertBefore
```

Parameters

pbstrID	[out, retval] Returns a string that specifies the identifier of the command this verb is placed before. If it is an empty string, the command is added at the end of the menu.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Access

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCVerb" on page 160

IBCVerb::Parent

The property returns the identifier of the parent menu item for a pop-up menu.

C++ Syntax

```
HRESULT Parent(
  BSTR* pbstrParent
);
```

Visual Basic Syntax

```
[ pbstrParent = ] BCVerb.Parent
```

Parameters

pbstrParent	[out, retval] Returns a string that uniquely specifies the command's parent.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Access

Read Only

Remarks

Plug-ins can refer to the menu items predefined by a browser:

Menu Item	ID
File	BROWSER.FILE
File | Exit	BROWSER.FILE_EXIT
View	BROWSER.VIEW
View | List	BROWSER.VIEW_LIST
View | Detail	BROWSER.VIEW_DETAIL
Help	BROWSER.HELP
Window	BROWSER.WINDOW

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCVerb" on page 160

IBCVerb::ShortName

The property returns the command short name that will appear on the toolbar.

C++ Syntax

```
HRESULT ShortName(
  BSTR* pbstrShortName
);
```

Visual Basic Syntax

```
[ pbstrShortName = ] BCVerb.ShortName
```

Parameters

pbstrShortName	[out, retval] Returns a string that specifies the command toolbar button name.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Access

Read Only

Remarks

If this property is empty a toolbar button will not be added for this command.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCVerb" on page 160

IBCVerb::ToolTip

The property returns the command tooltip.

C++ Syntax

```
HRESULT ToolTip(
  BSTR* pbstrTooltip
);
```

Visual Basic Syntax

```
[ pbstrTooltip = ] BCVerb.ToolTip
```

Parameters

pbstrTooltip	[out, retval] Returns a string that specifies the command tooltip.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Access

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IBCVerb" on page 160

Explorer Core Enumerations

BrowseObjectFlags Enumeration

This enumeration contains flags that specify how an object should be browsed. A combination of these flags is passed to IBCBrowser::BrowseObject as its sFlags parameter.

Elements

BrowseObjectFlags Elements	Description
BO_RELATIVE	Indicates that a specified path is relative from the current object. If this flag is not specified, the bstrPath parameter in the IBCBrowser::BrowseObject method specifies the full path for an object.
BO_NEWWINDOW	Indicates that an object should be browsed in a new window.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

See Also

"IBCBrowser" on page 25

BrowserInfoFlags Enumeration

This enumeration contains flags that specify browser information. A combination of these flags is used in IBCBrowser::GetBrowserInfo (the plBrowserInfo parameter).

Elements

BrowserInfoFlags Elements	Description
BC_MDI	Indicates that the browser supported multiple windows.
BCI_VERB_CALLBACK	Indicates that browser menu and toolbar commands can be modified later, not when selection actually changed. If this flag is not specified plug-ins should update verb's state before leaving the IBCContextMenuSupport::SetAsDefault method.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

See Also

"IBCBrowser" on page 25

"IBCContextMenuSupport" on page 38

CoreEvents Enumeration

This enumeration contains events that are passed to IBCObjectChanged::OnDataChange as its sReason parameter.

Elements

CoreEvents Elements	Description
bcevNone	Indicates that nothing has been changed.
bcevAdd	Indicated that a new object(s) has been added. Newly added objects are passed as the pUnk parameter for the IBCObjectChanged::OnDataChange method. This parameter may be BCObject if only one object has been added or BCObjectArray if more than one object has been added.
bcevDelete	Indicates that object has been deleted. The pUnk parameter for IBCObjectChanged::OnDataChange method contains reference for a deleted object. This method can be BCObject or BCObjectArray. If this parameter is empty then the current object has been deleted.
bcevReorder	Indicates that the child objects have been reordered.
bcevChange	Indicates that current object has been changed.
bcevRefresh	Indicates that the current object and/or its content has been changed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

See Also

"IBCObjectChanged" on page 110

"IBCObject" on page 80

IBCObjectArray

CoreFlags Enumeration

This enumeration contains flags that attribute object PIDL.

Elements

CoreFlags Elements	Description
CF_NEED_BIND	Indicates that the object PIDL does not contain display name.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

GetBrowserFlags Enumeration

This enumeration contains flags that are passed to IBCBrowserCore::GetBrowser as its sFlags parameter.

Elements

GetBrowserFlags Elements	Description
GB_RUN	Indicates that default browser should be started if no browser is registered for the specified session.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

See Also

"IBCBrowserCore" on page 34

KeyFlags Enumeration

This enumeration contains flags that specify keyboard state (the grfKeyState parameter) during IBCObjectArray::DoVerb and IBCContextMenuSupport::DoVerb method's call.

Elements

KeyFlags Elements	Description
kfKey	Indicates that any key is depressed.
kfLeftShift	Indicates that left Shift key is depressed.
kfRigtShift	Indicates that right Shift key is depressed.
kfShift	Indicates that any Shift key is depressed.
kfLeftCtrl	Indicates that left Ctrl key is depressed.
kfRightCtrl	Indicates that right Ctrl key is depressed.
kfCtrl	Indicates that any Ctrl key is depressed.
kfLeftAlt	Indicates that left Alt key is depressed.
kfRightAlt	Indicates that right Alt key is depressed.
kfAlt	Indicates that any Alt key is depressed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

See Also

"IBCObjectArray" on page 91

"IBCContextMenuSupport" on page 38

LastRowReason Enumeration

This enumeration contains flags that specify the reason (the sReason parameter) when the IBCRetrievalSink::LastRowRetrieved notification is sending.

Elements

LastRowReason Elements	Description
LRR_OK	Indicates that all objects have been retrieved.
LRR_CANCEL	Indicates that retrieval has been cancelled. Indicates that the object can be treated as a default (common) object.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

See Also

"IBCRetrievalSink" on page 151

PropertyArrayFlags Enumeration

This enumeration contains flags that are used in IBCPropertyArray::Flags property.

Elements

PropertyArrayFlags Elements	Description
PA_CAN_SAVE_AS_DEFAULT	Properties can be saved as defaults.
PA_CAN_RESTORE_DEFAULTS	Properties can be restored to its default state.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

See Also

"IBCPropertyArray" on page 127

PropertyArraySaveFlags Enumeration

This enumeration contains flags that specify flags (the sFlags parameter) in the IBCPropertyArray::Save method.

Elements

PropertyArraySaveFlags Elements	Description
PA_SHARED	Indicates that array of properties should be saved for all users.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

See Also

"IBCPropertyArray" on page 127

PropertyAttributes Enumeration

This enumeration contains flags that specify property attributes. A combination of these flags is used in the IBCProperty::Flags property.

Elements

PropertyAttributes Elements	Description
bcpaHidden	Indicates that property should not be displayed.
bcpaIcon	Indicates that property is icon.
bcpaRequired	Indicates that property is required and should not be removed from displayed properties.
bcpaDetail	Indicates that property is included in displayed properties.
bcpaJustifyLeft	Indicates that property should be left justified.
bcpaJustifyRight	Indicates that property should be right justified.
bcpaJustifyCenter	Indicates that property should be center justified.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

See Also

"IBCProperty" on page 119

RegisterFlags Enumeration

This enumeration contains flags that specify functionality of a programmable object. A combination of these flags is passed to IBCRegistrar::RegisterClass as its lFlags parameter.

Elements

RegisterFlags Elements	Description
REGISTER_DEFAULT	Indicates that the object can be treated as a default (common) object.
REGISTER_AGGREGATE	Indicates whether the object supports aggregation, i.e. can be created as part of an aggregate.
REGISTER_SINGLETON	Indicates whether the object is a singleton, i.e. only one instance of the object can exist.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

See Also

"IBCRegistrar" on page 141

RegistrarCommand Enumeration

This enumeration contains command that can be passed to IBCRegistrarExt::RegisterExtObjects its nCommand parameter.

Elements

RegistrarCommand Elements	Description
rcEnable	Enable object.
rcRegister	Register object.
rcUnregister	Unregister object.
rcDisable	Disable object.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: import "DECore.idl"

See Also

"IBCRegistrarExt::RegisterExtObject" on page 146

VerbFlags Enumeration

This enumeration contains flags that specify functionality of a menu command/toolbar button. A combination of these flags is used in the IBCVerb::Flags property.

Elements

VerbFlags Elements	Description
VF_ENABLED	Indicates that menu command is enabled.
VF_DISABLED	Indicates that menu command is disabled.
VF_UNHECKED	Indicates that menu command is unchecked.
VF_CHECKED	Indicates that menu command is checked.
VF_SEPARATOR	Indicates that menu command is separator.
VF_DEFAULT	Indicates that menu command is default.
VF_ON_TOOLBAR_BY_DEFAULT	Indicates that menu command should be added to the toolbar by default.
VF_INVISIBLE	Indicates that menu command is invisible (accelerator only command).
VF_POPUP	Indicates that a menu command is the pop-up sub-menu.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

See Also

"IBCVerb" on page 160

ViewMode Enumeration

This enumeration contains flags that specify view mode.

Elements

KeyFlags Elements	Description
VM_LIST	Indicates the List view mode.
VM_DETAIL	Indicates the Detail view mode.
VM_ICON	Indicates the Large Icons view mode.
VM_SMALLICON	Indicates the Small Icons view mode.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "DECore.idl"

See Also

"IBCObjectArray" on page 91

"IBCContextMenuSupport" on page 38

Chapter

3

DM Object Model

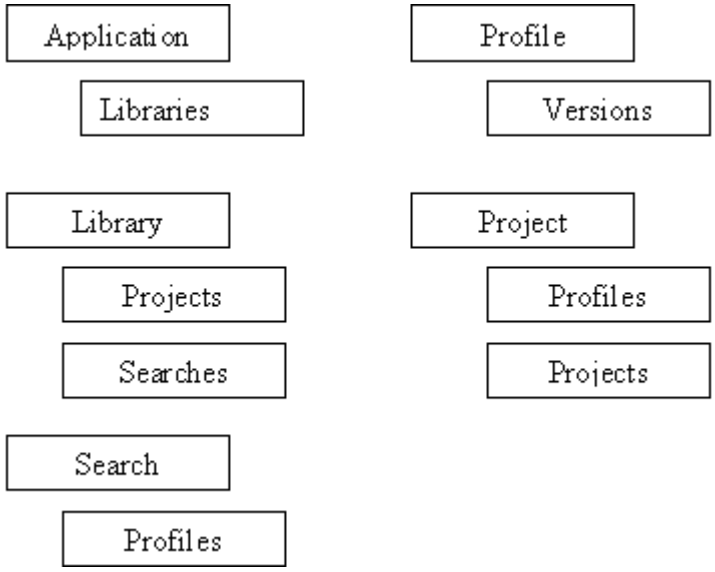
In This Chapter

- "Document Management Objects" on page 186
- "Document Management Objects and Services" on page 186
- "Document Management Objects Structure" on page 187
- "OM Creatable Objects" on page 192
- "Object Model Interfaces" on page 202
- "Object Model Enumerations" on page 679

Document Management Objects

The Document Management Objects are the central part of DM Extensions. They represent a high-level set of document management-oriented objects and related services used to access and manipulate these objects. Examples include a Profile object that encapsulates the entire document functionality and a Project object representing a folder.

The DM objects can operate in completely UI-less mode. Thus you can build your own product, which will use the DMO internally and will never expose any forms or message boxes. You can also use the DMO in its normal mode, when it pops up the forms to gather information from the user.



The developer has access to the entire hierarchy of the DM objects. This includes DM-specific objects such as Profile, Project, Search and several general-purpose objects such as Application.

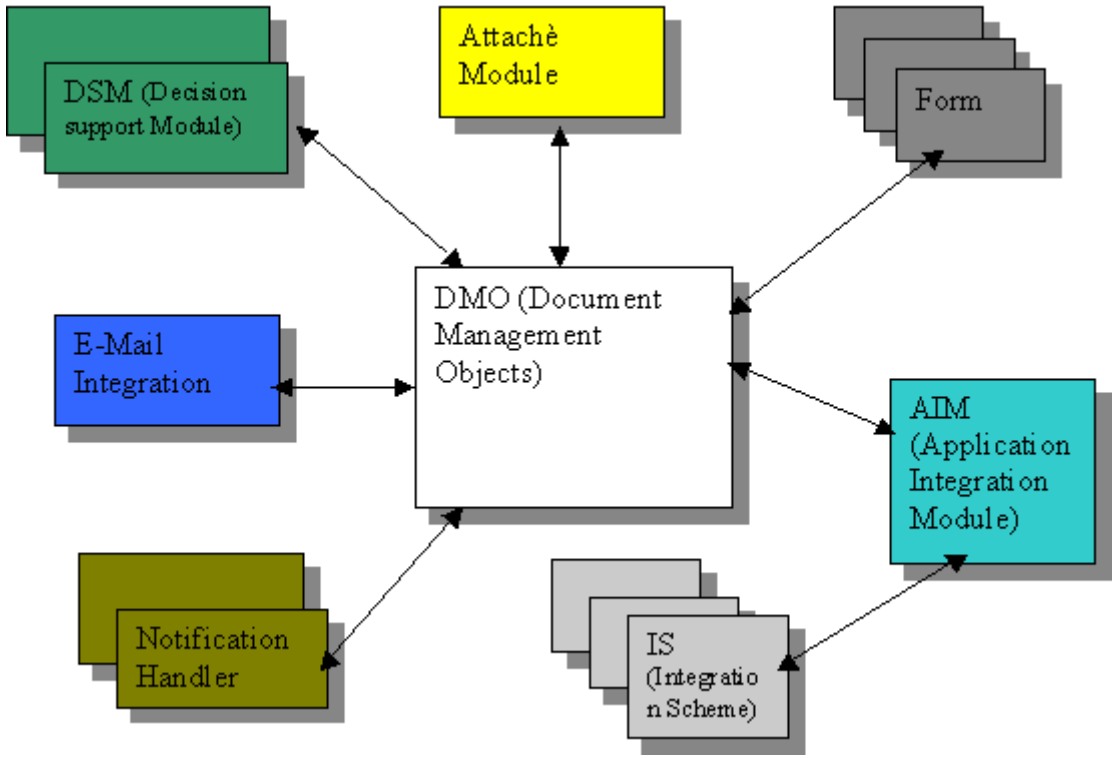
Document Management Objects and Services

The DM Objects and Services of DM Extensions (DMO) provide a solid foundation for solution developers. These objects provide a base-

level document management functionality, which can be used in all applications.

The DMO are COM-based so any modern development environment can be used to develop document management solutions.

The next picture presents the most important components of the DMO. Note that you can customize, extend, or even replace all colored components.

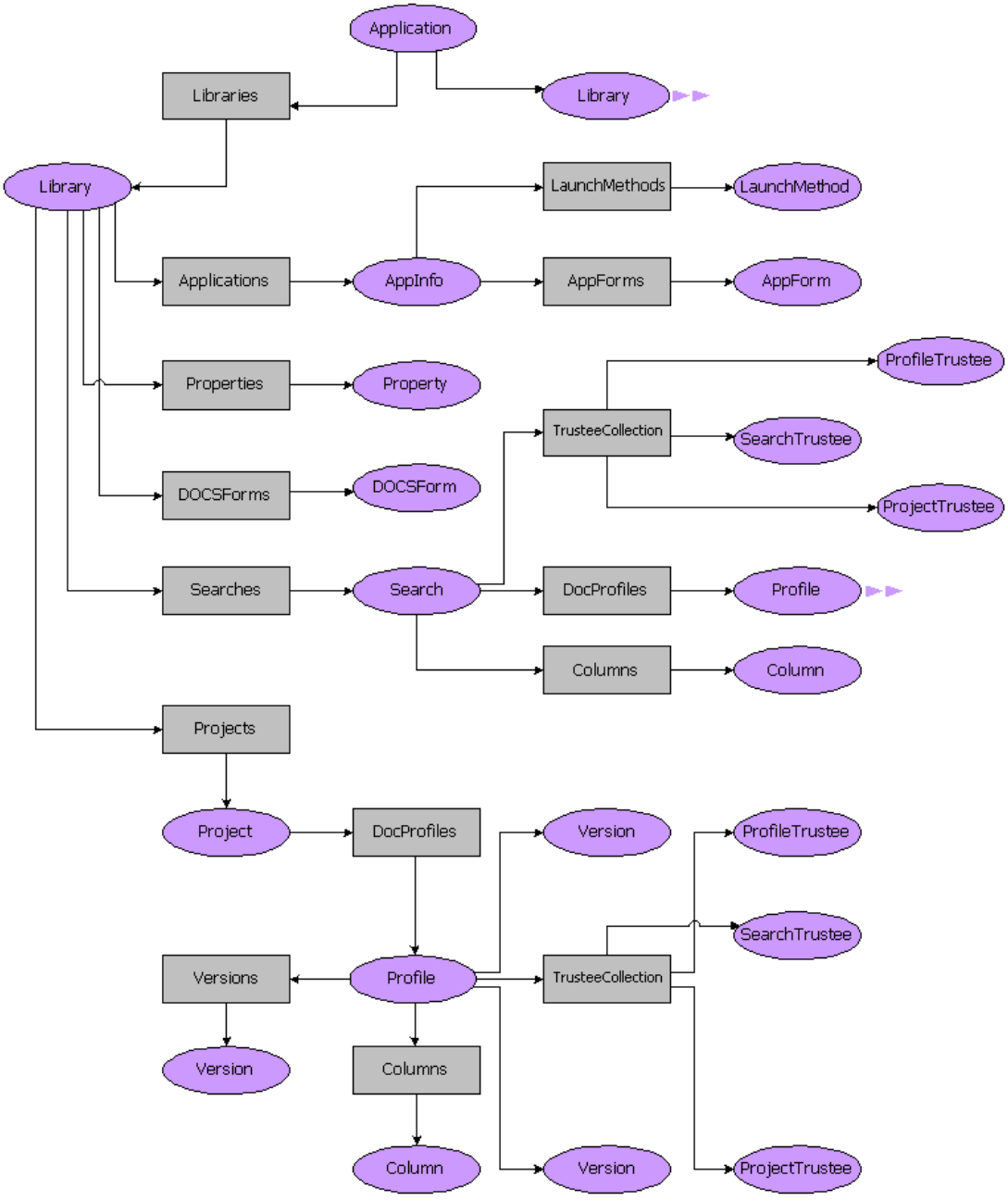


Document Management Objects Structure

- "Application Tree" on page 188
- "Rest Objects" on page 190
- "Using the DM Object Model" on page 191

Application Tree

Application Object Structure

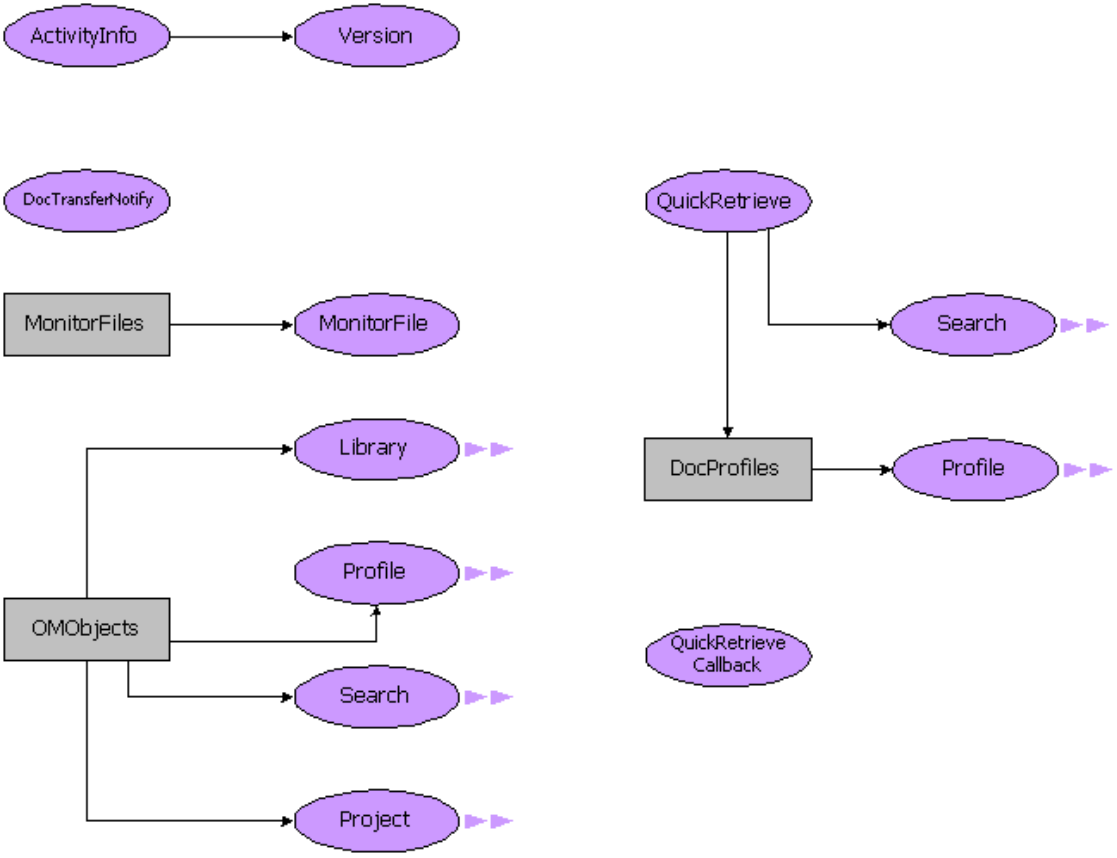


The following table shows how objects and collections are represented in the diagrams.

Object	
Collection	
Structure description of the object is not finished here. To be continued nearby.	

Rest Objects

DM objects structure.



The following table shows how objects and collections are represented in the diagrams.

Object	
Collection	

Structure description of the object is not finished here. To be continued nearby.



Using the DM Object Model

How to use Document Management Objects

This section presents several examples, which should help you to understand the basic principles of working with DMO. These examples should be used together with DMO Reference.

How to create Profile

```
' Path to the document you want to import
Dim path as String
Path = "c:\document.doc"

' Declare and create new Application object
Dim app As New Application
' Version object
Dim ver As Version
' Profile object
Dim p As Profile

' Create new profile based on the "def_prof" form
Set p = app.PrimaryLibrary.CreateProfile("def_prof")

' Specify the APPLICATION for the profile
p.Columns("APP_ID").Value = "MS WORD"

' Show profile form and allow user to make changes to it
If p.ShowProfile(True) Then
    ' If successful, attempt to save the profile to the database
    If p.Save(OF_NORMAL) Then
        ' if successful, copy file to the path, specified by the
Version object
        Set ver = p.Versions(1)
        ver.LockVersionForEdit(OF_NORMAL)
        FileCopy path, ver.FilePath
        ' and Upload file to the server
```

```
        ver.UploadFile
        ver.UnlockVersion(OF_NORMAL)
        ' since we are not going to use the profile anymore – free
it
        p.ReleaseDoc
    End If
End If_
```

Note:

If you want to perform a silent import (without any forms) you should replace call to p.ShowProfile() with lines like that:

```
p.Columns("AUTHOR_ID").Value = "AUTHOR_NAME"
p.Columns("APP_ID").Value = "MS WORD"
```

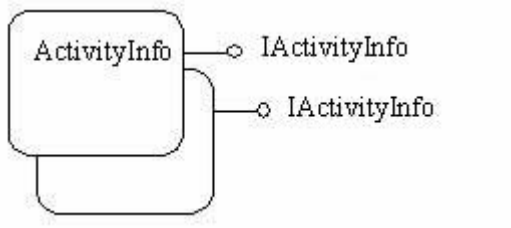
and so on. In other words, you should manually fill all the required profile fields.

OM Creatable Objects

- "ActivityInfo" on page 193
- "Application" on page 193
- "DocProfiles" on page 195
- "Profile" on page 195
- "Project" on page 197
- "Projects" on page 198
- "QuickRetrieve" on page 199
- "Search" on page 200
- "Versions" on page 201

ActivityInfo

The ActivityInfo object represents a record in the DM activity log. It contains information about an activity that occurs with a profiled document.



C++ Syntax

```
const CLSID CLSID_ActivityInfo =
{0xF372BD85,0xDF45,0x11D1,{0x91,0xFC,0x00,0x60,0x08,0x3F,0x9B,0x01
}};
IUnknown* pUnk;
HRESULT hr = ::CoCreateInstance(&CLSID_ActivityInfo, NULL,
CLSCTX_ALL, IID_IUnknown, (void**)&pUnk);
```

Visual Basic Syntax

```
Sub Form_Load()
...
Dim Activity As Object
' Create a DOCSObjects.ActivityInfo object
Set Activity = CreateObject("DOCSObjects.ActivityInfo")
...
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "fsplugin.dll"

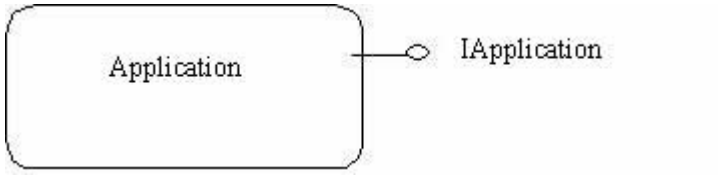
See Also

"IActivityInfo" on page 202

Application

The class identifies the Document Management plug-in. There is only one Application object. Creating the DOCSObjects.Application object will start the Document Management plug-in. No other of the

Document Management Objects properties or methods can be used until the plug-in is started.



C++ Syntax

```
const CLSID CLSID_Application =
{0x1CF0DE76,0x8CAE,0x11D1,{0x82,0xA2,0x00,0x60,0x08,0x3F,0x9B,0x01
}};

IUnknown* pUnk;

HRESULT hr = ::CoCreateInstance(&CLSID_Application, NULL,
CLSCTX_ALL, IID_IUnknown, (void**)&pUnk);
```

Visual Basic Syntax

```
Sub Form_Load()

...

Dim App As Object

...

' Create the DOCSObjects.Application object

Set App = CreateObject("DOCSObjects.Application")

...

End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

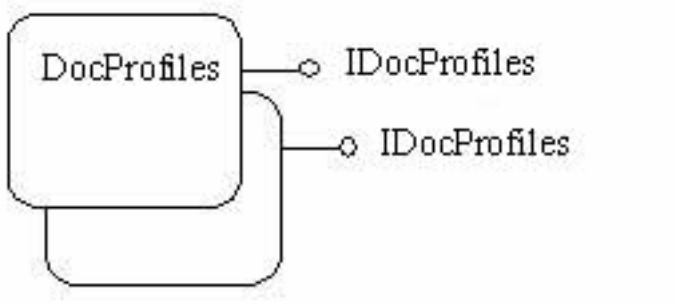
Header: #import "fsplugin.dll"

See Also

"IApplication" on page 241

DocProfiles

The DocProfiles object represents a collection of document profiles.



C++ Syntax

```
const CLSID CLSID_DocProfiles =
{0x1CF0DE88,0x8CAE,0x11D1,{0x82,0xA2,0x00,0x60,0x08,0x3F,0x9B,0x01
}};
IUnknown* pUnk;
HRESULT hr = ::CoCreateInstance(&CLSID_DocProfiles, NULL,
CLSCTX_ALL, IID_IUnknown, (void**)&pUnk);
```

Visual Basic Syntax

```
Sub Form_Load()
...
Dim Profs As Object
' Create a DOCSObjects.DocProfiles object
Set Profs = CreateObject("DOCSObjects.DocProfiles")
...
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "fsplugin.dll"

See Also

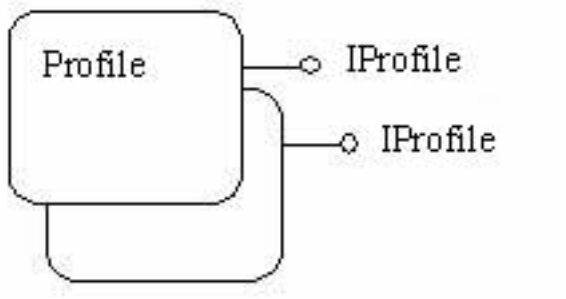
"IDocProfiles" on page 292

Profile

The Profile object represents a DM profile, a record in the SQL database that contains information about a single document. It is the means by which the user retrieves, views, and manipulates a document.

Each Profile object visually presented by its form. A profile form depends on the primary group of the user working with the system and the application a document should be associated with. Moreover, every application can have different profile forms. A profile form that will be used to store information about the document should be specified when profiling the document.

The Profile object inherits its definition from the IOMErrInfo interface so its implementation also includes all the properties of IOMErrInfo in addition to the ones described below.



C++ Syntax

```
const CLSID CLSID_Profile =
{0x1CF0DE7E,0x8CAE,0x11D1,{0x82,0xA2,0x00,0x60,0x08,0x3F,0x9B,0x01
}};
IUnknown* pUnk;
HRESULT hr = ::CoCreateInstance(&CLSID_Profile, NULL, CLSCTX_ALL,
IID_IUnknown, (void**)&pUnk);
```

Visual Basic Syntax

```
Sub Form_Load()
...
Dim Prof As Object
' Create a DOCSObjects.Profile object
Set Prof = CreateObject("DOCSObjects.Profile")
Prof.SetDocument "Test", 20
...
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "fsplugin.dll"

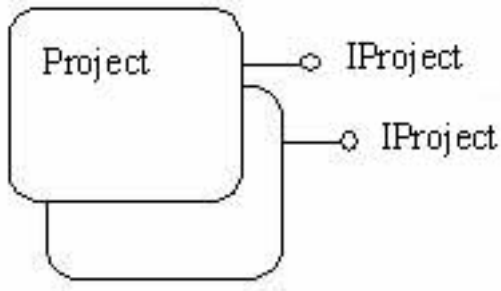
See Also

"IProfile" on page 419

Project

The Project object represents a DM project, a collection of related documents. Projects allow the user to group documents used for the same purpose. A single document may be included in multiple projects. A project may even be included in a larger project, becoming a sub-project to the larger one.

The Project object inherits its definition from the IOMErrInfo interface so its implementation also includes all the properties of IOMErrInfo in addition to the ones described below.



C++ Syntax

```
const CLSID CLSID_Project =
{0x1CF0DE7A,0x8CAE,0x11D1,{0x82,0xA2,0x00,0x60,0x08,0x3F,0x9B,0x01
}};
IUnknown* pUnk;
HRESULT hr = ::CoCreateInstance(&CLSID_Project, NULL, CLSCTX_ALL,
IID_IUnknown, (void**)&pUnk);
```

Visual Basic Syntax

```
Sub Form_Load()
...
Dim Prj As Object
' Create a DOCSObjects.Project object
Set Prj = CreateObject("DOCSObjects.Project")
Prj.SetProject "Test", "My Project"
...
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

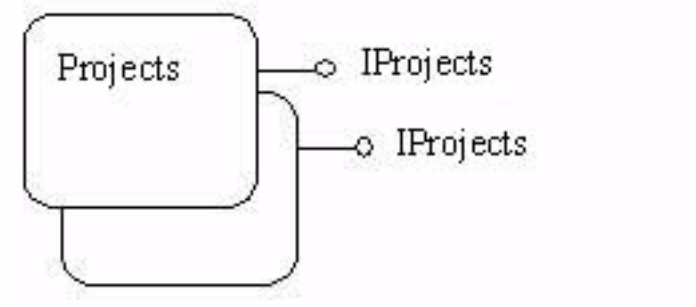
Header: #import "fsplugin.dll"

See Also

"IProject" on page 497

Projects

The Projects object represents a collection of Projects.



C++ Syntax

```
const CLSID CLSID_Projects =
{0x3DC10CF3,0xEFBD,0x11D1,{0x8B,0x37,0x00,0x60,0x97,0x75,0xD3,0x51
}};
IUnknown* pUnk;
HRESULT hr = ::CoCreateInstance(&CLSID_Projects, NULL, CLSCTX_ALL,
IID_IUnknown, (void**)&pUnk);
```

Visual Basic Syntax

```
Sub Form_Load()
...
Dim Prjs As Object
' Create a DOCSObjects.Projects object
Set Prjs = CreateObject("DOCSObjects.Projects")
...
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "fsplugin.dll"

See Also

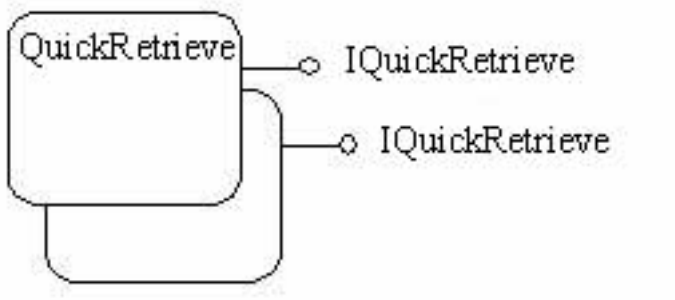
"IProjects" on page 534

"Project" on page 197

QuickRetrieve

The QuickRetrieve object represents a DM Extensions Quick Retrieve form.

The QuickRetrieve object inherits its definition from the IOMErrInfo interface so its implementation also includes all the properties of IOMErrInfo in addition to the ones described below.



C++ Syntax

```
const CLSID CLSID_QuickRetrieve =
{0x1CF0DE85,0x8CAE,0x11D1,{0x82,0xA2,0x00,0x60,0x08,0x3F,0x9B,0x01
}};
IUnknown* pUnk;
HRESULT hr = ::CoCreateInstance(&CLSID_QuickRetrieve, NULL,
CLSCTX_ALL, IID_IUnknown, (void**)&pUnk);
```

Visual Basic Syntax

```
Sub Form_Load()
...
Dim QR As Object
' Create a DOCSObjects.QuickRetrieve object
Set QR = CreateObject("DOCSObjects.QuickRetrieve")
...
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "fsplugin.dll"

See Also

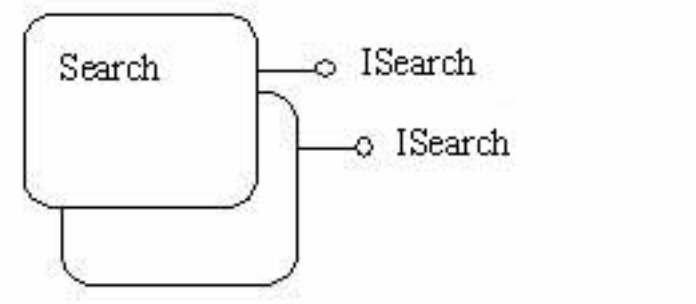
"IQuickRetrieve" on page 556

Search

The Search object represents a DM Extensions quick search, an easy way to find particular documents by providing information about these documents from the users.

Search object's criteria specified in the search criteria form. The form depends on the primary group of the user working with the system.

The Search object inherits its definition from the IOMErrInfo interface so its implementation also includes all the properties of IOMErrInfo in addition to the ones described below.



C++ Syntax

```
const CLSID CLSID_Search =
{0x1CF0DE7C,0x8CAE,0x11D1,{0x82,0xA2,0x00,0x60,0x08,0x3F,0x9B,0x01
}};
IUnknown* pUnk;
HRESULT hr = ::CoCreateInstance(&CLSID_Search, NULL, CLSCTX_ALL,
IID_IUnknown, (void**)&pUnk);
```

Visual Basic Syntax

```
Sub Form_Load()
...
Dim QS As Object
' Create a DOCSObjects.Search object
Set QS = CreateObject("DOCSObjects.Search")
QS.SetSearch "Test", "My Search"
```

```
...
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

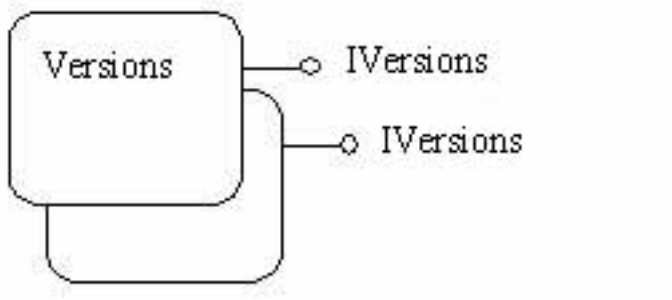
Header: #import "fsplugin.dll"

See Also

"ISearch" on page 567

Versions

The Versions object represents a collection of document's versions.



C++ Syntax

```
const CLSID CLSID_Versions =
{0xC47B9C93,0x9A59,0x11d2,{0xA1,0xA7,0x00,0xA0,0x24,0x74,0x62,0x93
}};
IUnknown* pUnk;
HRESULT hr = ::CoCreateInstance(&CLSID_Versions, NULL, CLSCTX_ALL,
IID_IUnknown, (void**)&pUnk);
```

Visual Basic Syntax

```
Sub Form_Load()
...
Dim Vers As Object
' Create a DOCSObjects.Versions object
Set Vers = CreateObject("DOCSObjects.Versions")
...
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "fsplugin.dll"

See Also

"IVersions" on page 673

Object Model Interfaces

IActivityInfo

The ActivityInfo object represents a record in the DM Extensions activity log. It contains data about an activity that occurs to a profiled document. The activity may be recorded with the AddDocumentActivity method on a Profile object.

An ActivityInfo object may be created with the CreateObject function or a Dim statement with the New keyword. The following example shows the Visual Basic syntax for creating a DOCSObjects.ActivityInfo object:

```
Sub Form_Load()
 ...
 Dim Activity As Object
 ' Create a DOCSObjects.ActivityInfo object
 Set Activity = CreateObject("DOCSObjects.ActivityInfo")
 ...
 End Sub
```

Properties

IActivityInfo Properties	Description
App	Returns/sets the application of an activity.
Author	Returns/sets the author of an activity.
Billable	Returns/sets whether an activity is billable.
ElapsedTime	Returns/sets the amount of elapsed time in seconds while opening a document.
Keystrokes	Returns/sets the number of activity keystrokes.
Pages	Returns/sets the number of pages printed for a print activity.
RefDoc	Returns/sets the source document in a copied-from activity or the target document in a copied-to activity.
TypeTime	Returns/sets the time in seconds spent editing the document in an activity.
Typist	Returns/sets the user who performed an activity.
Version	Returns/sets the activity document version.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

IActivityInfo::App

The property returns or sets the application used for the create,edit, access, look, or new version activity.

C++ Syntax

```
HRESULT App(VARIANT* pvApp);
HRESULT App(VARIANT vApp);
```

Visual Basic Syntax

```
[ pvApp = ] ActivityInfo.App
ActivityInfo.App [ = vApp ]
```

Parameters

pvApp	[out, retval] An expression that specifies the application used for a create, edit, access, look, or new version activity. Returns the application identifier.
vApp	[in] An expression that specifies the application used for a create, edit, access, look, or new version activity. If a string expression, vApp should specify the application identifier. If an object, vApp should specify a valid AppInfo object that identifies an application.

Return Values

S_OK	The operation succeeded.
E_INVALIDARG	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns/sets the application of an activity.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IActivityInfo" on page 202

"IAppInfo" on page 220

IActivityInfo::Author

The property returns or sets the author of the document for the create, edit, or new version activity.

C++ Syntax

```
HRESULT Author(VARIANT* pvAuthor);
HRESULT Author(VARIANT vAuthor);
```

Visual Basic Syntax

```
[ pvAuthor = ] ActivityInfo.Author
ActivityInfo.Author [ = vAuthor ]
```

Parameters

pvAuthor	[out, retval] A string that specifies the document author's name.
vAuthor	[in] A string that specifies the document author's name.

Return Values

S_OK	The operation succeeded.
E_INVALIDARG	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns/sets the author of an activity.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IActivityInfo" on page 202

IActivityInfo::Billable

The property returns or sets whether or not the create, edit, print, or new version activity is billable.

C++ Syntax

```
HRESULT Billable(VARIANT* pbBillable);
```

```
HRESULT Billable(VARIANT bBillable);
```

Visual Basic Syntax

```
[ pbBillable = ] ActivityInfo.Billable
ActivityInfo.Billable [ = bBillable ]
```

Parameters

pbBillable	[out, retval] A variant expression that indicates whether or not the activity is billable. If Empty, the parameter indicates that whether or not the activity is billable or is not specified.
bBillable	[in] A variant expression that indicates whether or not the activity is billable. If a Boolean expression, the parameter must be as follows: True: Activity is billable. False: Activity is not billable.

Return Values

S_OK	The operation succeeded.
E_INVALIDARG	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns/sets whether an activity is billable.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IActivityInfo" on page 202

IActivityInfo::ElapsedTime

The property returns or sets the amount of time in seconds the document was open for the edit or new version activity.

C++ Syntax

```
HRESULT ElapsedTime(long* plElapsedTime);
HRESULT ElapsedTime(long lElapsedTime);
```

Visual Basic Syntax

```
[ plElapsedTime = ] ActivityInfo.ElapsedTime
ActivityInfo.ElapsedTime [ = lElapsedTime ]
```

Parameters

plElapsedTime	[out, retval] A long that specifies the elapsed time in seconds.
lElapsedTime	[in] A long that specifies the elapsed time in seconds.

Return Values

S_OK	The operation succeeded.
E_INVALIDARG	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns/sets the amount of elapsed time in seconds while opening a document.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IActivityInfo" on page 202

IActivityInfo::Keystrokes

The property returns or sets the number of keystrokes in the edit or new version activity.

C++ Syntax

```
HRESULT Keystrokes(long* plKeystrokes);
HRESULT Keystrokes(long lKeystrokes);
```

Visual Basic Syntax

```
[ plKeystrokes = ] ActivityInfo.Keystrokes
ActivityInfo.Keystrokes [ = lKeystrokes ]
```

Parameters

plKeystrokes	[out, retval] A long that specifies the number of keystrokes.
lKeystrokes	[in] A long that specifies the number of keystrokes.

Return Values

S_OK	The operation succeeded.
E_INVALIDARG	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns/sets the number of activity keystrokes.

The value is always null if the Capture Keystrokes in Library Maintenance/System Parameters is set to "No".

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IActivityInfo" on page 202

IActivityInfo::Pages

The property returns or sets the number of pages printed for the print activity.

C++ Syntax

```
HRESULT Pages(long* plPages);
HRESULT Pages(long lPages);
```

Visual Basic Syntax

```
[ plPages = ] ActivityInfo.Pages
ActivityInfo.Pages [ = lPages ]
```

Parameters

plPages	[out, retval] A long that specifies the number of printed pages.
lPages	[in] A long that specifies the number of printed pages.

Return Values

S_OK	The operation succeeded.
E_INVALIDARG	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns/sets the number of pages printed for the print activity.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IActivityInfo" on page 202

IActivityInfo::RefDoc

The property returns or sets the source document in the copied from activity or the target document in the copied to activity.

C++ Syntax

```
HRESULT RefDoc(IProfile** ppDoc);

HRESULT RefDoc(IProfile* pDoc);
```

Visual Basic Syntax

```
[ Set ppDoc = ] ActivityInfo.RefDoc

Set ActivityInfo.RefDoc [ =  ]
```

Parameters

ppDoc	[out, retval] A Profile object that specifies the referenced document.
pDoc	[in] A Profile object that specifies the referenced document.

Return Values

S_OK	The operation succeeded.
E_INVALIDARG	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns/sets the source document in the copied from activity or the target document in the copied to activity.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IActivityInfo" on page 202

"IProfile" on page 419

IActivityInfo::TypeTime

The property returns or sets the time in seconds spent editing the document in the edit, create, or new version activity.

C++ Syntax

```
HRESULT TypeTime(long* plTime);

HRESULT TypeTime(long lTime);
```

Visual Basic Syntax

```
[ plTime = ] ActivityInfo.TypeTime

ActivityInfo.TypeTime [ = lTime ]
```

Parameters

plTime	[out, retval] A long that specifies the typing time in seconds.
lTime	[in] A long that specifies the typing time in seconds.

Return Values

S_OK	The operation succeeded.
E_INVALIDARG	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns/sets the time in seconds spent for editing the document in an activity.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IActivityInfo" on page 202

IActivityInfo::Typist

The property returns or sets the user who performed an activity.

C++ Syntax

```
HRESULT Typist(VARIANT* pvTypist);

HRESULT Typist(VARIANT vTypist);
```

Visual Basic Syntax

```
[ pvTypist = ] ActivityInfo.Typist

ActivityInfo.Typist [ = vTypist ]
```

Parameters

pvTypist	[out, retval] A string that specifies the user's name.
vTypist	[in] A string that specifies the user's name.

Return Values

S_OK	The operation succeeded.
E_INVALIDARG	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns/sets the user who performed an activity.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IActivityInfo" on page 202

IActivityInfo::Version

The property returns or sets the document version associated with the activity record.

C++ Syntax

```
HRESULT Version(VARIANT* pvVersion);

HRESULT Version(VARIANT vVersion);
```

Visual Basic Syntax

```
[ pvVersion = ] ActivityInfo.Version

ActivityInfo.Version [ = vVersion ]
```

Parameters

pvVersion	[out, retval] An expression that specifies the document version. Returns the version identifier.
vVersion	[in] An expression that specifies the document version. If a string expression, vVersion should specify the version identifier. If an object, vVersion should specify a valid Version object that identifies a document version.

Return Values

S_OK	The operation succeeded.
E_INVALIDARG	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns/sets the activity document version.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IActivityInfo" on page 202

"IVersion" on page 625

IAppForm

The AppForm object represents a form used for profiling documents created with a specific application. AppForm objects may be retrieved through the AppForms collection on an AppInfo object.

Properties

IAppForm Properties	Description
Default	Returns whether a form is default.
Description	Returns a description of a form.
Name	Returns the name of a form.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IAppForms" on page 217

"IAppInfo" on page 220

IAppForm::Default

The property returns whether or not a form is default.

C++ Syntax

```
HRESULT Default(VARIANT_BOOL* pbDefault);
```

Visual Basic Syntax

```
[ pbDefault = ] AppForm.Default
```

Parameters

pbDefault	[out, retval] Returns a Boolean value that specifies whether or not the form is default. True: Form is default. False: Form is not default.

Return Values

S_OK	The operation succeeded.
E_UNEXPECTED	The operation failed.

Remarks

Returns whether or not a form is default.

A default application form overrides all others for a particular application. When profiling documents in that application, only the override form will be displayed rather than a listing of available forms.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IAppForm" on page 213

IAppForm::Description

The property returns the description of a form.

C++ Syntax

```
HRESULT Description(BSTR* pbstrDescription);
```

Visual Basic Syntax

```
[ pbstrDescription = ] AppForm.Description
```

Parameters

pbstrDescription	[out, retval] Returns a string that specifies the form's description.

Return Values

S_OK	The operation succeeded.
E_UNEXPECTED	The operation failed.

Remarks

Returns a description of a form.

The following example displays the names and descriptions of all forms for Microsoft Word.

```
Dim App As New Application
 Dim objAppInfo As AppInfo, objFrm As AppForm
 Set objAppInfo = App.PrimaryLibrary.AppInformation("MS WORD")
 If Not objAppInfo Is Nothing Then
 For Each objFrm In objAppInfo.AppForms
 MsgBox "Name: " + objFrm.Name + Chr(13) + _
```

```
"Description: " + objFrm.Description
Next
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IAppForm" on page 213

IAppForm::Name

The property returns the short name of a form.

C++ Syntax

```
HRESULT Name(BSTR* pbstrName);
```

Visual Basic Syntax

```
[ pbstrName = ] AppForm.Name
```

Parameters

pbstrName	[out, retval] Returns a string that specifies the form's name.

Return Values

S_OK	The operation succeeded.
E_UNEXPECTED	The operation failed.

Remarks

Returns the name of a form.

The following example displays the names and descriptions of all forms for Microsoft Word.

```
Dim App As New Application
 Dim objAppInfo As AppInfo, objFrm As AppForm
 Set objAppInfo = App.PrimaryLibrary.AppInformation("MS WORD")
 If Not objAppInfo Is Nothing Then
 For Each objFrm In objAppInfo.AppForms
 MsgBox "Name: " + objFrm.Name + Chr(13) + _
 "Description: " + objFrm.Description
```

```
Next
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IAppForm" on page 213

IAppForms

The AppForms object represents a collection of forms used for profiling documents created with a specific application. The collection may be retrieved with the AppForms property on an AppInfo object.

Properties

IAppForms Properties	Description
Item	Returns a specific item of a collection either by position or by key.
Count	Returns the number of items in a collection.
_NewEnum	Returns an object that supports IEnumVARIANT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IAppInfo" on page 220

IAppForms::Count

The property returns the number of items in a collection.

C++ Syntax

```
HRESULT Count(long* plCount);
```

Visual Basic Syntax

```
[ plCount = ] AppForms.Count
```

Parameters

plCount	[out, retval] Returns a long that specifies the number of items in the collection.

Return Values

S_OK	The operation succeeded.
E_UNEXPECTED	The operation failed.

Remarks

Returns the number of items in a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IAppForms" on page 217

IAppForms::Item

The property returns a specific member of the collection either by position or by key.

C++ Syntax

```
HRESULT Item(VARIANT vIndex, IAppForm** ppAppForm);
```

Visual Basic Syntax

```
[ Set ppAppForm = ] AppForms.Item (vIndex)
```

Parameters

vIndex	[in] Required. An expression that specifies the position of a collection member. If a numeric expression, vIndex should be a number from 1 to the value of the collection's Count property. If a string expression, vIndex should correspond to a form's name.
ppAppForm	[out, retval] Returns an AppForm object that specifies the requested item.

Return Values

S_OK	The operation succeeded.
E_UNEXPECTED	The operation failed.

Remarks

Returns a specific item of the collection either by position or by key.

If the value provided as vIndex does not match any existing member of the collection, an error occurs. Item is the default property for a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IAppForm" on page 213

"IAppForms" on page 217

IAppForms::_NewEnum

The _NewEnum property returns the IEnumVARIANT interface on an object that can be used to enumerate the collection.

C++ Syntax

```
HRESULT _NewEnum(LPUNKNOWN* ppEnum);
```

Parameters

ppEnum	[out, retval] On return, an IEnumVARIANT object reference that can be used to enumerate the collection.

Remarks

Returns an object that supports IEnumVARIANT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IAppForms" on page 217

IAppInfo

The AppInfo object represents an application, the primary computer program that handles a document from a library. AppInfo objects may be retrieved through the AppInformation collection on a library.

Methods

IAppInfo Methods	Description
IAppInfo::CompatibleWith	Return whether an object is compatible with another application.
IAppInfo::ShowLaunchMethodChooserForm	Displays a form for selecting a launch method.

Properties

IAppInfo Properties	Description
IAppInfo::ID	Returns the identifier of an application.
IAppInfo::Description	Returns an application description.
IAppInfo::Disabled	Returns whether an application is disabled.
IAppInfo::FilingScheme	Returns the application filing scheme.
IAppInfo::PrintApp	Returns the application print identifier.
IAppInfo::ViewApp	Returns the application view identifier.
IAppInfo::DefaultExtension	Returns the default extension of an application.
IAppInfo::ShowsOnDesktop	Returns whether an application is shown on the Desktop.
IAppInfo::ValidOnProfile	Returns whether an application is valid on a profile.
IAppInfo::SupervisorApp	Returns whether an object is a supervisor application.
IAppInfo::UseUNC	Returns whether an application uses UNC.
IAppInfo::TimeOut	Returns the application timeout.
IAppInfo::LaunchMethods	Returns a collection of applications launch methods.
IAppInfo::Icon	Returns the application icon.
IAppInfo::AppForms	Returns a collection of application forms.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IAppInfo2" on page 240

"ILibrary" on page 360

IAppInfo::AppForms

The property returns a collection of profile forms for documents created with an application.

C++ Syntax

```
HRESULT AppForms(IAppForms** ppAppForms);
```

Visual Basic Syntax

```
[ Set ppAppForms = ] AppInfo.AppForms
```

Parameters

ppAppForms	[out, retval] Returns an AppForms object that specifies the collection of profile forms.

Return Values

S_OK	The operation succeeded.
E_FAIL	The operation failed.
E_POINTER	An output parameter should not be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns a collection of application forms.

Requirements

DM: Requires PowerDOCS 3.5.1 later.

Header: #import <fsplugin.dll>

See Also

"IAppInfo" on page 220

"IAppForms" on page 217

IAppInfo::CompatibleWith

The method checks whether or not a specific application is compatible with another application.

C++ Syntax

```
HRESULT CompatibleWith(VARIANT vApp, VARIANT_BOOL* pbCompatible);
```

Visual Basic Syntax

```
[ pbCompatible  = ] AppInfo.CompatibleWith (vApp)
```

Parameters

vApp	[in] Required. An expression that specifies the application to be checked for compatibility. If a string expression, vApp should specify the application identifier (e.g., "MS WORD"). If an object, vApp should specify a valid AppInfo object that identifies an application.
pbCompatible	[out, retval] Returns a Boolean value that specifies whether or not applications are compatible. True: Applications are compatible. False: Applications are not compatible.

Return Values

S_OK	The operation succeeded.
E_FAIL	The operation failed.
E_POINTER	An output parameter should not be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Return whether an object is compatible with another appication.

Compatible applications are typically software utilities that process files produced by some other application. For example, the user might run a redline or grammar checking application against a word processing

document. To do so, he/she would need to set up the word processor and the grammar checker as compatible.

In order to allow files to be converted from one application format to another, they must be set up as compatible. For example, if the user wish to import a WordPerfect document into Microsoft Word, Word must be set up to be a compatible application of WordPerfect.

Requirements

DM: Requires PowerDOCS 3.5.1 later.

Header: #import <fsplugin.dll>

See Also

"IAppInfo" on page 220

IAppInfo::DefaultExtension

The property returns the filename extension assigned to documents created with an application.

C++ Syntax

```
HRESULT DefaultExtension(BSTR* pbstrExt);
```

Visual Basic Syntax

```
[ pbstrExt = ] AppInfo.DefaultExtension
```

Parameters

pbstrExt	[out, retval] Returns a string that specifies the filename extension.

Return Values

S_OK	The operation succeeded.
E_FAIL	The operation failed.
E_POINTER	An output parameter should not be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the default extension of an application.

This property is valid only if the Enhanced of Unix-Compatible filing scheme are used.

Some applications require certain file extensions; others can manage with any extension you like. The extension is a portion of a DOS file name that follows a period.

Requirements

DM: Requires PowerDOCS 3.5.1 later.

Header: #import <fsplugin.dll>

See Also

"IAppInfo" on page 220

IAppInfo::Description

The property returns the full name of an application.

C++ Syntax

```
HRESULT Description(BSTR* pbstrDescription);
```

Visual Basic Syntax

```
[ pbstrDescription = ] AppInfo.Description
```

Parameters

pbstrDescription	[out, retval] Returns a string that specifies the application name.

Return Values

S_OK	The operation succeeded.
E_FAIL	The operation failed.
E_POINTER	An output parameter should not be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns an application description.

Requirements

DM: Requires PowerDOCS 3.5.1 later.

Header: #import <fsplugin.dll>

See Also

"IAppInfo" on page 220

IAppInfo::Disabled

The property returns whether or not an application is disabled.

C++ Syntax

```
HRESULT Disabled(VARIANT_BOOL* pbDisabled);
```

Visual Basic Syntax

```
[ pbDisabled = ] AppInfo.Disabled
```

Parameters

pbDisabled	[out, retval] Returns a Boolean value that specifies whether or not the application is disabled. True: Application is disabled. False: Application is not disabled.

Return Values

S_OK	The operation succeeded.
E_FAIL	The operation failed.
E_POINTER	An output parameter should not be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns whether an application is disabled.

If an application is disabled it cannot be used to create new documents.

Requirements

DM: Requires PowerDOCS 3.5.1 later.

Header: #import <fsplugin.dll>

See Also

"IAppInfo" on page 220

IAppInfo::FilingScheme

The property returns the method used to name documents created in an application.

C++ Syntax

```
HRESULT FilingScheme(short* psScheme);
```

Visual Basic Syntax

```
[ psScheme = ] AppInfo.FilingScheme
```

Parameters

psScheme	[out, retval] Returns an integer that specifies the naming method. It can be one of the following values: 1: Standard. The 7 digits of the filename are the document number, and the extension designates the document version and subversion number. 2: Enhanced. The first five characters of the filename designate the document number, the next 3 designate the version and subversion number, and the file extension designates the application. Enhanced filing scheme allows no vowels in any of the filenames, so words will not be randomly formed. 3: Unix-Compatible. This filing scheme is similar to the enhanced filing scheme except that special characters (such as # - !) have been removed.

Return Values

S_OK	The operation succeeded.
E_FAIL	The operation failed.
E_POINTER	An output parameter should not be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the application filing scheme.

Requirements

DM: Requires PowerDOCS 3.5.1 later.

Header: #import <fsplugin.dll>

See Also

"IAppInfo" on page 220

IAppInfo::Icon

The property returns the application's icon.

C++ Syntax

```
HRESULT Icon(VARIANT_BOOL bSmall, long* plIcon);
```

Visual Basic Syntax

```
[ plIcon = ] AppInfo.Icon (bSmall)
```

Parameters

bSmall	[in] Required. A Boolean value that indicates whether the small or large icon should be returned.
plIcon	[out, retval] Returns a long that specifies the HICON value of the application's icon.

Return Values

S_OK	The operation succeeded.
E_FAIL	The operation failed.
E_POINTER	An output parameter should not be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the application icon.

Requirements

DM: Requires PowerDOCS 3.5.1 later.

Header: #import <fsplugin.dll>

See Also

"IAppInfo" on page 220

IAppInfo::ID

The property returns the identifier of an application.

C++ Syntax

```
HRESULT ID(BSTR* pbstrID);
```

Visual Basic Syntax

```
[ pbstrID = ] AppInfo.ID
```

Parameters

pbstrID	[out, retval] Returns a string that specifies the application identifier, e.g., "WORD", "AMIPRO", or "L123".

Return Values

S_OK	The operation succeeded.
E_FAIL	The operation failed.
E_POINTER	An output parameter should not be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the identifier of an application.

The application identifier is unique throughout the library and uniquely identifies an application.

Requirements

DM: Requires PowerDOCS 3.5.1 later.

Header: #import <fsplugin.dll>

See Also

"IAppInfo" on page 220

IAppInfo::LaunchMethods

The property returns a collection of all application's launch methods.

C++ Syntax

```
HRESULT LaunchMethods(ILaunchMethods** ppMethods);
```

Visual Basic Syntax

```
[ Set ppMethods = ] AppInfo.LaunchMethods
```

Parameters

ppMethods	[out, retval] Returns a LaunchMethods object that represents the requested collection of launch methods.

Return Values

S_OK	The operation succeeded.
E_FAIL	The operation failed.
E_POINTER	An output parameter should not be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns a collection of applications launch methods.

Requirements

DM: Requires PowerDOCS 3.5.1 later.

Header: #import <fsplugin.dll>

See Also

"IAppInfo" on page 220

"ILaunchMethods" on page 354

IAppInfo::PrintApp

The property returns the identifier of the application used to print documents created with an application.

C++ Syntax

```
HRESULT PrintApp(BSTR* pbstrPrintApp);
```

Visual Basic Syntax

```
[ pbstrPrintApp = ] AppInfo.PrintApp
```

Parameters

pbstrPrintApp	[out, retval] Returns a string that specifies the print application identifier.

Return Values

S_OK	The operation succeeded.
E_FAIL	The operation failed.
E_POINTER	An output parameter should not be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the application print identifier.

Requirements

DM: Requires PowerDOCS 3.5.1 later.

Header: #import <fsplugin.dll>

See Also

"IAppInfo" on page 220

IAppInfo::ShowLaunchMethodChooserForm

The method displays a custom form for selecting a launch method and returns the launch method selected by the user.

C++ Syntax

```
HRESULT ShowLaunchMethodChooserForm(long lFlags, ILaunchMethod** ppMethod);
```

Visual Basic Syntax

```
[ Set ppMethod  = ] AppInfo.ShowLaunchMethodChooserForm (lFlags)
```

Parameters

lFlags	[in] Required. A long that specifies how the method should be executed. The parameter can be a combination of the values from the LaunchMethodChooserFlags enumeration.
ppMethod.	[out, retval] The method returns a LaunchMethod object that specifies the launch method selected by the user. If the user selected no launch method, the return value is NULL.

Return Values

S_OK	The operation succeeded.
E_FAIL	The operation failed.
E_POINTER	An output parameter should not be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Displays a form for selecting a launch method.

The method is called when there needs to display a form where the user can select a launch method. The dialog can be customized and its form should support the ILaunchMethodChooserForm interface. To display the form, the method calls up the ShowLaunchMethodChooserForm method on ILaunchMethodChooserForm.

Requirements

DM: Requires PowerDOCS 3.5.1 later.

Header: #import <fsplugin.dll>

See Also

"IAppInfo" on page 220

"ILaunchMethod" on page 331

"ILaunchMethodChooserForm" on page 885

IAppInfo::ShowsOnDesktop

The property returns whether or not an application's icon appears in the DM Extensions "Applications" folder.

C++ Syntax

```
HRESULT ShowsOnDesktop(VARIANT_BOOL* pbShown);
```

Visual Basic Syntax

```
[ pbShown = ] AppInfo.ShowsOnDesktop
```

Parameters

pbShown	[out, retval] Returns a Boolean value that specifies whether or not the application's icon appears in the application list. True: Application's icon appears in the application list. False: Application's icon does not appear in the application list.

Return Values

S_OK	The operation succeeded.
E_FAIL	The operation failed.
E_POINTER	An output parameter should not be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns whether an application is shown on the desktop.

Requirements

DM: Requires PowerDOCS 3.5.1 later.

Header: #import <fsplugin.dll>

See Also

"IAppInfo" on page 220

IAppInfo::SupervisorApp

The property returns whether or not an application is a supervisor application.

C++ Syntax

```
HRESULT SupervisorApp(VARIANT_BOOL* pbSupervisorApp);
```

Visual Basic Syntax

```
[ pbSupervisorApp = ] AppInfo.SupervisorApp
```

Parameters

pbSupervisorApp	[out, retval] Returns a Boolean value that specifies whether or not the application is a supervisor application. True: Application is a supervisor application. False: Application is not a supervisor application.

Return Values

S_OK	The operation succeeded.
E_FAIL	The operation failed.
E_POINTER	An output parameter should not be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns whether an object is a supervisor application.

A supervisor application can open documents created in any other application (using the application's own conversion filters), without need of making the applications compatible through DM Extensions. This feature is intended for applications that do not create their own documents, but use other applications' documents, i.e. Mail, Workflow, etc.

Requirements

DM: Requires PowerDOCS 3.5.1 later.

Header: #import <fsplugin.dll>

See Also

"IAppInfo" on page 220

IAppInfo::TimeOut

The property returns the maximum idle time, in minutes, before the editing time stops.

C++ Syntax

```
HRESULT TimeOut(short* psTimeout);
```

Visual Basic Syntax

```
[ psTimeout = ] AppInfo.TimeOut
```

Parameters

psTimeout	[out, retval] Returns an integer that specifies the application's timeout.

Return Values

S_OK	The operation succeeded.
E_FAIL	The operation failed.
E_POINTER	An output parameter should not be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the application timeout.

DM Extensions keeps track of how much time is spent for editing each document. The value the property returns is the maximum idle time before the timer stops counting. For example, suppose the timeout value is 10 minutes, and a user is editing a document. If the user leaves the building for an hour without exiting the document, after 10 minutes elapse, the timer will shut off. All 60 minutes the user is away will not be counted as edit time.

Any time a user switches to another Windows task the timer will automatically be suspended regardless of the timeout value. A value of 0 causes only the total elapsed time for each edit session to be recorded, idle time inclusive.

Requirements

DM: Requires PowerDOCS 3.5.1 later.

Header: #import <fsplugin.dll>

See Also

"IAppInfo" on page 220

IAppInfo::UseUNC

The property returns whether or not an application supports Universal Naming Convention (UNC).

C++ Syntax

```
HRESULT UseUNC(VARIANT_BOOL* pbUNC);
```

Visual Basic Syntax

```
[ pbUNC = ] AppInfo.UseUNC
```

Parameters

pbUNC	[out, retval] Returns a Boolean value that specifies whether or not the application supports UNC. True: Application supports UNC. False: Application does not support UNC.

Return Values

S_OK	The operation succeeded.
E_FAIL	The operation failed.
E_POINTER	An output parameter should not be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns whether an application uses UNC.

If an application supports UNC, DM Extensions will use the UNC name (i.e. \\Server\Volume) for this application instead of drive letters (i.e. E:).

Requirements

DM: Requires PowerDOCS 3.5.1 later.

Header: #import <fsplugin.dll>

See Also

"IAppInfo" on page 220

IAppInfo::ValidOnProfile

The property returns whether or not an application can be entered as an application on a profile form.

C++ Syntax

```
HRESULT ValidOnProfile(VARIANT_BOOL* pbValid);
```

Visual Basic Syntax

```
[ pbValid = ] AppInfo.ValidOnProfile
```

Parameters

pbValid	[out, retval] Returns a Boolean value that specifies whether or not the application can be entered on a profile form. True: Application can be entered on a profile form. False: Application cannot be entered on a profile form.

Return Values

S_OK	The operation succeeded.
E_FAIL	The operation failed.
E_POINTER	An output parameter should not be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns whether an application is valid on a profile.

Requirements

DM: Requires PowerDOCS 3.5.1 later.

Header: #import <fsplugin.dll>

See Also

"IAppInfo" on page 220

IAppInfo::ViewApp

The property returns the application identifier used to view documents created with an application.

C++ Syntax

```
HRESULT ViewApp(BSTR* pbstrViewApp);
```

Visual Basic Syntax

```
[ pbstrViewApp = ] AppInfo.ViewApp
```

Parameters

pbstrViewApp	[out, retval] Returns a string that specifies the view application identifier.

Return Values

S_OK	The operation succeeded.
E_FAIL	The operation failed.
E_POINTER	An output parameter should not be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the application view identifier.

Requirements

DM: Requires PowerDOCS 3.5.1 later.

Header: #import <fsplugin.dll>

See Also

"IAppInfo" on page 220

The interface inherits its definition from IAppInfo.

Properties

IAppInfo2 Properties	Description
FileTypes	Returns the application file types.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <fsplugin.dll>

See Also

"IAppInfo" on page 220

IAppInfo2::FileTypes

Returns the application file types.

C++ Syntax

```
HRESULT FileTypes(BSTR* pbstrFileTypes);
```

Visual Basic Syntax

```
[ pbstrFileTypes = ] AppInfo2.FileTypes
```

Parameters

pbstrFileTypes	[out, retval] Specifies an address, where a string with supported file types extensions should be returned.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns the application file types.

If returned value is S_FALSE methods of IOMErrInfo should be used for error description. Returned string contains supported file types extensions and descriptions in next format:
"FileExt1;FileExtDescription1, FileExt2; FileExtDescription2, …",

where FileExtX - file extension, FileExtDescriptionX - description for file extension X.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <fsplugin.dll>

See Also

"IAppInfo2" on page 240

"IOMErrInfo" on page 412

IApplication

The class identifies the Document Management plug-in. There is only one Application object. Creating the DOCSObjects.Application object will start the Document Management plug-in. No other the Document Management Objects properties or methods can be used until the plug-in is started.

When to Use

The Application object may be created with the CreateObject function or a Dim statement with the New keyword. The following example shows the Visual Basic syntax for creating a DOCSObjects.Application object:

```
Sub Form_Load()
 ...
 Dim App As Object
 ...
 ' Create the DOCSObjects.Application object
 Set App = CreateObject("DOCSObjects.Application")
 ...
 End Sub
```

Methods

IApplication Methods	Description
IApplication::ShowLibrariesForm	Displays a form for selecting search libraries.
IApplication::ShowProjectChooserForm	Displays a form for selecting projects.
IApplication::ShowDefaultSettings	Displays a default settings dialog.

IApplication Methods	Description
IApplication::ShowLibraryChooserForm	Displays a form for selecting a library.
IApplication::ShowTemplateChooserForm	Displays a form for selecting a document template.
IApplication::LaunchApplication	Launches an application.

Properties

IApplication Properties	Description
IApplication::Libraries	Returns all available libraries.
IApplication::CurrentLibrary	Returns the current library.
IApplication::PrimaryLibrary	Returns the primary library.
IApplication::DST	Returns a DST string.
IApplication::DefParentWindow	Returns/sets the default parent window handle.
IApplication::ShadowPathTemplate	Returns the template path to the Shadow directory.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IApplication2" on page 256

"IApplication3" on page 261

"IApplication4" on page 262

"IApplication5" on page 264

IApplication::CurrentLibrary

The property returns the current DM library.

C++ Syntax

```
HRESULT CurrentLibrary(ILibrary** ppCurrent);
```

Visual Basic Syntax

```
[ Set ppCurrent = ] Application.CurrentLibrary
```

Parameters

ppCurrent	[out, retval] Returns a Library object that specifies the current library that the user is presently accessing.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns the current library.

The following example sets the value of strCurLib to the current library name.

```
Dim App As New Application
 strCurLib = App.CurrentLibrary.Name
```

Requirements

DM: Requires PowerDOCS 3.5.1 later.

Header: #import <fsplugin.dll>

See Also

"IApplication" on page 241

"ILibrary" on page 360

IApplication::DefParentWindow

The property returns or sets the handle of a window that is used as the parent window when displaying modal dialogs.

C++ Syntax

```
HRESULT DefParentWindow(long* plwnd);
HRESULT DefParentWindow(long lwnd);
```

Visual Basic Syntax

```
[ plwnd = ] Application.DefParentWindow
Application.DefParentWindow [ = lwnd ]
```

Parameters

plWnd	[out, retval] A long that specifies the handle of a window.
lWnd	[in] A long that specifies the handle of a window.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns/sets the default parent window handle.

The property is used when displaying modal dialog windows to specify their parent window.

The following example sets a default parent window for all DMO forms to MyForm.

```
Dim App As New Application

 App.DefParentWindow = MyForm.hWnd
```

Requirements

DM: Requires PowerDOCS 3.5.1 later.

Header: #import <fsplugin.dll>

See Also

"IApplication" on page 241

IApplication::DST

The property returns a document security token (DST) string.

C++ Syntax

```
HRESULT DST(BSTR* pbstrDST);
```

Visual Basic Syntax

```
[ pbstrDST = ] Application.DST
```

Parameters

pbstrDST	[out, retval] Returns the DST string.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns a DST string.

The DST is a unique string returned from the DM Server, and is valid while the user is logged into the server. All subsequent transactions, such as search requests and lookups, require the token to verify that the user can perform the transaction.

The following example logs into the DM Server and sets the value of strDST to the DST that was returned from the DM Server.

```
Dim App As New Application
 Dim strDST As String
 strDST = App.DST
```

Requirements

DM: Requires PowerDOCS 3.5.1 later.

Header: #import <fsplugin.dll>

See Also

"IApplication" on page 241

IApplication::LaunchApplication

The method launches the specified application.

C++ Syntax

```
HRESULT LaunchApplication(IAppInfo* pAppInfo);
```

Visual Basic Syntax

```
Application.LaunchApplication (pAppInfo)
```

Parameters

pAppInfo	[in] Required. An AppInfo object that specifies an application to be launched.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Launches an application.

The AppInfo object used as the parameter should specify a valid application on the primary library.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IApplication" on page 241

IApplication::Libraries

The property returns a collection of all libraries available to the user.

C++ Syntax

```
HRESULT Libraries(ILibraries** ppLibraries);
```

Visual Basic Syntax

```
[ Set ppLibraries = ] Application.Libraries
```

Parameters

ppLibraries	[out, retval] Returns a Libraries object that represents the requested collection of libraries.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns all available libraries.

The property returns a collection of all the libraries available to the user including the primary library and remote ones. It is called when necessary to retrieve these libraries, e.g. when displaying all the available libraries in a form.

A library would not appear in the collection, if it is not included in the remote libraries of the primary library.

The following example enumerates all the libraries available to the user.

```
Dim App As New Application
 Dim Libs As Libraries
 Dim Lib As Library

 Set Libs = App.Libraries
 For i = 1 To Libs.Count
 Set Lib = LibColl(i)
 ' Some operations with the library
 ...
 Next
```

Requirements

DM: Requires PowerDOCS 3.5.1 later.

Header: #import <fsplugin.dll>

See Also

"IApplication" on page 241

"ILibraries" on page 357

IApplication::PrimaryLibrary

The property returns the primary library the user logged into.

C++ Syntax

```
HRESULT PrimaryLibrary([out, ILibrary** ppPrimary);
```

Visual Basic Syntax

```
[ Set ppPrimary = ] Application.PrimaryLibrary
```

Parameters

ppPrimary	[out, retval] Returns a Library object that specifies the primary library.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns the primary library.

The following example sets the value of strPrimaryLib to the current library name.

```
Dim App As New Application
 strPrimaryLib = App.PrimaryLibrary.Name
```

Requirements

DM: Requires PowerDOCS 3.5.1 later.

Header: #import <fsplugin.dll>

See Also

"IApplication" on page 241

"ILibrary" on page 360

IApplication::ShadowPathTemplate

The property returns the template path to the Shadow directory including a drive letter.

C++ Syntax

```
HRESULT ShadowPathTemplate(BSTR* pbstrTemplate);
```

Visual Basic Syntax

```
[ pbstrTemplate = ] Application.ShadowPathTemplate
```

Parameters

pbstrTemplate	[out, retval] Returns a string that specifies the directory structure where shadowed documents are located on the client's local drive.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns the template path to the Shadow directory.

Shadowing a document is a feature of DM Extensions that allows the user to work with a local copy of a document rather than a network copy. This cuts down on network traffic and allows the user to keep working on a document if the network fails. The returned directory structure consists of a drive letter and path that may be specified using variables.

The following example sets the value of strPath to the shadow template path.

```
Dim App As New Application
 strPath = App.ShadowPathTemplate
```

Requirements

DM: Requires PowerDOCS 3.5.1 later.

Header: #import <fsplugin.dll>

See Also

"IApplication" on page 241

IApplication::ShowDefaultSettings

The method displays the default settings dialog box.

C++ Syntax

```
HRESULT ShowDefaultSettings(short* psResult);
```

Visual Basic Syntax

```
[ psResult  = ] Application.ShowDefaultSettings
```

Parameters

psResult	[out, retval] Returns an integer that specifies user's action on the dialog. For possible values see FormsResults Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Displays a default settings dialog.

The following example displays the default settings dialog.

```
Dim App As New Application
 App.DefParentWindow = MyForm.hWnd

 If App.ShowDefaultSettings = FR_DATACHANGED Then
 ' The user changed the settings
 ...
 End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IApplication" on page 241

"FormsResults Enumeration" on page 686

IApplication::ShowLibrariesForm

The method displays a custom form for selecting search libraries.

C++ Syntax

```
HRESULT ShowLibrariesForm(short* psResult);
```

Visual Basic Syntax

```
[ psResult  = ] Application.ShowLibrariesForm
```

Parameters

psResult	[out, retval] Returns an integer that is reserved for future use and should be ignored.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Displays a form for selecting search libraries.

The method is called when there needs to display a form for selecting libraries searchable by default and for setting the current library. The dialog can be customized and its form should support the ILibrariesForm interface. To display the form, the method calls up the ShowLibrariesForm method on ILibrariesForm.

The following example displays a form for selecting search libraries.

```
Dim App As New Application


 If App.ShowLibrariesForm = FR_DATACHANGED Then
 ' The user changed either the current library or
 ' the libraries searchable by default
 ...
 End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IApplication" on page 241

"ILibrariesForm" on page 886

IApplication::ShowLibraryChooserForm

The method displays a custom form for selecting a library and returns the library selected by the user.

C++ Syntax

```
HRESULT ShowLibraryChooserForm(ILibrary** ppLibrary);
```

Visual Basic Syntax

```
[ Set ppLibrary  = ] Application.ShowLibraryChooserForm
```

Parameters

ppLibrary	[out, retval] Returns a Library object that specifies the library selected by the user. If the user selected no library by pressing a Cancel button, the return value is NULL.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Displays a form for selecting a library.

The method is called when there needs to display a form where the user can select a library. The dialog can be customized and its form should support the ILibraryChooserForm interface. To display the form, the method calls up the LibraryChooserForm method on ILibraryChooserForm.

The following example displays a form for selecting a library and sets the Lib value to the library the user selected in the dialog.

```
Dim App As New Application
 Dim Lib As Library

 Set Lib = App.ShowLibraryChooserForm
 If Not Lib Is Nothing Then
 ' Some operations with the selected library
 ...
 End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IApplication" on page 241

"ILibrary" on page 360

"ILibraryChooserForm" on page 887

IApplication::ShowProjectChooserForm

The method displays a Quick Retrieve form for selecting a Folder (or Folders if multiple selections are allowed) and returns a collection of the projects selected by the user.

C++ Syntax

```
HRESULT ShowProjectChooserForm(short sFlags, IProjects**
ppSelected);
```

Visual Basic Syntax

```
[ Set ppSelected  = ] Application.ShowProjectChooserForm (sFlags)
```

Parameters

sFlags	[in] Required. An integer value that specifies the form appearance. The parameter can be a combination of the values from ProjectChooserFlags enumeration.
ppSelected	[out,retval] Returns a Projects object that specifies the collection of all the folders selected by the user. If the user selected no folders, the return value is NULL.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Displays a form for selecting folders.

The method is called when there needs to display a form where the user can select one or more projects. The dialog can be customized and its form should support theIProjectChooserForm interface. To display the form, the method calls up the ShowProjectChooserForm method on IProjectChooserForm.

The following example adds all the folders selected by the user to another folder.

```
Dim App As New Application
 Dim Proj, Subproj As Project
 Dim Projs As Projects

 ' Select a folder sub-folders will be added to
 Set Projs = App.ShowProjectChooserForm(0)
 If Not Projs Is Nothing Then
 Set Proj = Projs(1)

 ' Select sub-folders that will be added to Proj
 Set Projs = App.ShowProjectChooserForm( _
 PCF_ALLOWMULTISEL)
 If Not Projs Is Nothing Then
 For Each Subproj In Projs
 Proj.AddSubProject Subproj
 Next
 End If
 End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IApplication" on page 241

"IProjects" on page 534

"IProjectChooserForm" on page 917

IApplication::ShowTemplateChooserForm

The method displays a custom form for selecting a template and returns the template profile selected by the user.

C++ Syntax

```
HRESULT ShowTemplateChooserForm(IProfile** ppProfile);
```

Visual Basic Syntax

```
[ Set ppProfile  = ] Application.ShowTemplateChooserForm
```

Parameters

ppProfile	[out, retval] Returns a Profile object that specifies the template selected by the user. If the user selected no template by pressing a Cancel button, the return value is NULL.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Displays a form for selecting a document template.

The method is called when there needs to display a form where the user can select a template, for instance, to create a new document. The dialog can be customized and its form should support the ITemplateChooserForm interface. To display the form, the method calls up the SelectTemplate method on ITemplateChooserForm.

The following example displays a form for selecting a template and sets the Template value to the profile the user selected in the dialog.

```
Dim App As New Application
 Dim Template As Profile


 Set Template = App.ShowTempateChooserForm
 If Not Template Is Nothing Then
 ' Some operations with the selected template
 ...
 End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IApplication" on page 241

"IProfile" on page 419

"ITemplateChooserForm" on page 951

IApplication2

The interface is implemented by the main Application object and inherits its definition from IApplication. This interface identifies the Document Management plug-in implemented by a singleton object. Creating this object will start the Document Management plug-in. No other the Document Management Objects properties or methods can be used until the plug-in is started.

The interface inherits its definition from IApplication, so its implementation should also include all properties and methods of IApplication.

Methods

IApplication2 Methods	Description
IApplication2::Login	Logs the user into the DM Server.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IApplication" on page 241

"IApplication3" on page 261

"IApplication4" on page 262

IApplication2::Login

Logs the user into the DM Server using supplied authentication information.

C++ Syntax

```
HRESULT Login(
BSTR bstrLibrary,       //A DM library name.  BSTR bstrUserName,
//User name  BSTR bstrPassword,    //Password  VARIANT
vNetworkName, //Network name  BSTR* pbstrDST       //Address of
output variable that receives the
                        //document security token (DST) string);
```

Visual Basic Syntax

```
[ pbstrDST = ] Application2.Login (bstrLibrary, bstrUserName,
bstrPassword, vNetworkName)
```

Parameters

bstrLibrary	[in] An address of a Basic wide character string (two bytes per character) to contain the name of a DM library the user will be logged into.
bstrUserName	[in] An address of a Basic wide character string to contain a user name that will be used to log into the DM Server. It can be either a user identifier in the DM library or a network user name if the user has a LAN account that DM Server can verify (e.g., a NT Domain user name).
bstrPassword	[in] An address of a Basic wide character string to contain a user password that will be used to log into the DM Server. It can be either a user password in the DM library or a LAN account password if bstrUserName specifies a LAN account.

vNetworkName	[in, optional] Specifies a network name that should be used to log into the DM Server server. It can be: • Variant data of the type VT_ERROR (DISP_E_PARAMNOTFOUND) or an address of a blank Basic wide character string (converted to VARIANT) if a DM library User ID is specified as bstrUserName. • An address of a Basic wide character string to contain the name of the NT Domain if an NT Domain LAN account is specified as bstrUserName. • An address of a Basic wide character string to contain the name of the Novell NetWare server if a Novell NetWare version 3 LAN account is specified as bstrUserName. • An address of a Basic wide character string to contain the name of the Novell NetWare tree if a Novell NetWare version 4 or 5 LAN account is specified as bstrUserName.

pbstrDST	[out, retval] An address of the BSTR pointer variable that receives a pointer to a DM Security Token (DST) string. Upon successful return, this string is a Basic wide character string containing the unique DST returned from the DM Server, and is valid while the user is logged into the server. All subsequent transactions, such as search requests and lookups, require the token to verify that the user can perform the transaction. The implementation should allocate the string returned in pbstrDST, and the caller is responsible for freeing it. If an error occurs, the implementation should set *pbstrDST to NULL.

Return Values

S_OK	The user logged successfully into the DM Server.
S_FALSE	The user did not log into the DM Server because the connection had been established (DST of that connection is returned and the OM_INVALID_OBJECT_STATE error code is set) or the DM Server returned an error (the OM_FUSION_ERROR error code is set). IOMErrInfo::ErrText should return a DM Server error description.

Remarks

Logs the user into the DM Server.

The method should be used to log the user into the DM Server without presenting any login dialogs in the completely UI-less mode. The implementation should return all error information (if any) through the IOMErrInfo interface. The caller is responsible for displaying the appropriate error message.

The user cannot log into the DM Server twice. If the connection has been established, all subsequent calls to Login will return the OM_INVALID_OBJECT_STATE error with description "Connection to the DM Server has been established.".

When successful, the method should set IApplication::DST to the DST string returned.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IApplication2" on page 256

"IOMErrInfo" on page 412

IApplication3

The interface is implemented by the main Application object and inherits its definition from IApplication2.

Properties

IApplication3 Properties	Description
Mode	Indicates in which mode the user is currently working.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IApplication" on page 241

"IApplication2" on page 256

"IApplication4" on page 262

IApplication3::Mode

Indicates in which mode the user is currently working.

C++ Syntax

```
HRESULT Mode(long* pnMode);
```

Visual Basic Syntax

```
[ pnMode = ] Application3.Mode
```

Parameters

pnMode	[out, retval] The current mode. For possible values see Modes Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Indicates in which mode the user is currently working.

Requirements

DM: Requires PowerDOCS 3.5.1 or later.

Header: #import <fsplugin.dll>

See Also

"IApplication3" on page 261

IApplication4

The interface is implemented by the main Application object and inherits its definition from IApplication3.

Methods

IApplication4 Methods	Description
IApplication4::LoginCAP	Logs the user into the DM Server using CAP Authentication Token.
IApplication4::DisplayMessage	Displays a message.

Requirements

DM: Requires DM Extensions 5.0 later.

Header: #import <fsplugin.dll>

See Also

"IApplication" on page 241

"IApplication2" on page 256

"IApplication3" on page 261

IApplication4::DisplayMessage

Displays a message.

C++ Syntax

```
HRESULT DisplayMessage(BSTR bstrMessage, long nType);
```

Visual Basic Syntax

```
Application4.DisplayMessage (bstrMessage, nType)
```

Parameters

bstrMessage	[in] A message to be displayed.
nType	[in] A message box type. For possible values see DisplayMessageTypes Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Displays a message.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IApplication4" on page 262

IApplication4::LoginCAP

Logs the user into the DM Server using CAP Authentication Token.

C++ Syntax

```
HRESULT LoginCAP(
BSTR bstrLibrary, BSTR bstrCAPToken, BSTR* pbstrDST);
```

Visual Basic Syntax

```
[ pbstrDST = ] Application4.LoginCAP (bstrLibrary, bstrCAPToken)
```

Parameters

bstrLibrary	[in] A library name
bstrCAPToken	[in] CAP authentication token
pbstrDST	[out, retval] DST that received from DM Server if logon was successful.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Logs the user into the DM Server using CAP Authentication Token.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IApplication4" on page 262

IApplication5

The interface is implemented by the main Application object and inherits its definition from IApplication4.

Methods

IApplication5 Methods	Description
IApplication5::GetTerminology	Extracts terminology information from the login library and returns a Terminology object.

Requirements

DM: Requires DM Extensions 5.1 later.

Header: #import <fsplugin.dll>

See Also

"IApplication" on page 241

"IApplication2" on page 256

"IApplication3" on page 261

"IApplication4" on page 262

IApplication5::GetTerminology

Extracts Terminology information from the login library and returns a Terminology object.

C++ Syntax

```
HRESULT GetTerminology(ITerminology** ppTerminology );
```

Visual Basic Syntax

```
[ ppTerminology = ] Application5.GetTerminology
```

Parameters

ppTerminology	[out, retval] Returns a Terminology object that can be used in string, window, or menu text. Sub-strings of the form "%[Term]%" will be replaced with the corresponding value from the TERMINOLOGY table for the specified term entry. Term information is always extracted from the primary library.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.1 or later.

Header: #import <fsplugin.dll>

See Also

"IApplication5" on page 264

IApplications

The Applications object represents a collection of all registered applications on a library. The collection may be retrieved with the AppInformation property on a library.

Properties

IApplications Properties	Description
Item	Returns a specific item of a collection either by position or by key.
Count	Returns the number of items in a collection.
_NewEnum	Returns an object that supports IEnumVARIANT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

IApplications::Count

The property returns the number of items in a collection.

C++ Syntax

```
HRESULT Count(long* plCount);
```

Visual Basic Syntax

```
[ plCount = ] Applications.Count
```

Parameters

plCount	[out, retval] Returns a long that specifies the number of items in a collection.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns the number of items in a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IApplications" on page 266

IApplications::Item

The property returns a specific member of a collection either by position or by key.

C++ Syntax

```
HRESULT Item(VARIANT vIndex, IAppInfo** ppAppInfo);
```

Visual Basic Syntax

```
[ Set ppAppInfo = ] Applications.Item (vIndex)
```

Parameters

vIndex	[in] Required. An expression that specifies the member position in the collection. If a numeric expression, vIndex should be a number from 1 to the value of the collection's Count property. If a string expression, vIndex should correspond to an application identifier.
ppAppInfo	[out, retval] Returns an AppInfo object that specifies the requested item.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns a specific item of a collection either by position or by key.

If the value provided as vIndex does not match any existing member of the collection, an error occurs. Item is the default property for a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IApplications" on page 266

"IAppInfo" on page 220

IApplications::_NewEnum

The _NewEnum property returns the IEnumVARIANT interface on an object that can be used to enumerate the collection.

C++ Syntax

```
HRESULT _NewEnum(LPUNKNOWN* ppEnum);
```

Parameters

ppEnum	[out, retval] On return, an IEnumVARIANT object reference that can be used to enumerate the collection.

Remarks

Returns an object that supports IEnumVARIANT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IApplications" on page 266

IColumn

The Column object represents a field (column) on a profile or search form.

The Column object inherits its definition from the IOMErrInfo interface so its implementation also includes all the properties of IOMErrInfo in addition to the ones described below.

Methods

IColumn Methods	Description
IColumn::ClearValue	Clears the column value.
IColumn::DoLookup	Opens a lookup form associated with a column.

Properties

IColumn Properties	Description
IColumn::Name	Returns the name of a column.
IColumn::Type	Returns the type of a column.
IColumn::Value	Returns/sets the value of a column.
IColumn::MVFValue	Returns the value of a multi-valued column.
IColumn::SQLName	Returns the SQL name of a column.
IColumn::HasValidValue	Returns whether the column value is valid.
IColumn::HasLookup	Returns whether a column has a lookup form.
IColumn::ReadOnly	Returns whether a column is read-only.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IColumn2" on page 281

"IOMErrInfo" on page 412

IColumn::ClearValue

The method sets the value of a column to null.

C++ Syntax

```
HRESULT ClearValue();
```

Visual Basic Syntax

```
Column.ClearValue
```

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Clears the column value.

If ClearValue is used on a required column, a new value must be entered before the changes can be saved to the database.

The column cannot be assigned null if the column value cannot be modified, i.e. if its ReadOnly property returns True.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IColumn" on page 269

IColumn::DoLookup

The method displays a lookup form associated with a column, validates the column value, or auto-completes it.

C++ Syntax

```
HRESULT DoLookup(
long lParentWindow, short sMode, IBCCoreCtx** ppCtx);
```

Visual Basic Syntax

```
[ Set ppCtx  = ] Column.DoLookup (lParentWindow, sMode)
```

Parameters

lParentWindow	[in] Required. A long that specifies the handle of a window that will be used as the parent window for the lookup form.
sMode	[in] Required. An integer value that specifies how the method should be executed. The parameter can be one of the OMFlagsLookUp enumeration values.
ppCtx	[out, retval] Returns a BCCoreCtx object that specifies which columns have changed. The context object contains the key ChangedColumns that lists the names of all columns that have to be changed as a result of calling DoLookup. For each name in ChangedColumns there exists a registered string with a key being the column name and the value is equal to its new value (assigned by the lookup).

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Opens a lookup form associated with a column.

The method does nothing if the column has no associated lookup. The method does not change column values. The developer has to change them manually using the Value property.

The lookup form can be customized and should support the ILookupForm interface. To display the form, the method calls up the DoLookup method on ILookupForm.

The following example displays a lookup for the column "AUTHOR_ID" on the profile with number 20 on the primary library.

```
Dim App As New Application
 Dim Prof As Profile, Col As Column
```

```
Set Prof = App.PrimaryLibrary.GetProfile(20)
If Not Prof Is Nothing Then
Set Col = Prof.Columns("AUTHOR_ID")
If Col.HasLookup Then
Col.DoLookup 0, DL_SHOW_LOOKUP
End If
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IColumn" on page 269

"IBCCoreCtx" on page 46

"ILookupForm" on page 890

IColumn::HasLookup

The property returns whether a column has an associated lookup form.

C++ Syntax

```
HRESULT HasLookup(VARIANT_BOOL* pbLookup);
```

Visual Basic Syntax

```
[ pbLookup = ] Column.HasLookup
```

Parameters

pbLookup	[out, retval] Returns a Boolean value that specifies whether the column has a lookup. True: Column has a lookup. False: Column does not have a lookup.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns whether a column has a lookup form.

If the column has an associated lookup form, its value can be obtained or validated by browsing this lookup.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IColumn" on page 269

IColumn::HasValidValue

The property returns whether the column value is valid.

C++ Syntax

```
HRESULT HasValidValue(VARIANT_BOOL* pbValid);
```

Visual Basic Syntax

```
[ pbValid = ] Column.HasValidValue
```

Parameters

pbValid	[out, retval] Returns a Boolean value that specifies whether the column value is valid. True: Column has been assigned a valid value. False: Column has not been assigned a valid value.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns whether the column value is valid.

The property will make sure that any required fields are not null as well as validate any dependencies on other tables.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IColumn" on page 269

IColumn::MVFValue

The property returns the value of a multi-valued column.

C++ Syntax

```
HRESULT MVFValue(BSTR* pbstrValue);
```

Visual Basic Syntax

```
[ pbstrValue = ] Column.MVFValue
```

Parameters

pbstrValue	[out, retval] Returns a string that specifies the multi-valued column. Different values are separated by a semicolon.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns the value of a multi-valued column.

Use this property instead of Value to get all values of a multi-valued field. The Value property may return a truncated string.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IColumn" on page 269

IColumn::Name

The property returns the name of a column.

C++ Syntax

```
HRESULT Name(BSTR* pbstrName);
```

Visual Basic Syntax

```
[ pbstrName = ] Column.Name
```

Parameters

pbstrName	[out, retval] Returns a string that specifies the column name.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns the name of a column.

The name can be used to identify the object in a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IColumn" on page 269

IColumn::ReadOnly

The property returns whether the column value can be changed.

C++ Syntax

```
HRESULT ReadOnly(VARIANT_BOOL* pbReadOnly);
```

Visual Basic Syntax

```
[ pbReadOnly = ] Column.ReadOnly
```

Parameters

pbReadOnly	[out, retval] Returns a Boolean value that specifies whether the column is read-only. True: Column is read-only. False: Column is not read-only.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns whether a column is read-only.

If the column is read-only, it cannot be assigned a value by the Value property.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IColumn" on page 269

IColumn::SQLName

The property returns the name of the SQL column associated with a form's field.

C++ Syntax

```
HRESULT SQLName(BSTR* pbstrSQLName);
```

Visual Basic Syntax

```
[ pbstrSQLName = ] Column.SQLName
```

Parameters

pbstrSQLName	[out, retval] Returns a string that specifies the SQL column name.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns the SQL name of a column.

The following example retrieves the SQL name of the column "AUTHOR_ID" on the profile with number 20 on the primary library.

```
Dim App As New Application
 Dim Prof As Profile, Col As Column
 Set Prof = App.PrimaryLibrary.GetProfile(20)
 If Not Prof Is Nothing Then
 Set Col = Prof.Columns("AUTHOR_ID")
 If Not Col Is Nothing Then
 MsgBox Col.SQLName
 End If
 End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IColumn" on page 269

IColumn::Type

The property returns or sets the type of data contained in a column.

C++ Syntax

```
HRESULT Type(short* psType);
```

Visual Basic Syntax

```
[ psType = ] Column.Type
```

Parameters

psType	[out, retval] Returns an integer that specifies the column type. It can be one of the values from ColumnTypes enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns the type of a column.

The column types are defined through DM and not through the database engine even though the database may have similar column types. This frees the programmer from worrying about the various data types on different database servers, since DM handles the data types.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IColumn" on page 269

IColumn::Value

The property returns or sets the value contained in a column.

C++ Syntax

```
HRESULT Value(VARIANT* pvValue);
HRESULT Value(VARIANT vValue);
```

Visual Basic Syntax

```
[ pvValue = ] Column.Value
Column.Value [ = vValue ]
```

Parameters

pvValue	[out, retval] An expression that specifies the column value.
vValue	[in] An expression that specifies the column value. The type of this parameter should conform to the column type returned by the Type property.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns/sets the value of a column.

The column value can not be changed if the column is read-only, i.e. if its Read-Only property returns True.

The Value property may return a truncated result for a multi-valued column. To get all values of a multi-valued column, use the MVFValue property instead.

The following example changes the name of the profile with number 20.

```
Dim App As New Application
 Dim Prof As Profile
 Set Prof = App.PrimaryLibrary.GetProfile(20)
 If Not Prof Is Nothing Then
 If Prof.SelectForEdit(OF_NORMAL + SFE_LOCK_PROFILE) = _
 SFER_OK Then
 Prof.Columns("$DOCNAME").Value = "New Name"
 Prof.Save OF_NORMAL
 Prof.ReleaseDoc
 If Prof.Name <> "New Name" Then
 MsgBox "Document name hasn't changed"
 End If
 End If
 End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IColumn" on page 269

IColumn2

The interface is implemented by the column object, inherits its definition from IColumn, and represents a field (column) on a profile or search form.

Properties

IColumn2 Properties	Description
Properties	Returns the column's attributes.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IColumn" on page 269

IColumn2::Properties

The property returns a collection of column's attributes specified by the user in the DM Designer.

C++ Syntax

```
HRESULT Properties(IProperties** ppProps)
```

Visual Basic Syntax

```
[ Set ppProps = ] Column2.Properties
```

Parameters

ppProps	[out, retval] An address of IProperties* pointer variable that receives the interface pointer to the collection of persistent properties. Each time the object receives a successful call to Properties, it should increase the reference count on the *ppProps pointer the property returns. It is the caller's responsibility to call IUnknown::Release when it is done with the pointer. If an error occurs, the implementation should set *ppProps to NULL

Return Values

S_OK	A collection was returned.
E_OUTOFMEMORY	There is not enough memory to complete the operation.
E_UNEXPECTED	An unexpected error occurred.
E_INVALIDARG	ppProps is a NULL pointer. The OM_INVALID_ARGUMENTS error code should be also set.

Remarks

Returns the column's attributes.

The property is called when there needs to get a collection of column's attributes from the column definition.

To get a specific Property object, the developer can use the standard IProperties::Item property of the collection. The following column's attributes are supported:

Name	Type	Access	Description
MaxChars	DT_Long	r/o	The maximum number of characters that can be entered or displayed for the column value. The attribute is supported only if the IColumn type is CT_VARCHAR. Otherwise, IProperties::Item should return CTL_E_INVALIDPROPERTY VALUE.
Required	DT_Boolean	r/o	Indicates whether the column is required as follows: VARIANT_TRUE: Column is required. VARIANT_FALSE: Column is not required. If the column is required, the column cannot be left blank and should have a value before the Profile or Maintenance record can be saved.
Flags	DT_Long	r/o	Returns column flags.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IColumn2" on page 281

"IProperty" on page 551

"IProperties" on page 546

IColumns

The Columns object represents a collection of columns that can be accessed through a profile or the search criteria form of a search. The collection may be retrieved with the Columns property on a document's profile or search.

Methods

IColumns Methods	Description
IColumns::SetColumnDefaults	Sets default values of the columns in a collection.

Properties

IColumns Properties	Description
IColumns::Item	Returns a specific item of a collection either by position or by key.
IColumns::Count	Returns the number of items in a collection.
IColumns::_NewEnum	Returns an object that supports IEnumVARIANT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

IColumns::Count

The property returns the number of items in a collection.

C++ Syntax

```
HRESULT Count(long* plCount);
```

Visual Basic Syntax

```
[ plCount = ] Columns.Count
```

Parameters

plCount	[out, retval] Returns a long that specifies the number of items in a collection.

Return Values

S_OK	The operation succeeded.
E_POINTER	An output parameter should not be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the number of items in a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IColumns" on page 285

IColumns::Item

The property returns a specific member of a collection either by position or by key.

C++ Syntax

```
HRESULT Item(VARIANT vIndex, IColumn** ppColumn);
```

Visual Basic Syntax

```
[ Set ppColumn = ] Columns.Item (vIndex)
```

Parameters

vIndex	Required. An expression that specifies the member position in a collection. If a numeric expression, vIndex should be a number from 1 to the value of the collection's Count property. If a string expression, vIndex should correspond to a column name. A Profile object support hidden columns that can be accessed only by column name. These columns are not located on a profile form and can be used to obtain Profile object properties. Here is list of these columns:

Name	Return value type	Access	Description
CUTrustee	VT_DISPATCH	r/o	An IProfileTrustee object that contains access rights for the current object. If a document is not secured this object contains full rights.
%FormattedStatus	VT_BSTR	r/o	A document status in a formatted string (“Available”, “Checked out”,…).
%CUCheckoutVersion	VT_BSTR	r/o	A label of version that has been checked-out by the current user. If no version of a document has been checked-out by the current user this column returns an empty value.
%CheckLocked	VT_I4	r/o	Returns the document lock status: 0 – a document is not locked 2 – a document is locked by the current user 1 – a document is locked by other user.

Name	Return value type	Access	Description
%GetCopy	VT_DISPATCH	r/o	Returns a copy of the profile object. When the Save method will be called on returned copied object an original Profile object will be updated too.
%ProfileSystemID	VT_I4	r/o	A document SYSTEM_ID in the PROFILE table
%GetProfileLockInfo	VT_ARRAY | VT_BSTR	r/o	An array that contains lock information of a document: • a document status (number); • an information string ("Document is locked" or "Document is being edited"); • a check-out date; • an estimate check-in date; • a check-out typist name; • a check-out typist phone; • a check-out typist extension; • check-out comments .
%RefreshAll	None	r/o	Refreshes all document properties.
%RefreshVersions	None	r/o	Refreshes the document version list.
CheckSelectForEdit	VT_I4	r/o	Checks if a document is selected for editing: 0 – a document was not selected for editing; 1 – a document was selected for editing.
%FetchStatus	VT_I4	r/o	Returns a document status. A document status will be read from the database even if it was cached.
%ReferenceFileName	VT_BSTR	r/o	Returns a reference file name (DRF file) for a document.

Name	Return value type	Access	Description
%ApplyToAll	VT_I4	r/w	The “Apply to all” status. 1 – the “Apply to all” check-box was selected when the profile form was edited; 0 – the “Apply to all” check-box was cleared.
%CreateCopy	VT_DISPATCH	r/o	Creates a new Profile object and copies all properties from the current Profile object to new one.
%MassUpdateProfile	VT_I4	r/w	Indicates whether Profile object is used for mass update profiles operation. 1 - Profile object is used for mass update operation. 0 - Profile object is not used for mass update operation.
%UpdateTrusteesType	VT_I4	r/w	Indicates update trustees option when Profile object is used for mass update profiles operation. 1 - Trustees should be added to updated profiles. 2 - Trustees should be removed from updates profiles. 3 - Trustees should be replaced in updated profiles.

ppColumn	[out, retval] Returns an Column object that specifies the requested item.

Return Values

S_OK	The operation succeeded.
E_POINTER	An output parameter should not be NULL.
E_INVALIDARG	Invalid argument passed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns a specific item of a collection either by position or by key.

If the value provided as vIndex does not match any existing member of a collection, an error occurs. Item is the default property for a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IColumns" on page 285

"IProfile" on page 419

"IProfileTrustee" on page 474

"IColumn" on page 269

IColumns::SetColumnDefaults

The method sets the values of all the columns from a collection to their defaults.

C++ Syntax

```
HRESULT SetColumnDefaults();
```

Visual Basic Syntax

```
Column.SetColumnDefaults
```

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Sets default values of the columns in a collection.

To facilitate the process of filling out a document profile, DM allows each user or the system administrator to set up their own or group default entries to appear in the profile fields each time they create a document. For example, if the documents the user generally types are documents authored by him, he/she can set himself/herself as the default Author entry.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IColumns" on page 285

IColumns::_NewEnum

The _NewEnum property returns the IEnumVARIANT interface on an object that can be used to enumerate the collection.

C++ Syntax

```
HRESULT _NewEnum(LPUNKNOWN* ppEnum);
```

Parameters

ppEnum	[out, retval] On return, an IEnumVARIANT object reference that can be used to enumerate the collection.

Remarks

Returns an object that supports IEnumVARIANT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IColumns" on page 285

IDocProfiles

The DocProfiles object represents a collection of document profiles.

When to Use

A DocProfiles object may be created with the CreateObject function or a Dim statement with the New keyword. The following example shows the Visual Basic syntax for creating a DOCSObjects.DocProfiles object:

```
Sub Form_Load()
 ...
 Dim Profs As Object
 ' Create a DOCSObjects.DocProfiles object
 Set Profs = CreateObject("DOCSObjects.DocProfiles")
 ...
 End Sub
```

Methods

IDocProfiles Methods	Description
IDocProfiles::Add	Adds a new item to a collection.
IDocProfiles::Remove	Removes a specific item from a collection.
IDocProfiles::Clear	Removes all items from a collection.
IDocProfiles::CheckOut	Checks out documents in a collection.
IDocProfiles::GetMenuItems	Returns menu items for documents in a collection.
IDocProfiles::ExecuteMenuItems	Performs an action for documents in a collection.
IDocProfiles::ShowSelectVersionsForm	Displays a form for selecting document versions/ attachments.

Properties

IDocProfiles Properties	Description
IDocProfiles::Item	Returns a specific item of a collection either by position or by key.

IDocProfiles Properties	Description
IDocProfiles::Count	Returns the number of items in a collection.
IDocProfiles::_NewEnum	Returns an object that supports IEnumVARIANT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

IDocProfiles::Add

The method adds a new item to a collection.

C++ Syntax

```
HRESULT Add(VARIANT vItem);
```

Visual Basic Syntax

```
DocProfiles.Add (vItem)
```

Parameters

vItem	[in] Required. An expression that specifies the item to add to the collection. If a string expression, vItem should specify the library name and document number as follows: LibraryName#DocNumber, where LibraryName is the name of the library and DocNumber is the unique document number for the profiled document to be added to the collection. If an object expression, vItem should specify a valid Profile object.

Return Values

S_OK	The operation succeeded.
E_OUTOFMEMORY	Out of memory.
E_INAVLIDARG	Invalid parameter has been passed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Adds a new item to a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IDocProfiles" on page 292

"IProfile" on page 419

IDocProfiles::CheckOut

The method checks out all documents in a collection.

C++ Syntax

```
HRESULT CheckOut(short sFlags);
```

Visual Basic Syntax

```
DocProfiles.CheckOut (sFlags)
```

Parameters

sFlags	[in] Required. An integer value that specifies how the method should be executed. The parameter can be a combination of the FormsResults enumeration values.

Return Values

S_OK	The operation succeeded.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Checks out documents in a collection.

If the show form bit is on, check-out dialog pops up. It can be customized and its form should support the IGroupCheckoutForm interface. To display the form, the method calls up the ShowGroupCheckoutForm method on IGroupCheckoutForm.

The following example checks out the document with number 20 on the primary library.

```
Dim App As New Application
```

```
Dim Profs As New DocProfiles
Dim Prof As Profile
Set Prof = App.PrimaryLibrary.GetProfile(20)
If Not Prof Is Nothing Then
Profs.Add Prof
Profs.CheckOut OF_SHOWFORM
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IDocProfiles" on page 292

"IGroupCheckoutForm" on page 881

IDocProfiles::Clear

The method removes all items from a collection.

C++ Syntax

```
HRESULT Clear();
```

Visual Basic Syntax

```
DocProfiles.Clear
```

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Removes all items from a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IDocProfiles" on page 292

IDocProfiles::Count

The property returns the number of items in a collection.

C++ Syntax

```
HRESULT Count(long* plCount);
```

Visual Basic Syntax

```
[ plCount = ] DocProfiles.Count
```

Parameters

plCount	[out, retval] Returns a long that specifies the number of items in the collection.

Return Values

S_OK	The operation succeeded.
E_POINTER	An output parameter should not be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the number of items in a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IDocProfiles" on page 292

IDocProfiles::ExecuteMenuItems

The method executes a command for documents in a collection.

C++ Syntax

```
HRESULT ExecuteMenuItems(BSTR bstrVerb);
```

Visual Basic Syntax

```
DocProfiles.ExecuteMenuItems (bstrVerb)
```

Parameters

bstrVerb	[in] Required. A string that specifies the command to execute.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Performs an action for documents in a collection.

The following example checks out documents the user selected in the Quick Retrieve form.

```
Dim App As New Application
 Dim Profs As DocProfiles
 ' Displays the QR form
 QR.DisplayModal QR_NORMAL
 Set Profs = QR.SelectedDocuments
 If Profs.Count > 0 Then
 'Execute a menu command (check-out) on the collection
 Profs.ExecuteMenuItems Profs.GetMenuItems(5).ID
 End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IDocProfiles" on page 292

IDocProfiles::GetMenuItems

The method enumerates menu items available to documents in a collection.

C++ Syntax

```
HRESULT GetMenuItems(IBCEnumVerbs** ppEnumVerbs);
```

Visual Basic Syntax

```
[ Set ppEnumVerbs  = ] DocProfiles.GetMenuItems
```

Parameters

ppEnumVerbs	[out, retval] Returns a BCEnumVerbs object that specifies the requested menu items.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns menu items for documents in a collection.

The following example checks out documents the user selected in the Quick Retrieve form.

```
Dim App As New Application
 Dim Profs As DocProfiles
 ' Displays the QR form
 QR.DisplayModal QR_NORMAL
 Set Profs = QR.SelectedDocuments
 If Profs.Count > 0 Then
 'Execute a menu command (check-out) on the collection
 Profs.ExecuteMenuItems Profs.GetMenuItems(5).ID
 End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IDocProfiles" on page 292

"IBCEnumVerbs" on page 74

IDocProfiles::Item

The property returns a specific member of a collection either by position or by key.

C++ Syntax

```
HRESULT Item(VARIANT vIndex, IProfile** ppProfile);
```

Visual Basic Syntax

```
[ Set ppProfile = ] DocProfiles.Item (vIndex)
```

Parameters

vIndex	[in] Required. An expression that specifies the member position in the collection. If a numeric expression, vIndex should be a number from 1 to the value of the collection's Count property. If a string expression, vIndex should correspond to the key argument specified when the member referred to was added to the collection.
ppProfile	[out, retval] Returns a Profile object that specifies the requested item.

Return Values

S_OK	The operation succeeded.
E_POINTER	An output parameter should not be NULL.
E_INVALIDARG	Invalid argument passed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns a specific item of a collection either by position or by key.

If the value provided as vIndex does not match any existing member of the collection, an error occurs. Item is the default property for a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IDocProfiles" on page 292

"IProfile" on page 419

IDocProfiles::Remove

The method removes a specific item from a collection.

C++ Syntax

```
HRESULT Remove(VARIANT vIndex);
```

Visual Basic Syntax

```
DocProfiles.Remove (vIndex)
```

Parameters

vIndex	[in] Required. A variant expression specifying the name or index in the collection of the object to be accessed.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Removes a specific item from a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IDocProfiles" on page 292

IDocProfiles::ShowSelectVersionsForm

The method displays a form for selecting document versions and/or attachments.

C++ Syntax

```
HRESULT ShowSelectVersionsForm(
short sFlags, BSTR bstrCaption, IVersions** ppVersions);
```

Visual Basic Syntax

```
[ Set ppVersions  = ] DocProfiles.ShowSelectVersionsForm (sFlags,
[ bstrCaption ] )
```

Parameters

sFlags	[in] Required. An integer value that specifies the form appearance. The parameter can be one of the SelectVersionsFormFlags enumeration values.
bstrCaption	[in, optional] Required. A string that specifies the form caption.
ppVersions	[out, retval] Returns a Versions object that specifies the collection of versions selected by the user. If the user selected no versions, the return value is NULL.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Displays a form for selecting document versions/attachments.

The form can be customized and should support the ISelectVersionsForm interface. To display the form, the method calls up the ShowSelectVersionsForm method on ISelectVersionsForm.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IDocProfiles" on page 292

"IVersions" on page 673

"ISelectVersionsForm" on page 947

IDocProfiles::_NewEnum

The _NewEnum property returns the IEnumVARIANT interface on an object that can be used to enumerate the collection.

C++ Syntax

```
HRESULT _NewEnum(LPUNKNOWN* ppEnum);
```

Parameters

ppEnum	[out, retval] On return, an IEnumVARIANT object reference that can be used to enumerate the collection.

Remarks

Returns an object that supports IEnumVARIANT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IDocProfiles" on page 292

IDocProfiles2

The IDocProfiles2 interface extends IDocProfiles object.

Methods

IDocProfiles2 Methods	Description
IDocProfiles2::ShowRelatedItemsForm	Shows related documents form.
IDocProfiles2::AddToRelated	Adds document to related documents.

IDocProfiles2 Methods	Description
IDocProfiles2::RemoveFromRelated	Removes document from related documents.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import "fsplugin.dll"

IDocProfiles2::AddToRelated

The method adds profiled item(s) to relation.

C++ Syntax

```
HRESULT AddToRelated(  VARIANT vDocument,   VARIANT_BOOL*
pbSuccess);
```

Visual Basic Syntax

```
[ Set pbSuccess = ] DocProfiles2.AddToRelated (vDocument)
```

Parameters

vDocument	[in] An expression that specifies the item(s) to add to relationship. If a string expression, vItem must specify the library name and document number as follows: LibraryName#DocNumber, where LibraryName is the name of the library and DocNumber is the unique document number for the profiled item to be added to relationship. If an object expression, vItem should be a valid Profile or DocProfiles object.
pbSuccess	[out, retval] Returns a result of operation.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_OUTOFMEMORY	Out of memory.
E_INAVLIDARG	Invalid parameter has been passed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Adds document to related documents.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import "fsplugin.dll"

See Also

"IDocProfiles2" on page 302

IDocProfiles2::RemoveFromRelated

The method removes profiled item(s) from relation.

C++ Syntax

```
HRESULT RemoveFromRelated(  VARIANT vDocument,   VARIANT_BOOL*
```

pbSuccess);

Visual Basic Syntax

```
[ Set pbSuccess = ] DocProfiles2.RemoveFromRelated (vDocument)
```

Parameters

vDocument	[in] An expression that specifies the item(s) to remove from relationship. If a string expression, vItem must specify the library name and document number as follows: LibraryName#DocNumber, where LibraryName is the name of the library and DocNumber is the unique document number for the profiled item to be removed from relationship. If an object expression, vItem should be a valid Profile or DocProfiles object.
pbSuccess	[out, retval] Returns a result of operation.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_OUTOFMEMORY	Out of memory.
E_INAVLIDARG	Invalid parameter has been passed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Removes document from related documents.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import "fsplugin.dll"

See Also

"IDocProfiles2" on page 302

IDocProfiles2::ShowRelatedItemsForm

The method displays Related Items form for profiled items.

C++ Syntax

```
HRESULT ShowRelatedItemsForm(  short sFlags,   short* psResult);
```

Visual Basic Syntax

```
[ Set psResult = ] DocProfiles2.ShowRelatedItemsForm (sFlags)
```

Parameters

sFlags	[in] Reserved. Should be 0.
psResult	[out, retval] Returns an integer that specifies user's action on the form. For possible values see FormsResults Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_OUTOFMEMORY	Out of memory.
E_INAVLIDARG	Invalid parameter has been passed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Shows related documents form.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import "fsplugin.dll"

See Also

"IDocProfiles2" on page 302

IDOCSForm

The DOCSForm object represents a DM form customizable through DM Designer. It is used for retrieving the form's definition (for the form interpreter) as well as translating the SQL column names into the names of the corresponding controls on the form.

The DOCSForm object may be retrieved through the Forms collection on a library.

The DOCSForm object inherits its definition from the IOMErrInfo interface so its implementation also includes all the properties of IOMErrInfo in addition to the ones described below.

Methods

IDOCSForm Methods	Description
IDOCSForm::GetNameFromSQL	Returns the SQL name of a form.

Properties

IDOCSForm Properties	Description
IDOCSForm::Name	Returns the name of a form.
IDOCSForm::Title	Returns the title of a form.
IDOCSForm::FormDefinition	Returns the definition of a form.
IDOCSForm::Type	Returns the form type.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IDOCSForms" on page 312

"ILibrary" on page 360

"IOMErrInfo" on page 412

IDOCSForm::FormDefinition

The property returns the definition of a form.

C++ Syntax

```
HRESULT FormDefinition(BSTR* pbstrDefinition);
```

Visual Basic Syntax

```
[ pbstrDefinition = ] DOCSForm.FormDefinition
```

Parameters

pbstrDefinition	[out, retval] Returns a string that specifies the form's definition.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns the definition of a form.

The form's definition is information used by DM to generate the form. It describes form's properties and all the controls the form contains.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IDOCSForm" on page 306

IDOCSForm::GetNameFromSQL

The method translates a SQL column name into the name of the object associated with this column on a form.

C++ Syntax

```
HRESULT GetNameFromSQL(
BSTR bstrSQL, BSTR* pbstrName);
```

Visual Basic Syntax

```
[ pbstrName = ] DOCSForm.GetNameFromSQL (bstrSQL)
```

Parameters

bstrSQL	[in] Required. A string that specifies a SQL column.
pbstrName.	[out, retval] The method returns a string with the name of the object associated with the SQL column.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The method searches the form's definition for the name of the object whose SQLInfo description completely coincides with the passed string or begins with this string immediately followed by a semicolon.

The following example retrieves the name of the object associated with the SQL column AUTHOR.USER_ID;DOCSADM.PEOPLE.SYSTEM_ID.

```
Dim App As New Application
 Dim Frm As DOCSForm
 Set Frm = App.PrimaryLibrary.Forms("DEF_PROF")
 MsgBox "Name: " + _
 Frm.GetNameFromSQL("AUTHOR.USER_ID;DOCSADM.PEOPLE.SYSTEM_ID")
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IDOCSForm" on page 306

IDOCSForm::Name

The property returns the short name of a form.

C++ Syntax

```
HRESULT Name(BSTR* pbstrName);
```

Visual Basic Syntax

```
[ pbstrName = ] DOCSForm.Name
```

Parameters

pbstrName	[out, retval] Returns a string that specifies the form's name.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns the name of a form.

The following example displays the names and titles of all forms on the primary library.

```
Dim App As New Application
 Dim Frm As DOCSForm
 For Each Frm In App.PrimaryLibrary.Forms
 MsgBox "Name: " + Frm.Name + Chr(13) + "Title: " + Frm.Title
 Next
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IDOCSForm" on page 306

IDOCSForm::Title

The property returns the title of a form.

C++ Syntax

```
HRESULT Title(BSTR* pbstrTitle);
```

Visual Basic Syntax

```
[ pbstrTitle = ] DOCSForm.Title
```

Parameters

pbstrTitle	[out, retval] Returns a string that specifies the form's title.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns the title of a form.

The form title is its full descriptive name. When form is displayed, this text will appear in the form's title bar.

The following example displays the names and titles of all forms on the primary library.

```
Dim App As New Application
 Dim Frm As DOCSForm
 For Each Frm In App.PrimaryLibrary.Forms
 MsgBox "Name: " + Frm.Name + Chr(13) + "Title: " + Frm.Title
 Next
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IDOCSForm" on page 306

IDOCSForm::Type

The property returns the type of a form.

C++ Syntax

```
HRESULT Type(short* psType);
```

Visual Basic Syntax

```
[ psType = ] DOCSForm.Type
```

Parameters

psType	[out, retval] Returns an integer that specifies the form's type. It can be one of the FTypes enumeration values.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns the form type.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IDOCSForm" on page 306

IDOCSForms

The DOCSForms object represents a collection of forms from a library including the Quick Retrieve and Profile/QBE forms. The collection may be retrieved with the Forms property on a library.

Properties

IDOCSForms Properties	Description
Item	Returns a specific item of a collection either by position or by key.
Count	Returns the number of items in a collection.
_NewEnum	Returns an object that supports IEnumVARIANT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

IDOCSForms::Count

The property returns the number of items in a collection.

C++ Syntax

```
HRESULT Count(long* plCount);
```

Visual Basic Syntax

```
[ plCount = ] DOCSForms.Count
```

Parameters

plCount	[out, retval] Returns a long that specifies the number of items in the collection.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns the number of items in a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IDOCSForms" on page 312

IDOCSForms::Item

The property returns a specific member of a collection either by position or by key.

C++ Syntax

```
HRESULT Item(VARIANT vIndex, IDOCSForm** ppForm);
```

Visual Basic Syntax

```
[ Set ppForm = ] DOCSForms.Item (vIndex)
```

Parameters

vIndex	[in] Required. An expression that specifies the member position in the collection. If a numeric expression, vIndex should be a number from 1 to the value of the collection's Count property. If a string expression, vIndex should correspond to a form name.
ppForm	[out, retval] Returns a DOCSForm object that specifies the requested item.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns a specific item of a collection either by position or by key.

If the value provided as vIndex does not match any existing member of the collection, an error occurs. Item is the default property for a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IDOCSForms" on page 312

"IDOCSForm" on page 306

IDOCSForms::_NewEnum

The _NewEnum property returns the IEnumVARIANT interface on an object that can be used to enumerate the collection.

C++ Syntax

```
HRESULT _NewEnum(LPUNKNOWN* ppEnum);
```

Parameters

ppEnum	[out, retval] On return, an IEnumVARIANT object reference that can be used to enumerate the collection.

Remarks

Returns an object that supports IEnumVARIANT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

IDOComponent

The DOComponent object represents a information about component info (for folders in Public Folders node or for documents/sub-folders in profiled folder).

Properties

IDOComponent Properties	Description
Profile	Returns/sets profile.
VersionType	Returns/sets version type.
DisplayName	Returns/sets display name.
Order	Returns/sets order.
ParentVersion	Returns/sets parent version.
ID	Returns/sets ID.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import "fsplugin.dll"

IDOComponent::DisplayName

The property returns/sets display name of component.

C++ Syntax

```
HRESULT DisplayName(  BSTR* pstrDisplayName);
HRESULT DisplayName(  BSTR strDisplayName);
```

Visual Basic Syntax

```
[ pstrDisplayName = ] DOComponent.DisplayName
DOComponent.DisplayName [ = strDisplayName ]
```

Parameters

pstrDisplayName	[out, retval] Returns display name of component.
strDisplayName	[in] Sets display name of component.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns/sets display name.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IDOComponent" on page 315

IDOComponent::ID

The property returns/sets System ID of component.

C++ Syntax

```
HRESULT ID(  long* pnID);
HRESULT ID(  long nID);
```

Visual Basic Syntax

```
[ pnID = ] DOComponent.ID
DOComponent.ID [ = nID ]
```

Parameters

pnID	[out, retval] Returns System ID of component.
nID	[in] Sets System ID of component.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns/sets ID.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IDOComponent" on page 315

IDOComponent::Order

The property returns/sets order.

C++ Syntax

```
HRESULT Order(  long* pnOrder);
HRESULT Order(  long nOrder);
```

Visual Basic Syntax

```
[ pnOrder = ] DOComponent.Order
DOComponent.Order [ = nOrder ]
```

Parameters

pnOrder	[out, retval] Returns order of component.
nOrder	[in] Sets order of component.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns/sets order.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IDOComponent" on page 315

IDOComponent::ParentVersion

The property returns/sets parent version.

C++ Syntax

```
HRESULT ParentVersion(  BSTR* pstrParentVersion);
HRESULT ParentVersion(  BSTR strParentVersion);
```

Visual Basic Syntax

```
[ pstrParentVersion = ] DOComponent.ParentVersion
DOComponent.ParentVersion [ = strParentVersion ]
```

Parameters

pstrParentVersion	[out, retval] Returns version label of parent object.
strParentVersion	[in] Sets version label of parent object.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns/sets parent version.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IDOComponent" on page 315

IDOComponent::Profile

The property returns/sets Profile object of component.

C++ Syntax

```
HRESULT Profile(  IProfile** ppProfile);
HRESULT Profile(  IProfile* pProfile);
```

Visual Basic Syntax

```
[ ppProfile = ] DOComponent.Profile
DOComponent.Profile = [ pProfile ]
```

Parameters

ppProfile	[out, retval] Returns Profile object.
pProfile	[in] Sets Profile object.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns/sets profile.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IDOComponent" on page 315

"IProfile" on page 419

IDOComponent::VersionType

The property returns/sets Version Type of component.

C++ Syntax

```
HRESULT VersionType(  BSTR* pstrVersionType);
HRESULT VersionType(  BSTR strVersionType);
```

Visual Basic Syntax

```
[ pstrVersionType = ] DOComponent.VersionType
DOComponent.VersionType [ = strVersionType ]
```

Parameters

pstrVersionType	[out, retval] Returns version type of component. R: Current (most recent) version P: Published (last published) version VersionLabel: Specific version
strVersionType	[in] Sets version type of component.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns/sets version type.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IDOComponent" on page 315

IDOComponents

The DOComponents object represents a collection of DOComponent objects.

Methods

IDOComponents Methods	Description
IDOComponents::Add	Adds a new item to a collection.
IDOComponents::Remove	Removes a specific item from a collection.
IDOComponents::Clear	Removes all items from a collection.

Properties

IDOComponents Properties	Description
IDOComponents::Item	Returns a specific item of a collection either by position or by key.
IDOComponents::Count	Returns the number of items in a collection.
IDOComponents::_NewEnum	Returns an object that supports IEnumVARIANT.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

IDOComponents::Add

The method adds a new item to a collection.

C++ Syntax

```
HRESULT Add(IDOComponent* pComponent);
```

Visual Basic Syntax

```
DOComponents.Add (pComponent)
```

Parameters

pComponent	[in] An DOComponent object that specifies the item to add to the collection.

Return Values

S_OK	The operation succeeded.
E_OUTOFMEMORY	Out of memory.
E_INAVLIDARG	Invalid parameter has been passed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Adds a new item to a collection.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IDOComponents" on page 321

IDOComponents::Clear

The method removes all items from a collection.

C++ Syntax

```
HRESULT Clear();
```

Visual Basic Syntax

```
DOComponents.Clear
```

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Removes all items from a collection.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IDOComponents" on page 321

IDOComponents::Count

The property returns the number of items in a collection.

C++ Syntax

```
HRESULT Count(long* plCount);
```

Visual Basic Syntax

```
[ plCount = ] DOComponents.Count
```

Parameters

plCount	[out, retval] Returns a long that specifies the number of items in the collection.

Return Values

S_OK	The operation succeeded.
E_POINTER	An output parameter should not be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the number of items in a collection.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IDOComponents" on page 321

IDOComponents::Item

The property returns a specific member of a collection either by position or by key.

C++ Syntax

```
HRESULT Item(VARIANT vIndex, IDOComponent** ppComponent);
```

Visual Basic Syntax

```
[ Set ppComponent = ] DOComponents.Item (vIndex)
```

Parameters

vIndex	[in] An expression that specifies the position of a member of the collection. If a numeric expression, vIndex must be a number from 1 to the value of the collection's Count property. If a string expression, vIndex must correspond to a display name of component.
ppProfile	[out, retval] Returns a DOComponent object that specifies the requested item.

Return Values

S_OK	The operation succeeded.
E_POINTER	An output parameter should not be NULL.
E_INVALIDARG	Invalid argument passed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns a specific item of a collection either by position or by key.

If the value provided as vIndex does not match any existing member of the collection, an error occurs. Item is the default property for a collection.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IDOComponents" on page 321

IDOComponents::Remove

The method removes a specific item from a collection.

C++ Syntax

```
HRESULT Remove(VARIANT vIndex);
```

Visual Basic Syntax

```
DOComponents.Remove (vIndex)
```

Parameters

vIndex	[in] Required. A variant expression specifying the name or index in the collection of the object to be accessed.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Removes a specific item from a collection.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IDOComponents" on page 321

IDOComponents::_NewEnum

The _NewEnum property returns the IEnumVARIANT interface on an object that can be used to enumerate the collection.

C++ Syntax

```
HRESULT _NewEnum(LPUNKNOWN* ppEnum);
```

Parameters

ppEnum	[out, retval] On return, an IEnumVARIANT object reference that can be used to enumerate the collection.

Remarks

Returns an object that supports IEnumVARIANT.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IDOComponents" on page 321

IDocTransferNotify

The DocTransferNotify object is used for receiving notifications about downloading a document from the server with the DownloadFile method on a version or uploading a document to the server with the UploadFile method.

When to Implement

Clients should implement this object to receive such notifications.

Methods

IDocTransferNotify Methods	Description
IDocTransferNotify::OnProgressNotify	Notifies an object that a portion of data has been transferred.
IDocTransferNotify::OnError	Notifies about an error.
IDocTransferNotify::Completed	Notifies an object that data transfer has been completed.
IDocTransferNotify::InitTransfer	Notifies an object that data transfer is about to begin.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

IDocTransferNotify::Completed

The method indicates that the data transfer has been completed.

C++ Syntax

```
HRESULT Completed(short sReason);
```

Visual Basic Syntax

```
DocTransferNotify.Completed (sReason)
```

Parameters

sReason	[in] An integer that specifies why the process has finished. The parameter may be one of the TransferNotifyCompletedReason enumeration values.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Notifies an object that data transfer has been completed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IDocTransferNotify" on page 327

IDocTransferNotify::InitTransfer

The method is called when data transfer is about to begin.

C++ Syntax

```
HRESULT InitTransfer(long lTotalSize);
```

Visual Basic Syntax

```
DocTransferNotify.InitTransfer (lTotalSize)
```

Parameters

lTotalSize	[in] A long that specifies the size of the document being transferred in bytes.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Notifies an object that data transfer is about to begin.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IDocTransferNotify" on page 327

IDocTransferNotify::OnError

The method is called when an error occurs while uploading or downloading the document's contents.

C++ Syntax

```
HRESULT OnError(BSTR bstrErrorText);
```

Visual Basic Syntax

```
DocTransferNotify.OnError (bstrErrorText)
```

Parameters

bstrErrorText	[in] A string with the error description.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Notifies about an error.

The following example displays a message box if an error occurs while uploading or downloading a document.

```
Public Sub DocTransferNotify_OnError(ByVal Description As String)

 MsgBox Description

 End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IDocTransferNotify" on page 327

IDocTransferNotify::OnProgressNotify

The method receives notifications about document contents' transfer progress.

C++ Syntax

```
HRESULT OnProgressNotify(long lBytesRead);
```

Visual Basic Syntax

```
DocTransferNotify.OnProgressNotify (lBytesRead)
```

Parameters

lBytesRead	[in] A long that specifies the number of bytes that have been transferred since the last call to the method.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Notifies an object that a portion of data has been transferred.

The method can be used to show the user the status of transfer process.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IDocTransferNotify" on page 327

ILaunchMethod

The LaunchMethod object represents a launch method. Launch methods provide different ways to launch the same application from DM Extensions.

LaunchMethod objects may be retrieved through the LaunchMethods collection on an AppInfo object.

Properties

ILaunchMethod Properties	Description
ID	Returns the identifier of a launch method.
Application	Returns the parent application information object of a launch method.
Description	Returns a description of a launch method.
Location	Returns the location of a launch method.
CommandLine	Returns the command line of a launch method.
DefaultDirectory	Returns the default directory of a launch method.
IntegrationScheme	Returns the integration scheme of a launch method.
DDEOpenAppName	Returns the DDE open application name of a launch method.
DDEOpenTopic	Returns the DDE open topic of a launch method.
DDEOpenCmd	Returns the main DDE open command of a launch method.
DDEOpenCmd2	Returns a subsidiary DDE open command of a launch method.
DDEPrintAppName	Returns the DDE print application name of a launch method.
DDEPrintTopic	Returns the DDE print topic of a launch method.
DDEPrintCmd	Returns the main DDE print command of a launch method.
DDEPrintCmd2	Returns a subsidiary DDE print command of a launch method.
MonitorDir	Returns the monitor directory of a launch method.

MonitorSubdirs	Returns whether it is necessary to include subdirectories while monitoring.
MonitorFiles	Returns a collection of launch method monitor files.
Default	Returns/sets whether a launch method is default.
ValidForOpen	Returns whether a launch method is valid for opening.
ValidForPrint	Returns whether a launch method is valid for printing.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILaunchMethods" on page 354

"IAppInfo" on page 220

ILaunchMethod::Application

The property returns the application associated with a launch method.

C++ Syntax

```
HRESULT Application(IAppInfo** ppAppInfo);
```

Visual Basic Syntax

```
[ Set ppAppInfo = ] LaunchMethod.Application
```

Parameters

ppAppInfo	[out, retval] Returns an AppInfo object that specifies the parent application for this launch method.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the parent application information object of a launch method.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILaunchMethod" on page 331

"IAppInfo" on page 220

ILaunchMethod::CommandLine

The property returns command line parameters including any switches or text sent to the application associated with a launch method upon execution.

C++ Syntax

```
HRESULT CommandLine(BSTR* pbstrCommandLine);
```

Visual Basic Syntax

```
[ pbstrCommandLine  = ] LaunchMethod.CommandLine
```

Parameters

pbstrCommandLine	[out, retval] Returns a string that specifies the command line parameters.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the command line of a launch method.

The command line parameters are defined by each application vendor in the application documentation.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILaunchMethod" on page 331

ILaunchMethod::DDEOpenAppName

The property returns the name for the application associated with a launch method to open in DDE conversation.

C++ Syntax

```
HRESULT DDEOpenAppName(BSTR* pbstrDDEOpenAppName);
```

Visual Basic Syntax

```
[ pbstrDDEOpenAppName = ] LaunchMethod.DDEOpenAppName
```

Parameters

pbstrDDEOpenAppName	[out, retval] Returns a string that specifies the DDE name of the application.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the DDE open application name of a launch method.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILaunchMethod" on page 331

ILaunchMethod::DDEOpenCmd

The property returns the DDE command to open the application associated with a launch method, when the application is already running.

C++ Syntax

```
HRESULT DDEOpenCmd(BSTR* pbstrDDEOpenCommand);
```

Visual Basic Syntax

```
[ pbstrDDEOpenCommand = ] LaunchMethod.DDEOpenCmd
```

Parameters

pbstrDDEOpenCommand	[out, retval] Returns a string that specifies the DDE command.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the main DDE open command of a launch method.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILaunchMethod" on page 331

ILaunchMethod::DDEOpenCmd2

The property returns the DDE command used to open a document inside the application associated with a launch method when the application is not already running.

C++ Syntax

```
HRESULT DDEOpenCmd2(BSTR* pbstrDDEOpenCommand);
```

Visual Basic Syntax

```
[ pbstrDDEOpenCommand  = ] LaunchMethod.DDEOpenCmd2
```

Parameters

pbstrDDEOpenCommand	[out, retval] Returns a string that specifies the DDE command.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns a subsidiary DDE open command of a launch method.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILaunchMethod" on page 331

ILaunchMethod::DDEOpenTopic

The property returns the DDE topic name for the application associated with a launch method to open.

C++ Syntax

```
HRESULT DDEOpenTopic(BSTR* pbstrDDEOpenTopic);
```

Visual Basic Syntax

```
[ pbstrDDEOpenTopic  = ] LaunchMethod.DDEOpenTopic
```

Parameters

pbstrDDEOpenTopic	[out, retval] Returns a string that specifies the DDE topic name.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the DDE open topic of a launch method.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILaunchMethod" on page 331

ILaunchMethod::DDEPrintAppName

The property returns the name for the application associated with a launch method to print in DDE conversation.

C++ Syntax

```
HRESULT DDEPrintAppName(BSTR* pbstrDDEPrintAppName);
```

Visual Basic Syntax

```
[ pbstrDDEPrintAppName = ] LaunchMethod.DDEPrintAppName
```

Parameters

pbstrDDEPrintAppName	[out, retval] Returns a string that specifies the DDE name of the application.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the DDE print application name of a launch method.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILaunchMethod" on page 331

ILaunchMethod::DDEPrintCmd

The property returns the DDE command to start the print application associated with a launch method, when the application is already running.

C++ Syntax

```
HRESULT DDEPrintCmd(BSTR* pbstrDDEPrintCommand);
```

Visual Basic Syntax

```
[ pbstrDDEPrintCommand = ] LaunchMethod.DDEPrintCmd
```

Parameters

pbstrDDEPrintCommand	[out, retval] Returns a string that specifies the DDE command.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the main DDE print command of a launch method.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILaunchMethod" on page 331

ILaunchMethod::DDEPrintCmd2

The property returns the DDE command used to print a document when the print application associated with a launch method is not already running.

C++ Syntax

```
HRESULT DDEPrintCmd2(BSTR* pbstrDDEPrintCommand);
```

Visual Basic Syntax

```
[ pbstrDDEPrintCommand  = ] LaunchMethod.DDEPrintCmd2
```

Parameters

pbstrDDEPrintCommand	[out, retval] Returns a string that specifies the DDE command.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns a subsidiary DDE print command of a launch method.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILaunchMethod" on page 331

ILaunchMethod::DDEPrintTopic

The property returns the DDE topic name for the print application associated with a launch method.

C++ Syntax

```
HRESULT DDEPrintTopic(BSTR* pbstrDDEPrintTopic);
```

Visual Basic Syntax

```
[ pbstrDDEPrintTopic = ] LaunchMethod.DDEPrintTopic
```

Parameters

pbstrDDEPrintTopic	[out, retval] Returns a string that specifies the DDE topic name.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the DDE print topic of a launch method.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILaunchMethod" on page 331

ILaunchMethod::Default

The property returns or sets whether or not a launch method is default.

C++ Syntax

```
HRESULT Default(VARIANT_BOOL* pbDefault);
HRESULT Default(VARIANT_BOOL bDefault);
```

Visual Basic Syntax

```
[ pbDefault = ] LaunchMethod.Default
LaunchMethod.Default [ = bDefault ]
```

Parameters

pbDefault	[out, retval] A Boolean value that specifies whether or not the launch method is default.
bDefault	[in] A Boolean value that specifies whether or not the launch method is default. True: The launch method is default. False: The launch method is not default.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns/sets whether a launch method is default.

A default launch method overrides all others for a particular application. When opening a document inside the application the user will default to this launch method without displaying the launch methods dialog beforehand.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILaunchMethod" on page 331

ILaunchMethod::DefaultDirectory

The property returns the directory the user is placed in before launching the application with a launch method.

C++ Syntax

```
HRESULT DefaultDirectory(BSTR* pbstrDefaultDirectory);
```

Visual Basic Syntax

```
[ pbstrDefaultDirectory = ] LaunchMethod.DefaultDirectory
```

Parameters

pbstrDefaultDirectory	[out, retval] Returns a string that specifies the default directory.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the default directory of a launch method.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILaunchMethod" on page 331

ILaunchMethod::Description

The property returns the long description for a launch method, usually including its application version number.

C++ Syntax

```
HRESULT Description(BSTR* pbstrDescription);
```

Visual Basic Syntax

```
[ pbstrDescription = ] LaunchMethod.Description
```

Parameters

pbstrDescription	[out, retval] Returns a string that specifies the launch method description.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns a description of a launch method.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILaunchMethod" on page 331

ILaunchMethod::ID

The property returns the identifier of a launch method.

C++ Syntax

```
HRESULT ID(long* plID);
```

Visual Basic Syntax

```
[ plID = ] LaunchMethod.ID
```

Parameters

plID	[out, retval] Returns a long that uniquely identifies the launch method.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the identifier of a launch method.

The launch method's identifier is unique within a library, i.e. two launch methods on the same library have different identifiers.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILaunchMethod" on page 331

ILaunchMethod::IntegrationScheme

The property returns the type of integration used to configure a launch method for use with DM Extensions.

C++ Syntax

```
HRESULT IntegrationScheme(short* psScheme);
```

Visual Basic Syntax

```
[ psScheme  = ] LaunchMethod.IntegrationScheme
```

Parameters

psScheme	[out, retval] Returns an integer that identifies the integration type. It can be one of the LM_INTEGRATIONSCHEME values.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the integration scheme of a launch method.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILaunchMethod" on page 331

ILaunchMethod::Location

The property returns the path and executable file name for the application associated with a launch method.

C++ Syntax

```
HRESULT Location(BSTR* pbstrLocation);
```

Visual Basic Syntax

```
[ pbstrLocation = ] LaunchMethod.Location
```

Parameters

pbstrLocation	[out, retval] Returns a string that specifies the path to the application.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the location of a launch method.

If the application's directory is in the client's path statement, the executable name alone may be specified.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILaunchMethod" on page 331

ILaunchMethod::MonitorDir

The property returns the path DM Extensions monitors for new documents if a launch method uses Directory Monitoring integration.

C++ Syntax

```
HRESULT MonitorDir(BSTR* pbstrMonitorDir);
```

Visual Basic Syntax

```
[ pbstrMonitorDir = ] LaunchMethod.MonitorDir
```

Parameters

pbstrMonitorDir	[out, retval] Returns a string that specifies the monitoring directory.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the monitor directory of a launch method.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILaunchMethod" on page 331

ILaunchMethod::MonitorFiles

The property returns a collection of filename masks DM Extensions uses to monitor directories if a launch method uses Directory Monitoring integration.

C++ Syntax

```
HRESULT MonitorFiles(IMonitorFiles** ppMonitorFiles);
```

Visual Basic Syntax

```
[ Set ppMonitorFiles  = ] LaunchMethod.MonitorFiles
```

Parameters

ppMonitorFiles	[out, retval] Returns a MonitorFiles object that represents the collection of filename masks.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns a collection of launch method monitor files.

Filename masks specify files DM Extensions will either look for or exclude while monitoring files.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILaunchMethod" on page 331

"IMonitorFiles" on page 409

ILaunchMethod::MonitorSubdirs

The property returns whether or not subdirectories are monitored by DM Extensions if a launch method uses Directory Monitoring integration.

C++ Syntax

```
HRESULT MonitorSubdirs(VARIANT_BOOL* pbMonitor);
```

Visual Basic Syntax

```
[ pbMonitor = ] LaunchMethod.MonitorSubdirs
```

Parameters

pbMonitor	[out, retval] Returns a Boolean value that specifies whether or not subdirectories are monitored. True: Subdirectories are monitored. False: Subdirectories are not monitored.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns whether it is necessary to include subdirectories while monitoring.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILaunchMethod" on page 331

ILaunchMethod::ValidForOpen

The property returns whether or not the launch method is valid when opening a document.

C++ Syntax

```
HRESULT ValidForOpen(VARIANT_BOOL* pbValid);
```

Visual Basic Syntax

```
[ pbValid = ] LaunchMethod.ValidForOpen
```

Parameters

pbValid	[out, retval] Returns a Boolean value that specifies whether or not the launch method is valid for opening documents. True: The launch method is valid for opening documents. False: The launch method is not valid for opening documents.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns whether a launch method is valid for opening.

The property is called when there needs to determine whether a launch method is valid when opening a document. For instance, method validation is necessary when displaying the list of launch methods to choose one to open a document.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILaunchMethod" on page 331

ILaunchMethod::ValidForPrint

The property returns whether or not the launch method is valid when printing a document.

C++ Syntax

```
HRESULT ValidForPrint(VARIANT_BOOL* pbValid);
```

Visual Basic Syntax

```
[ pbValid = ] LaunchMethod.ValidForPrint
```

Parameters

pbValid	[out, retval] Returns a Boolean value that specifies whether or not the launch method is valid for printing documents. True: The launch method is valid for printing documents. False: The launch method is not valid for printing documents.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns whether a launch method is valid for printing.

The property is called when there needs to determine whether a launch method is valid for printing a document. For instance, method validation is necessary when displaying the list of launch methods to choose one to print a document.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILaunchMethod" on page 331

ILaunchMethods

The LaunchMethods object represents a collection of application's launch methods. The collection may be retrieved with the LaunchMethods property on an AppInfo object.

Properties

ILaunchMethods Properties	Description
Item	Returns a specific item of a collection either by position or by key.
Count	Returns the number of items in a collection.
_NewEnum	Returns an object that supports IEnumVARIANT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IAppInfo" on page 220

ILaunchMethods::Count

The property returns the number of items in a collection.

C++ Syntax

```
HRESULT Count(long* plCount);
```

Visual Basic Syntax

```
[ plCount = ] LaunchMethods.Count
```

Parameters

plCount	[out, retval] Returns a long that specifies the number of items in a collection.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the number of items in a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILaunchMethods" on page 354

ILaunchMethods::Item

The property returns a specific member of a collection either by position or by key.

C++ Syntax

```
HRESULT Item(VARIANT vIndex, ILaunchMethod** ppMethod);
```

Visual Basic Syntax

```
[ Set ppMethod = ] LaunchMethods.Item (vIndex)
```

Parameters

vIndex	[in] Required. An expression that specifies the member position in a collection. If a numeric expression, vIndex should be a number from 1 to the value of the collection's Count property. If a string expression, vIndex should correspond to a launch method description.
ppMethod	[out, retval] Returns a LaunchMethod object that specifies the requested item.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns a specific item of a collection either by position or by key.

If the value provided as vIndex does not match any existing member of the collection, an error occurs. Item is the default property for a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILaunchMethods" on page 354

ILaunchMethods::_NewEnum

The _NewEnum property returns the IEnumVARIANT interface on an object that can be used to enumerate the collection.

C++ Syntax

```
HRESULT _NewEnum(LPUNKNOWN* ppEnum);
```

Parameters

ppEnum	[out, retval] On return, an IEnumVARIANT object reference that can be used to enumerate the collection.

Remarks

Returns an object that supports IEnumVARIANT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILaunchMethods" on page 354

ILibraries

The Libraries object represents a collection of all libraries available to the user. The collection may be retrieved with the Libraries property on the Application object.

Properties

ILibraries Properties	Description
ILibraries::Item	Returns a specific item of a collection either by position or by key.
ILibraries::Count	Returns the number of items in a collection.
ILibraries::_NewEnum	Returns an object that supports IEnumVARIANT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IApplication" on page 241

ILibraries::Count

The property returns the number of items in a collection.

C++ Syntax

```
HRESULT Count(long* plCount);
```

Visual Basic Syntax

```
[ plCount = ] Libraries.Count
```

Parameters

plCount	[out, retval] Returns a long that specifies the number of items in the collection.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the number of items in a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibraries" on page 357

ILibraries::Item

The property returns a specific member of a collection either by position or by key.

C++ Syntax

```
HRESULT Item(VARIANT vIndex, ILibrary** ppLibrary);
```

Visual Basic Syntax

```
[ Set ppLibrary = ] Libraries.Item (vIndex)
```

Parameters

vIndex	[in] Required. An expression that specifies the member position in a collection. If a numeric expression, vIndex should be a number from 1 to the value of the collection's Count property. If a string expression, vIndex should correspond to a library name.
ppLibrary	[out, retval] Returns a Library object that specifies the requested item.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns a specific item of a collection either by position or by key.

If the value provided as vIndex does not match any existing member of a collection, an error occurs. Item is the default property for a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

"ILibraries" on page 357

ILibraries::_NewEnum

The _NewEnum property returns the IEnumVARIANT interface on an object that can be used to enumerate the collection.

C++ Syntax

```
HRESULT _NewEnum(LPUNKNOWN* ppEnum);
```

Parameters

ppEnum	[out, retval] On return, an IEnumVARIANT object reference that can be used to enumerate the collection.

Remarks

Returns an object that supports IEnumVARIANT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibraries" on page 357

ILibrary

The Library object represents a DM library.

The Library object inherits its definition from the IOMErrInfo interface so its implementation also includes all the properties of IOMErrInfo in addition to the ones described below.

Methods

ILibrary Methods	Description
ILibrary::GetProfile	Returns a document profile by its number or name.
ILibrary::CreateProfile	Creates a new document.
ILibrary::ProfileFile	Imports a file into a library.
ILibrary::SelectProfileForm	Displays a dialog for selecting a profile form.
ILibrary::CreateDocFromTemplate	Creates a new document based on a template.
ILibrary::CreateSearch	Creates a new search.
ILibrary::SelectSearchForm	Displays a dialog for selecting a QBE form.
ILibrary::CreateProject	Creates a new profiled folder.
ILibrary::GetProject	Returns a profiled folder by its name or ID.
ILibrary::MakeCurrent	Sets a library as current.

Properties

ILibrary Properties	Description
ILibrary::Application	Returns an application object.
ILibrary::IsCurrent	Returns whether a library is current.
ILibrary::IsPrimary	Returns whether a library is primary.
ILibrary::Name	Returns the name of a library.
ILibrary::UserName	Returns the current user's name.
ILibrary::Searches	Returns a collection of all searches for a library.
ILibrary::Forms	Returns a collection of all library forms.
ILibrary::Shadow	Returns whether a library is shadowed.
ILibrary::ShadowPathTemplate	Returns the template path to the Shadow directory.
ILibrary::LibraryGroup	Returns the name of the library group a library belongs to.
ILibrary::SearchByDefault	Returns/sets whether a library is marked for search by default.
ILibrary::Projects	Returns a collection of all library public folders.
ILibrary::Description	Returns a library description.
ILibrary::UserPrimaryGroup	Returns the user's primary group.
ILibrary::AppInformation	Returns a collection of all library applications.
ILibrary::Properties	Returns a collection of library persistent properties.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IOMErrInfo" on page 412

ILibrary::AppInformation

The property returns a collection of all applications, AppInfo objects, from a library.

C++ Syntax

```
HRESULT AppInformation(IApplications** ppApps);
```

Visual Basic Syntax

```
[ Set ppApps = ] Library.AppInformation
```

Parameters

ppApps	[out, retval] Returns an Applications object that specifies the collection of all library applications.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns a collection of all library applications.

An application is the primary computer program that handles a document from a library. The property specifies which applications can be used with documents from a library.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

"IAppInfo" on page 220

"IApplications" on page 266

ILibrary::Application

The property returns the Application object that is the parent of a library.

C++ Syntax

```
HRESULT Application(IApplication** ppApplication);
```

Visual Basic Syntax

```
[ Set ppApplication = ] Library.Application
```

Parameters

ppApplication	[out, retval] Returns the Application object that specifies the library parent.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns an application object.

The following example retrieves the parent Application object of a library.

```
Dim Lib As Library
 ' Some operations with a library
 ...
 Dim App As Application
 Set App = Lib.Application
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

"IApplication" on page 241

ILibrary::CreateDocFromTemplate

The method creates a new document profile by a specific document template.

C++ Syntax

```
HRESULT CreateDocFromTemplate(VARIANT vTemplate, IProfile**
ppProfile);
```

Visual Basic Syntax

```
[ Set ppProfile  = ] Library.CreateDocFromTemplate (vTemplate)
```

Parameters

vTemplate	in] Required. An expression that specifies a document template profile. If a numeric expression, vTemplate should specify the number of a template document from the library. If an object, vTemplate should specify a template Profile object.
ppProfile	[out, retval] Returns a Profile object that specifies the created profile. If the user did not create the profile by pressing a Cancel button in its form, the return value is NULL.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Creates a new document based on a template.

The following example displays the form for selecting a template and creates a new document based on the template the user selected.

```
Dim App As New Application
 Dim Template, Prof As Profile
 Set Template = App.ShowTemplateChooserForm
 If Not Template Is Nothing Then
 Set Prof = App.PrimaryLibrary.CreateDocFromTemplate(Template)
 End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

"IProfile" on page 419

ILibrary::CreateProfile

The method creates an empty document profile on the library by a specified profile form.

C++ Syntax

```
HRESULT CreateProfile(BSTR bstrFormName, IProfile** ppProfile);
```

Visual Basic Syntax

```
[ Set ppProfile  = ] Library.CreateProfile (vTemplate)
```

Parameters

bstrFormName	[in] Required. A string that specifies a profile form on the library.
Param2	[out, retval] Returns a Profile object that specifies the created profile.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Creates a new document.

Creating a new profile simply creates a blank Profile object. The object is pre-filled out with the default profile values for the current user. Besides, the typist field is also set to the current user. Values must be assigned for all required fields in the profile before it can be saved. Once the profile object is created all of the properties and methods associated with the Profile class can be used with the object.

The following example logs into the DM Server and creates a new profile paper document on the primary library.

```
Dim App As New Application
 Dim Prof As Profile
```

```
' Log into the DM Server
If App.DST = "" Then Exit Sub

' Create a new blank profile on the primary library
Set Prof = App.PrimaryLibrary.CreateProfile("DEF_PROF")
' Display the profile form
If Prof.ShowProfile(SP_EDIT) = FR_DATACHANGED Then
' Set the document type to paper document and save the profile
Prof.Columns("$STORAGETYPE").Value = "P"
Prof.Save OF_NORMAL
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

"IProfile" on page 419

ILibrary::CreateProject

The method creates a new empty profiled folder on a library.

C++ Syntax

```
HRESULT CreateProject(IProject** ppProject);
```

Visual Basic Syntax

```
[ Set ppProject  = ] Library.CreateProject
```

Parameters

ppProject	[out, retval] Returns a Project object that specifies the created profiled folder.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Creates a new folder.

The folder created with the CreateProject method is an empty profiled folder. Use the Project methods AddDocument and AddSubProject to add components to this profiled folder.

The following example creates and saves a new folder on the current library.

```
Dim App As New Application
 Dim Prj As Project
 Set Prj = App.CurrentLibrary.CreateProject
 Prj.DisplayModal True
 Prj.Save OF_NORMAL
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

"IProject" on page 497

ILibrary::CreateSearch

The method creates a new search on the library by a specific search form.

C++ Syntax

```
HRESULT CreateSearch(BSTR bstrFormName, ISearch** ppSearch);
```

Visual Basic Syntax

```
[ Set ppSearch  = ] Library.CreateSearch (bstrFormName)
```

Parameters

bstrFormName	[in] Required. A string that specifies a search form on the library.
ppSearch	[out, retval] Returns a Search object that specifies the created search.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Creates a new search.

Creating a new search simply creates a blank Search object. The object is pre-filled out with the default search form values for the current user. Once the search object is created all of the properties and methods associated with the Search class can be used with the object. Before the new search can be saved it should be assigned a name.

The following example displays the dialog for selecting a search form, creates a new search by the form the user selected, and pops up the criteria form for the created search.

```
Dim App As New Application
 Dim Lib As Library
 Dim QS As Search
 Dim strFrm As String

 Set Lib = App.CurrentLibrary
 strFrm = Lib.SelectSearchForm(OF_SHOWFORM)
 Set QS = Lib.CreateSearch(strFrm)
 QS.ShowEditCriteriaForm True
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

"ISearch" on page 567

ILibrary::Description

The property returns the library description.

C++ Syntax

```
HRESULT Description(BSTR* pbstrDescription);
```

Visual Basic Syntax

```
[ pbstrDescription = ] Library.Description
```

Parameters

pbstrDescription	[out, retval] Returns a string that specifies the description of the library.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns a library description.

The following example sets the value of strDescription to the description of the DM primary library.

```
Dim App As New Application
 Dim strDescription As String
 strDescription = App.PrimaryLibrary.Description
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

ILibrary::Forms

The property returns a collection of forms from a library including the Quick Retrieve and Profile/QBE forms.

C++ Syntax

```
HRESULT Forms(IDOCSForms** ppForms);
```

Visual Basic Syntax

```
[ Set ppForms = ] Library.Forms
```

Parameters

ppForms	[out, retval] Returns a DOCSForms object that specifies a collection of library forms.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns a collection of all library forms.

The following example retrieves some properties of all forms from the current library.

```
Dim App As New Application
 Dim LibForms As DOCSForms

 Set LibForms = App.CurrentLibrary.Forms
 For i = 1 To LibForms.Count
 Dim strDef, strName, strTitle As String
 Dim bQR As Boolean

 With LibForms(i)
 strDef = .FormDefinition
 bQR = .IsQRForm
 strName = .Name
 strTitle = .Title
 End With

 ' Some operations with the form properties
```

```
...
Next
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

"IDOCSForms" on page 312

ILibrary::GetProfile

The method retrieves a document profile from a library by its number or name.

C++ Syntax

```
HRESULT GetProfile(VARIANT vDoc, IProfile** ppProfile);
```

Visual Basic Syntax

```
[ Set ppProfile  = ] Library.GetProfile (vDoc)
```

Parameters

vDoc	[in] Required. An expression that specifies a document profile. If a numeric expression, vDoc should specify the number of a document from the library. If a string expression, vDoc should specify the document name.
ppProfile	[out,retval] Returns a Profile object that specifies the requested document.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns a document profile by its number or name.

The following example retrieves the profile with number 20 from the current library.

```
Dim App As New Application
 Dim Doc As Profile
 Set Doc = App.CurrentLibrary.GetProfile(20)
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

"IProfile" on page 419

ILibrary::GetProject

The method retrieves a profiled folder from the library by its name or number.

C++ Syntax

```
HRESULT GetProject(VARIANT vProject, IProject** ppProject);
```

Visual Basic Syntax

```
[ Set ppProject  = ] Library.GetProject (vProject)
```

Parameters

vProject	[in] Required. An expression that specifies a profiled folder on the library. If a numeric expression, vProject should specify the document number of a folder's profile from the library. If a string expression, vProject should specify the project document name of a folder's profile.
ppProject	[out, retval] Returns a Project object that specifies the requested profiled folder.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns a profiled folder by its name or ID.

The following example retrieves the folder with number 8 from the current library.

```
Dim App As New Application
 Dim Prj As Project
 Set Prj = App.CurrentLibrary.GetProject(8)
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

"IProject" on page 497

ILibrary::IsCurrent

The property returns whether or not a library is current.

C++ Syntax

```
HRESULT IsCurrent(VARIANT_BOOL* pbCurrent);
```

Visual Basic Syntax

```
[ pbCurrent = ] Library.IsCurrent
```

Parameters

pbCurrent	[out, retval] Returns a Boolean value that specifies whether or not a library is current. True: A library is current. False: A library is not current.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns whether a library is current.

A library can be set as current using the MakeCurrent method.

Only one library can be current. When MakeCurrent sets one library as current, IsCurrent is automatically set to False for all other available libraries.

The following example creates a new folder in a library if it is current.

```
Dim Lib As Library
 ' Some operations with a library
 ...
 Dim Proj As Project
 If Lib.IsCurrent Then
 Set Proj = Lib.CreateProject
 ...
 End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

ILibrary::IsPrimary

The property returns whether or not a library is primary.

C++ Syntax

```
HRESULT IsPrimary(VARIANT_BOOL* pbPrimary);
```

Visual Basic Syntax

```
[ pbPrimary = ] Library.IsPrimary
```

Parameters

pbPrimary	[out, retval] Returns a Boolean value that specifies whether or not a library is primary. True: A library is primary. False: A library is not primary.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns whether a library is primary.

The following example checks to see if the user's current library is primary. If it is the primary library a check box is selected. If it is not the primary library the check box is cleared.

```
Dim App As New Application
 ...
 Sub cmdPrimary_Click()
 If App.CurrentLibrary.IsPrimary Then
 chkPrimary.Value = 1
 Else
 chkPrimary.Value = 0
 End If
 End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

ILibrary::LibraryGroup

The property returns the name of the library group a library belongs to.

C++ Syntax

```
HRESULT LibraryGroup(BSTR* pbstrGroup);
```

Visual Basic Syntax

```
[ pbstrGroup = ] Library.LibraryGroup
```

Parameters

pbstrGroup	[out, retval] Returns a string that specifies the requested library group. The string is empty if the library does not belong to any library group.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns the name of the library group a library belongs to.

The following example sets the value of strGroup to the name of the library group for the user's current library.

```
Dim App As New Application
 Dim strGroup As String
 strGroup = App.CurrentLibrary.LibraryGroup
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

ILibrary::MakeCurrent

The method makes a library to be the current library.

C++ Syntax

```
HRESULT MakeCurrent();
```

Visual Basic Syntax

```
Library.MakeCurrent
```

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Sets a library as current.

Only one library can be current. When MakeCurrent sets one library as current, IsCurrent is automatically set to False for all other available libraries.

The following example makes the primary library to be the current library.

```
Dim App As New Application
 App.PrimaryLibrary.MakeCurrent
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

ILibrary::Name

The property returns the name of a library.

C++ Syntax

```
HRESULT Name(BSTR* pbstrName);
```

Visual Basic Syntax

```
[ pbstrName = ] Library.Name
```

Parameters

pbstrName	[out, retval] Returns a string that specifies the library name.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns the name of a library.

The following example sets the value of strName to the name of the current library.

```
Dim App As New Application
 Dim strName As String
 strName = App.CurrentLibrary.Name
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

ILibrary::ProfileFile

The method imports a file into a library.

C++ Syntax

```
HRESULT ProfileFile(BSTR bstrFilePath, IProfile** ppProfile);
```

Visual Basic Syntax

```
[ Set ppProfile  = ] Library.ProfileFile (bstrFilePath)
```

Parameters

bstrFilePath	[in] Required. A string that specifies the full path to the file to be imported.
ppProfile	[out, retval] Returns a Profile object that specifies the created profile for the imported document. If the user did not create the profile by pressing the Cancel button in its form, the return value is NULL.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Imports a file into a library.

The following example imports a file into the primary library.

```
Dim App As New Application
 Dim Prof As Profile
 ' Import a file into the primary library
 Set Prof = App.PrimaryLibrary.ProfileFile("C:\MyDoc.doc")
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

"IProfile" on page 419

ILibrary::Projects

The property returns a collection of all public folders from the library.

C++ Syntax

```
HRESULT Projects(IProjects** ppProjects);
```

Visual Basic Syntax

```
[ Set ppProjects = ] Library.Projects
```

Parameters

ppProjects	[out, retval] Returns a Projects object that specifies a collection of all public folders from the library.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns a collection of all library projects.

The following example iterates through public folders from the current library.

```
Dim App As New Application
 Dim CurLibProject As Project

 For Each CurLibProject In App.CurrentLibrary.Projects
 ' Some operations with the folder
 ...
 Next
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

"IProjects" on page 534

ILibrary::Properties

The property returns a collection of all the persistent properties supported by a library.

C++ Syntax

```
HRESULT Properties(IProperties** ppProps);
```

Visual Basic Syntax

```
[ Set ppProps = ] Library.Properties
```

Parameters

ppProps	[out, retval] Returns a Properties object that specifies a collection of the library persistent properties.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns a collection of library persistent properties.

To get a specific Property object, the user can use the standard Properties::Item property of the collection. The following library persistent properties are supported:

Name	Type	Access	Description
DOCSParms.ProjectMembName	DT_String	r/o	An alias for “Project Membership”. Obsolete.
DOCSParms.ProjectName	DT_String	r/o	An alias for “Project”. Obsolete.
DOCSParms.ProjectsName	DT_String	r/o	An alias for “Projects”. Obsolete.
DOCSParms.SubprojectName	DT_String	r/o	An alias for “Sub-Projects”. Obsolete.
DOCSParms. NullDate	DT_Date	r/o	The date that used by DM Server as NULL date.
Dynamic.Unplugged	DT_Boolean	r/o	Indicates whether or not current mode is offline mode.
Dynamic.NetworkName	DT_String	r/o	Network library name. For online mode the same as library name.
DOCSParms. ProjectOrderSupported	DT_Boolean	r/o	Indicates whether or not ordering documents and sub-folders in folders is supported. Obsolete. Ordering is always supported.
Terminology	DT_Object	r/o	Returns collection (IProperties object) of items in Terminology table.

Parameters from DOCSPARMS table

Name	Type	Access	Description
DOCSParms.PreconnectSearchLibraries	DT_Boolean	r/o	Indicates whether or not libraries can be pre-connected.
DOCSParms.AcceptUserSupliedIdentification	DT_Boolean	r/o	Indicates whether or not user supplied identification allowed.
DOCSParms.MultiplePublishedVersions	DT_Boolean	r/o	Indicates whether or not multiple published versions allowed.
DOCSParms.CaptureKeystroke	DT_Boolean	r/o	Indicates whether or not cost recovery information should be saved.
DOCSParms.EnhancedACLSettings	DT_Boolean	r/o	Indicates whether or not enhanced ACL settings (Deny bits and “View Versions Published Only” access right) should be available.
DOCSParms.UniqueDocNames	DT_Boolean	r/o	Indicates whether or not document names must be unique.
DOCSParms.SaveToRemoteLibrary	DT_Boolean	r/o	Indicates whether the user can save documents to remote libraries.
DOCSParms.EditPrevVersion	DT_Boolean	r/o	Indicates whether the user can edit previous versions of a document.
DOCSParms.CreateNewVersions	DT_Boolean	r/o	Indicates whether the user can save changes as a new version of a document.
DOCSParms.DocDelete	DT_Boolean	r/o	Indicates whether the user can delete documents.

Name	Type	Access	Description
DOCSParms.DeleteContent	DT_Boolean	r/o	Indicates whether the user can delete document contents.
DOCSParms.QueueForDeletion	DT_Boolean	r/o	Indicates whether the user can queue document for deletion.
DOCSParms.Checkout	DT_Boolean	r/o	Indicates whether the user can check out documents.
DOCSParms.CopyInUse	DT_Boolean	r/o	Indicates whether the user can retrieve a copy of a document if the document is in use or checked out.
DOCSParms.CheckinOthers	DT_Boolean	r/o	Indicates whether the user can check in or unlock a document that was checked out or locked by another user.
DOCSParms.TemplateManager	DT_Boolean	r/o	Indicates whether the user can create and edit template documents.
DOCSParms.DisplayVersionList	DT_Boolean	r/o	Indicates whether or not a version selection list is always displayed when opening a document.
DOCSParms.ProfileLevelSecurity	DT_Boolean	r/o	Indicates whether profile level security should be used.
DOCSParms.AutoCleanupShadow	DT_Boolean	r/o	Indicates whether shadow should be cleaned at startup.
DOCSParms.EditVersionAuthor	DT_Boolean	r/o	Indicates whether the user can edit the author of a document version.
DOCSParms.EditVersionTypist	DT_Boolean	r/o	Indicates whether the user can edit the typist of a document version.

Name	Type	Access	Description
DOCSParms.DisableNative	DT_Boolean	r/o	Indicates whether the Open and SaveAs application native dialogs are disabled by the system administrator.
DOCSParms.DeleteVersions	DT_Boolean	r/o	Indicates whether the user can delete versions.
DOCSParms.SilentLogin	DT_Boolean	r/o	Indicates whether the user is allowed to auto login into DM.
DOCSParms.ShadowDocs	DT_Boolean	r/o	Indicates whether documents are shadowed to a local drive.
DOCSParms.ShadowSecuredDocs	DT_Boolean	r/o	Indicates whether secured documents are shadowed to a local drive.
DOCSParms.AllowMassUpdate	DT_Boolean	r/o	Indicates whether multiple plofiles can be updated.
DOCSParms.PublishVersion	DT_Boolean	r/o	Indicates whether the user can publish version.
DOCSParms.UnpublishVersion	DT_Boolean	r/o	Indicates whether the user can reset published version.
DOCSParms.MakeVersionReadOnly	DT_Boolean	r/o	Indicates whether the user can make version read-only.
DOCSParms.MakeVersionWritable	DT_Boolean	r/o	Indicates whether the user can remove read-only status for version.
DOCSParms.MakeReadOnly	DT_Boolean	r/o	Indicates whether the user can make read-only document or folder.

Name	Type	Access	Description
DOCSParms.MakeWritable	DT_Boolean	r/o	Indicates whether the user can remove read-only status for document or folder.
DOCSParms.CreateRelation	DT_Boolean	r/o	Indicates whether the user can create relation between profiled items.
DOCSParms.ShowRelation	DT_Boolean	r/o	Indicates whether the user can see related items.
DOCSParms.RemoveRelation	DT_Boolean	r/o	Indicates whether the user can remove relation.
DOCSParms.CreateFolder	DT_Boolean	r/o	Indicates whether the user can create folder.
DOCSParms.ManageRootFolders	DT_Boolean	r/o	Indicates whether the user can manage Public Folders.
DOCSParms.MaxVersions	DT_Long	r/o	The maximal number of versions a document can have.
DOCSParms.MaxSubversions	DT_Long	r/o	The maximal number of subversions a document can have.
DOCSParms.DefaultShadowRetention	DT_Long	r/o	Minumum age for delete shadow documents (days).
DOCSParms.MinimumDiskFree	DT_Long	r/o	Minimum local disk space for delete shadow documents (MB).

Parameters from GROUPS table for user's primary group

Name	Type	Access	Description
User.SaveToRemoteLibrary	DT_Boolean	r/o	Indicates whether the user can save documents to remote libraries.

Name	Type	Access	Description
User.EditPrevVersion	DT_Boolean	r/o	Indicates whether the user can edit previous versions of a document.
User.CreateNewVersions	DT_Boolean	r/o	Indicates whether the user can save changes as a new version of a document.
User.DocDelete	DT_Boolean	r/o	Indicates whether the user can delete documents.
User.DeleteContent	DT_Boolean	r/o	Indicates whether the user can delete document contents.
User.QueueForDeletion	DT_Boolean	r/o	Indicates whether the user can queue document for deletion.
User.Checkout	DT_Boolean	r/o	Indicates whether the user can check out documents.
User.CopyInUse	DT_Boolean	r/o	Indicates whether the user can retrieve a copy of a document if the document is in use or checked out.
User.CheckinOthers	DT_Boolean	r/o	Indicates whether the user can check in or unlock a document that was checked out or locked by another user.
User.TemplateManager	DT_Boolean	r/o	Indicates whether the user can create and edit template documents.
User.DisplayVersionList	DT_Boolean	r/o	Indicates whether or not a version selection list is always displayed when opening a document.
User.ProfileLevelSecurity	DT_Boolean	r/o	Indicates whether profile level security should be used.
User.AutoCleanupShadow	DT_Boolean	r/o	Indicates whether shadow should be cleaned at startup.
User.EditVersionAuthor	DT_Boolean	r/o	Indicates whether the user can edit the author of a document version.
User.EditVersionTypist	DT_Boolean	r/o	Indicates whether the user can edit the typist of a document version.

Name	Type	Access	Description
User.DisableNative	DT_Boolean	r/o	Indicates whether the Open and SaveAs application native dialogs are disabled by the system administrator.
User.DeleteVersions	DT_Boolean	r/o	Indicates whether the user can delete versions.
User.SilentLogin	DT_Boolean	r/o	Indicates whether the user is allowed to auto login into DM.
User.ShadowDocs	DT_Boolean	r/o	Indicates whether documents are shadowed to a local drive.
User.ShadowSecuredDocs	DT_Boolean	r/o	Indicates whether secured documents are shadowed to a local drive.
User.AllowMassUpdate	DT_Boolean	r/o	Indicates whether multiple plofiles can be updated.
User.PublishVersion	DT_Boolean	r/o	Indicates whether the user can publish version.
User.UnpublishVersion	DT_Boolean	r/o	Indicates whether the user can reset published version.
User.MakeVersionReadOnly	DT_Boolean	r/o	Indicates whether the user can make version read-only.
User.MakeVersionWritable	DT_Boolean	r/o	Indicates whether the user can remove read-only status for version.
User.MakeReadOnly	DT_Boolean	r/o	Indicates whether the user can make read-only document or folder.
User.MakeWritable	DT_Boolean	r/o	Indicates whether the user can remove read-only status for document or folder.
User.CreateDocument	DT_Boolean	r/o	Indicates whether the user can create documents.
User.ViewUnsecured	DT_Boolean	r/o	Indicates whether the user can access to unsecured documents.
User.CreateRelation	DT_Boolean	r/o	Indicates whether the user can create relation between profiled items.

Name	Type	Access	Description
User.ShowRelation	DT_Boolean	r/o	Indicates whether the user can see related items.
User.RemoveRelation	DT_Boolean	r/o	Indicates whether the user can remove relation.
User.CreateFolder	DT_Boolean	r/o	Indicates whether the user can create folder.
User.ManageRootFolders	DT_Boolean	r/o	Indicates whether the user can manage Public Folders.
User.MaxVersions	DT_Long	r/o	The maximal number of versions a document can have.
User.MaxSubversions	DT_Long	r/o	The maximal number of subversions a document can have.
User.DefaultShadowRetention	DT_Long	r/o	Minumum age for delete shadow documents (days).
User.MinimumDiskFree	DT_Long	r/o	Minimum local disk space for delete shadow documents (MB).

The following example determines whether the user can delete documents from the primary library.

```
Dim App As New Application
 Dim DocDelete As Boolean
 DocDelete = App.PrimaryLibrary.Properties("User.DocDelete")
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

"IProperties" on page 546

ILibrary::SearchByDefault

The property returns or sets whether or not a library is defined as a searchable library.

C++ Syntax

```
HRESULT SearchByDefault(VARIANT_BOOL* pbSearch);
HRESULT SearchByDefault(VARIANT_BOOL bSearch);
```

Visual Basic Syntax

```
[ pbSearch = ] Library.SearchByDefault
Library.SearchByDefault [ = bSearch ]
```

Parameters

pbSearch	[out, retval] A Boolean value that specifies whether or not a library is searchable.
bSearch	[in] A Boolean value that specifies whether or not a library is searchable. True: A library is searchable. False: A library is not searchable.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns/sets whether a library is marked as searchable by default.

The property determines whether or not a connected library will be searched. This property can be used to determine whether or not a library is searchable or it can be used to change the searchable status of a library.

The following example makes only the current library searchable. All other connected libraries are made unsearchable.

```
Dim App As New Application
 Dim Libs As Libraries

 Set Libs = App.Libraries
 For i = 1 To Libs.Count
 Libs(i).SearchByDefault = Libs(i).IsCurrent
 Next
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

ILibrary::Searches

The property returns a collection of all searches from the library including both saved and new searches.

C++ Syntax

```
HRESULT Searches(ISearches** ppSearches);
```

Visual Basic Syntax

```
[ Set ppSearches = ] Library.Searches
```

Parameters

ppSearches	[out, retval] Returns a Searches object that specifies a collection of all searches from the library.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns a collection of all searches from the library.

The following example iterates through all searches of the current library.

```
Dim App As New Application
 Dim CurLibSearch As Search

 For Each CurLibSearch In App.CurrentLibrary.Searches
 ' Some operations with the search
 ...
 Next
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

"ISearches" on page 591

ILibrary::SelectProfileForm

The method creates an empty document profile on a library with the profile form selected by the user and by a specific application.

C++ Syntax

```
HRESULT SelectProfileForm(
short sFlags, VARIANT vInfo, IProfile** ppProfile);
```

Visual Basic Syntax

```
[ Set ppProfile  = ] Library.SelectProfileForm (sFlags, vInfo)
```

Parameters

sFlags	[in] Required. An integer value that specifies how the method should be executed. The parameter can be a combination of the SPFFlags values.
vInfo	[in] Required. An expression that specifies the application the Profile object will be associated with. If a string expression, vInfo should specify either the application extension or the application identifier. An empty string identifies profile paper documents if the SPF_FROM_FILE flag is off. If on, the user will be able to specify the application when filling out the profile form. If an object, vInfo should specify a valid AppInfo object that identifies an application.
ppProfile	[out, retval] Returns a Profile object that specifies the created profile. If the method failed when creating a new profile (for instance, if the user pressed the Cancel button), the return value is NULL.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Displays the dialog for selecting a profile form.

The method creates a blank Profile object. Before creating the profile, a dialog is presented where the user can select a profile form associated with the specified application the profile will be based on. The dialog can be customized and its form should support the IProfileFormChooser interface. To display the dialog, the method calls up the SelectProfileForm method on IProfileFormChooser.

The created Profile object will be associated with the specified application. It is pre-filled out with the default profile values for the

current user. Besides, the typist field is also set to the current user. Values must be assigned for all required fields in the profile before it can be saved.

The following example creates a new blank profile for an MS Word document on the primary library.

```
Dim App As New Application

 Dim Prof As Profile

 ' Create a new blank profile for an MS Word document

 ' on the primary library

 Set Prof = App.PrimaryLibrary.SelectProfileForm( _

 SPF_FROM_FILE, "*.doc")
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

"IProfile" on page 419

"IAppInfo" on page 220

"IProfileFormChooser" on page 906

ILibrary::SelectSearchForm

The method displays the custom form that prompts the user to select a search form and returns the selected form's name.

C++ Syntax

```
HRESULT SelectSearchForm(short sFlags, BSTR* pbstrFormName);
```

Visual Basic Syntax

```
[ pbstrFormName  = ] Library.SelectSearchForm (sFlags)
```

Parameters

sFlags	[in] Required. An integer value that specifies how the method should be executed. The parameter can be a combination of the SSFFlags values.
pbstrFormName	[out,retval] Returns a string that contains the name of the form the user selected. If the user selected no search form by pressing the Cancel button, an empty string is returned.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Displays the dialog for selecting a QBE form.

The form can be customized and should support the ISearchFormChooser interface. To display the form, the method calls up the SelectSearchForm method on ISearchFormChooser.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

"ISearchFormChooser" on page 944

ILibrary::Shadow

The property returns whether or not documents from a library are shadowed.

C++ Syntax

```
HRESULT Shadow(VARIANT_BOOL* pbShadow);
```

Visual Basic Syntax

```
[ pbShadow  = ] Library.Shadow
```

Parameters

pbShadow	[out, retval] Returns a Boolean value that specifies whether or not documents from the library are shadowed. True: Documents are shadowed. False: Documents are not shadowed.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns whether a library is shadowed.

Shadowing a document is a feature of DM Extensions that allows the user to work with a local copy of a document rather than a network copy. This cuts down on network traffic and allows the user to keep working on a document if the network fails.

The following example checks to see if the user's current library is shadowed. If it is shadowed a check box is selected. If not, the check box is cleared.

```
Dim App As New Application
 ...
 Sub cmdShadow_Click()
 If App.CurrentLibrary.Shadow Then
 chkShadow.Value = 1
 Else
 chkShadow.Value = 0
 End If
 End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

ILibrary::ShadowPathTemplate

The property returns the template path to the Shadow directory including a drive letter.

C++ Syntax

```
HRESULT ShadowPathTemplate(BSTR* pbstrTemplate);
```

Visual Basic Syntax

```
[ pbstrTemplate = ] Library.ShadowPathTemplate
```

Parameters

pbstrTemplate	[out, retval] Returns a string that specifies the directory structure where shadowed documents are located on the client's local drive.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns the template path to the Shadow directory.

Shadowing a document is a feature of DM Extensions that allows the user to work with a local copy of a document rather than a network copy. This cuts down on network traffic and allows the user to keep working on a document if the network fails. The returned directory structure consists of a drive letter and path that may be specified using variables.

Requirements

DM: Requires PowerDOCS 3.5.1 later.

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

ILibrary::UserName

The property returns the DM user name used to connect to a library.

C++ Syntax

```
HRESULT UserName(BSTR* pbstrUserName);
```

Visual Basic Syntax

```
[ pbstrUserName = ] Library.UserName
```

Parameters

pbstrUserName	[out, retval] Returns a string that specifies the current user's name.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns the current user's name.

The UserName property contains the user's name inside DM.

The following example sets the value of strUserName to the current user's name.

```
Dim App As New Application
 Dim strUserName As String
 strUserName = App.CurrentLibrary.UserName
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

ILibrary::UserPrimaryGroup

The property returns the primary group of the connected user.

C++ Syntax

```
HRESULT UserPrimaryGroup(BSTR* pbstrPrimaryGroup);
```

Visual Basic Syntax

```
[ pbstrPrimaryGroup = ] Library.UserPrimaryGroup
```

Parameters

pbstrPrimaryGroup	[out, retval] Returns a string that specifies the name of the user's primary group on the library.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns the user's primary group.

A user's primary group determines what access the user has to DM utilities. By displaying the primary group information, a system administrator can quickly determine the source of many user problems in the product environment.

The following example expands the example presented in the previous section. It retrieves the user's name and the user's primary group on the current library.

```
Dim App As New Application
 Dim Lib As Library
 Set Lib = App.CurrentLibrary

 Dim strUserName, strPrimaryGroup As String
 strUserName = Lib.UserName
 strPrimaryGroup = Lib.UserPrimaryGroup
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

ILibrary2

The ILibrary2 interface extends functionality of a library object.

The interface inherits its definition from ILibrary.

Properties

ILibrary2 Properties	Description
ProfileUserDefaults	Returns/sets a profile with default user settings.
DefaultForm	Returns/sets a default form.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

ILibrary2::DefaultForm

Returns/sets the default form.

C++ Syntax

```
HRESULT DefaultForm(BSTR bstrAppName, BSTR* pbstrFormName);
HRESULT DefaultForm(BSTR bstrAppName, BSTR bstrFormName);
```

Visual Basic Syntax

```
[ pbstrFormName = ] Library2.DefaultForm (bstrAppName)
Library2.DefaultForm(bstrAppName) [ =  bstrFormName ]
```

Parameters

bstrAppName	[in] Specifies an application identifier.
pbstrFormName	[out, retval] Specifies an address, where the default form name should be returned. If returned string is NULL or empty, a form by default does not exists for an application specified by bstrAppName.
bstrFormName	[in] Specifies a form name, which should be saved as default for this application. If bstrFormName is NULL or empty string, a form by default will be deleted for an application specified by bstrAppName.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns/sets a default form.

If returned value is S_FALSE, methods of IOMErrInfo should be used for error description. Also "%Search" may be used as application name to obtain or setup a search form by default.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary2" on page 400

"IOMErrInfo" on page 412

ILibrary2::ProfileUserDefaults

Returns/sets a profile with default user settings.

C++ Syntax

```
HRESULT ProfileUserDefaults(
BSTR bstrAppName, BSTR bstrFormName, IProfile** ppProfile);
HRESULT ProfileUserDefaults(
BSTR bstrAppName, BSTR bstrFormName, IProfile* pProfile);
```

Visual Basic Syntax

```
[ Set ppProfile = ] Library2.ProfileUserDefaults (bstrAppName,
bstrFormName)
Library2.ProfileUserDefaults (bstrAppName, bstrFormName) [ =
pProfile ]
```

Parameters

bstrAppName	[in] Specifies an application identifier.
bstrFormName	[in] Specifies a form name.
ppProfile	[out, retval] Specifies an address, where a pointer to the IProfile interface of a Profile object with default user settings should be returned.
pProfile	[in] Specifies a pointer to the IProfile interface of a Profile object user settings, which should be saved as defaults settings.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns/sets a profile with default user settings.

If returned value is S_FALSE methods of IOMErrInfo should be used for error description.

Returned Profile object may be used to setup its columns and trustees. After saving, its settings will be used as default for these application and form. Profile settings may be saved by next two ways: by using the ProfileUserDefaults property (propput method) of a Library object, or by using the Save method of retuned Profile object. Furthermore, returned profile object may be configured with a user interface by

calling the ShowProfile (SP_NOCHECK_REQUIRED | SP_EDIT) method.

Also "%Primary" may be used as application name to configure <Primary> form or "%Search" to configure search form. bstrFormName is not used for the <Primary> form. Default user settings cannot be saved for search form.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary2" on page 400

"IProfile" on page 419

"IOMErrInfo" on page 412

ILibrary3

Interface ILibrary3 is implemented to extend functionality of Library object. The interface inherits its definition from ILibrary2.

Properties

ILibrary3 Properties	Description
ILibrary3::PublicFolders	Returns a collection of all public folders.
ILibrary3::FavoritesFolders	Returns a collection of all favorites folders.

Methods

ILibrary3 Methods	Description
ILibrary3::ProfileDirectory	Profile file directory.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import "fsplugin.dll"

ILibrary3::FavoritesFolders

Returns collection of favorites folders.

C++ Syntax

```
HRESULT FavoritesFolders(  IDOComponents** ppFolders);
```

Visual Basic Syntax

```
[ Set ppFolders = ] Library3.FavoritesFolders
```

Parameters

ppFolders	[out, retval] Returns DOComponents object that contain all available public folders.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns a collection of all favorites folders.

Requirements

DM: Requires Humminbird DM 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

"ILibrary2" on page 400

"IDOComponents" on page 321

ILibrary3::ProfileDirectory

Import directory (files and sub-directories) into DM.

C++ Syntax

```
HRESULT ProfileDirectory(  short sFlags,   BSTR bstrFilePath,
VARIANT vParent,   IProject** ppProject);
```

Visual Basic Syntax

```
[ Set ppProject = ] Library2.ProfileDirectory (sFlags,
```

bstrFilePath, *vParent*)

Parameters

sFlags	[in] An integer value that specifies how the method should be executed. The parameter can be a combination of the MethodsFlags values. If OF_SHOWFORM flag is specified Directory chooser dialog will be displayed where user can specify directory to import.
bstrFilePath	[in] A string value that specifies what directory should be imported. This parameter is ignored if OF_SHOWFORM is specified in sFlags parameter.
vParent	[in, optional] An expression that allow to specify parent folder. Imported directory will be added to this folder as sub-folder.
ppProject	[out, retval] Returns created folder.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Profile file directory.

Requirements

DM: Requires Humminbird DM 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

"ILibrary2" on page 400

ILibrary3::PublicFolders

Returns collection of profiled folders.

C++ Syntax

```
HRESULT PublicFolders(  IDOComponents** ppFolders);
```

Visual Basic Syntax

```
[ Set ppFolders = ] Library3.PublicFolders
```

Parameters

ppFolders	[out, retval] Returns DOComponents object that contain all available public folders.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns a collection of all public folders.

Requirements

DM: Requires Humminbird DM 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

"ILibrary2" on page 400

"IDOComponents" on page 321

IMonitorFile

The MonitorFile object represents a filename mask DM Extensions uses while monitoring directories through Directory Monitoring integration. Filename masks specify files DM Extensions will either look for or exclude when monitoring files.

MonitorFile objects may be retrieved through the MonitorFiles collection on a launch method.

Properties

IMonitorFile Properties	Description
FileNameMask	Returns the file name mask.
Application	Returns the application for a monitor file.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IMonitorFiles" on page 409

IMonitorFile::Application

The property returns the identifier of the application. For each application own file mask is corresponded. Any filename that matches the monitor file's mask will be marked as belongs to this application.

C++ Syntax

```
HRESULT Application(BSTR* pbstrAppName);
```

Visual Basic Syntax

```
[ pbstrAppName = ] MonitorFile.Application
```

Parameters

pbstrAppName	[out, retval] Returns a string that specifies the application identifier.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the application for a monitor file.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IMonitorFile" on page 406

IMonitorFile::FileNameMask

The property returns the filename mask a monitor file represents.

C++ Syntax

```
HRESULT FileNameMask(BSTR* pbstrMask);
```

Visual Basic Syntax

```
[ pbstrMask = ] MonitorFile.FileNameMask
```

Parameters

pbstrMask	[out, retval] Returns a string that specifies the filename mask.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the file name mask.

A filename mask is a usual filename that contains the wildcard characters, ? and *, that stands for any character or a sequence of characters, respectively; for instance, *.TXT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IMonitorFile" on page 406

IMonitorFiles

The MonitorFiles object represents a collection of filename masks DM Extensions uses to monitor directories if a launch method uses Directory Monitoring integration. The collection may be retrieved with the MonitorFiles property on a launch method.

Properties

IMonitorFiles Properties	Description
Item	Returns a specific item of a collection either by position or by key.
Count	Returns the number of items in a collection.
_NewEnum	Returns an object that supports IEnumVARIANT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILaunchMethod" on page 331

IMonitorFiles::Count

The property returns the number of items in a collection.

C++ Syntax

```
HRESULT Count(long* plCount);
```

Visual Basic Syntax

```
[ plCount = ] MonitorFiles.Count
```

Parameters

plCount	[out, retval] Returns a long that specifies the number of items in a collection.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the number of items in a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IMonitorFiles" on page 409

IMonitorFiles::Item

The property returns the specific member of a collection either by position or by key.

C++ Syntax

```
HRESULT Item(VARIANT vIndex, IMonitorFile** ppFile);
```

Visual Basic Syntax

```
[ Set ppFile = ] MonitorFiles.Item (vIndex)
```

Parameters

vIndex	[in] Required. An expression that specifies the member position in the collection. If a numeric expression, vIndex should be a number from 1 to the value of the collection's Count property. If a string expression, vIndex should correspond to a filename mask.
ppFile	[out, retval] Returns a MonitorFile object that specifies the requested item.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the specific item of a collection either by position or by key.

If the value provided as vIndex does not match any existing member of the collection, an error occurs. Item is the default property for a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IMonitorFiles" on page 409

IMonitorFiles::_NewEnum

The _NewEnum property returns the IEnumVARIANT interface on an object that can be used to enumerate the collection.

C++ Syntax

```
HRESULT _NewEnum(LPUNKNOWN* ppEnum);
```

Parameters

ppEnum	[out, retval] On return, an IEnumVARIANT object reference that can be used to enumerate the collection.

Remarks

Returns an object that supports IEnumVARIANT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IMonitorFiles" on page 409

IOMErrInfo

This interface is the base interface for many other DMO interfaces. It provides common properties that you access through the other interfaces to get error information.

Properties

IOMErrInfo Properties	Description
ErrCode	Returns a numeric code specifying an error.
ErrText	Returns a text description of an error.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

IOMErrInfo::ErrCode

The property returns the numeric error code from the last call of a method on an object.

C++ Syntax

```
HRESULT ErrCode(long* lCode);
```

Visual Basic Syntax

```
[ lCode = ] OMErrInfo.ErrCode
```

Parameters

lCode	[out, retval] Returns a long that specifies the requested error code. It can be one of the OMErrorCodes values.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns the numeric code specifying an error.

All interfaces that inherit their definition from IOMErrInfo have this property.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IOMErrInfo" on page 412

IOMErrInfo::ErrText

This property returns a text description of the error from the last call of a method on an object.

C++ Syntax

```
HRESULT ErrText(BSTR* pbstrText);
```

Visual Basic Syntax

```
[ pbstrText = ] OMErrInfo.ErrText
```

Parameters

pbstrText	[out, retval] Returns a string that contains the text error description.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns a text description of the error.

All interfaces that inherit their definition from IOMErrInfo have this property. If the error code of the last call is OM_SUCCESS the returned string is empty.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IOMErrInfo" on page 412

IOMObjects

The interface is implemented by a collection of Document Management plug-in objects. This collection is used to pass the currently selected items to event handlers. The implementation should filter all unsupported IBCObjectArray's items (those that cannot be converted to ILibrary, IProfile, ISearch, or IProject) passed as a parameter to menu-related methods to build the collection.

Properties

IOMObjects Properties	Description
Item	Returns a specific item of a collection either by position or by key.
Count	Returns the number of items in a collection.
_NewEnum	Returns an object that supports IEnumVARIANT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IBCObjectArray" on page 91

"ILibrary" on page 360

"IProfile" on page 419

"ISearch" on page 567

"IProject" on page 497

IOMObjects::Count

The property returns the number of items in a collection.

C++ Syntax

```
HRESULT Count(long* plCount);
```

Visual Basic Syntax

```
[ plCount = ] OMObjects.Count
```

Parameters

plCount	[out, retval] Returns a long that specifies the number of items in a collection.

Return Values

S_OK	The operation succeeded.
E_INVALIDARG	plCount is not a valid pointer.

Remarks

Returns the number of items in a collection.

It is the standard property of a collection. It returns the number of objects in a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IOMObjects" on page 414

IOMObjects::Item

This standard default collection property returns a specific item of the collection by its position.

C++ Syntax

```
HRESULT Item(VARIANT vIndex, LPDISPATCH** ppObj);
```

Visual Basic Syntax

```
[ ppObj = ] OMObjects.Item (vIndex)
```

Parameters

vIndex	[in] Indicates an item in the collection. The parameter is the item position in the collection. The parameter has to be of type VT_I2 or VT_I4.
ppObj	[out, retval] An address of the IDispatch* pointer variable that receives the interface pointer to a Document Management object. Upon successful return, this object implements the ILibrary, IProfile, ISearch, or IProject interface. Each time the collection receives a successful call to Item, it should increase the reference count on the *ppObj pointer the property returns. It is the caller's responsibility to call IUnknown::Release when it is done with the pointer. If an error occurs, the implementation should set *ppObj to NULL

Return Values

S_OK	The operation succeeded.
E_OUTOFMEMORY	There is not enough memory to complete the operation.
E_INVALIDARG	ppObj is not a valid pointer.
E_UNEXPECTED	Unexpected error occurred.
DISP_E_BADINDEX	vIndex is out of range.
DISP_E_TYPEMISMATCH	vIndex is not of valid type.

Remarks

Returns the specific item of a collection either by position or by key.

It is the standard property of a collection. This property is default, i.e. it has the DISPID_VALUE (0) identifier. It returns the specific item in a collection by its position.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IOMObjects" on page 414

"ILibrary" on page 360

"IProfile" on page 419

"ISearch" on page 567

"IProject" on page 497

IOMObjects::_NewEnum

The _NewEnum property returns the IEnumVARIANT interface on an object that can be used to enumerate the collection.

C++ Syntax

```
HRESULT _NewEnum(LPUNKNOWN* ppEnum);
```

Parameters

ppEnum	[out, retval] An address of the IUnknown* pointer variable that receives the interface pointer to the enumeration object. Each time the collection receives a successful call to _NewEnum, it should increase the reference count on the *ppEnum pointer the property returns. It is the caller's responsibility to call IUnknown::Release when it is done with the pointer. If an error occurs, the implementation should set *ppEnum to NULL

Return Values

S_OK	The operation succeeded.
E_OUTOFMEMORY	There is not enough memory to complete the operation.
E_INVALIDARG	ppEnum is not a valid pointer.
E_UNEXPECTED	Unexpected error occurred.

Remarks

It is the standard property of a collection. It returns an enumeration object, which implements the IEnumVariant interface. This property should have the DISPID_NEWENUM identifier. The user can use the enumeration interface to step through the objects in the collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IOMObjects" on page 414

IProfile

The Profile object represents a DM profile, a record in the SQL database that contains information about a single document. It is the means by which the user retrieves, views, and manipulates a document.

Each Profile object visually presented by its form. A profile form depends on the primary group of the user working with the system and the application a document should be associated with. Moreover, every application can have different profile forms. A profile form that will be used to store information about the document should be specified when profiling the document.

The Profile object inherits its definition from the IOMErrInfo interface so its implementation also includes all the properties ofIOMErrInfo in addition to the ones described below.

When to Use

A Profile object may be created with the CreateObject function or the Dim statement with the New keyword. To initialize the object the SetDocument method is to be called up before any other method or property. The following example shows the Visual Basic syntax for creating a DOCSObjects.Profile object:

```
Sub Form_Load()
 ...
 Dim Prof As Object
 ' Create a DOCSObjects.Profile object
 Set Prof = CreateObject("DOCSObjects.Profile")
 Prof.SetDocument "Test", 20
 ...
 End Sub
```

Methods

IProfile Methods	Description
IProfile::Open	Opens a document.
IProfile::Preview	Previews a document.
IProfile::PrintDoc	Prints a document.
IProfile::Mail	Sends a document by e-mail.
IProfile::Export	Exports a document to a file.
IProfile::LockProfile	Locks a document.
IProfile::UnlockProfile	Unlocks a document.
IProfile::OpenInCompatibleApp	Opens a document in a compatible application.
IProfile::SetDocument	Initializes a document object.

IProfile Methods	Description
IProfile::SelectForEdit	Selects a document for editing.
IProfile::ReleaseDoc	Releases a document.
IProfile::Fetch	Fetches a document.
IProfile::Save	Saves a document.
IProfile::Delete	Deletes a document.
IProfile::Copy	Copies a document.
IProfile::AddDocumentActivity	Adds an activity to the document's history.
IProfile::ShowProfile	Displays a document profile.
IProfile::ShowHistoryForm	Displays a document history form.
IProfile::ShowProjectsMembershipForm	Displays a projects membership form.
IProfile::ShowVersionChooserForm	Displays a form for selecting a document version/ attachment.
IProfile::ShowSaveAsOptionsForm	Displays a Save As options form.
IProfile::ShowAccessForm	Displays a document access form.
IProfile::CreateVersionFromFile	Creates a new version/attachment from an external file.
IProfile::DeleteAllVersions	Deletes all versions of a document.

Properties

IProfile Properties	Description
IProfile::Library	Returns the library of a document.
IProfile::Name	Returns the name of a document.
IProfile::DocNumber	Returns the number of a document.
IProfile::Versions	Returns a collection of document versions/ attachments.
IProfile::CurrentVersion	Returns the current version of a document.
IProfile::WorkingVersion	Returns the version locked for editing.
IProfile::Columns	Returns a collection of profile form columns.
IProfile::FormName	Returns the name of the document form.
IProfile::Trustees	Returns a collection of document trustees.

IProfile Properties	Description
IProfile::ParentProjects	Returns a collection of parent projects.
IProfile::CheckoutPath	Returns the check-out path.
IProfile::Icon	Returns the document icon.
IProfile::LastVersion	Returns the last version of a document.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "fsplugin.dll"

See Also

"IOMErrInfo" on page 412

IProfile::AddDocumentActivity

The method adds a document activity to the document's activity log.

C++ Syntax

```
HRESULT AddDocumentActivity(  VARIANT vDescription,
IActivityInfo* pActivityInfo);
```

Visual Basic Syntax

```
Profile.AddDocumentActivity (vDescription, pActivityInfo)
```

Parameters

vDescription	[in] Required. An expression that specifies the activity description. If a string expression, vDescription should contain description or comment to be added to the activity log. In this case, the activity is user-defined. If a numeric expression, vDescription should be one of the activity identifiers from OMActivityConsts.
pActivityInfo	[in, optional] Optional. An ActivityInfo object that specifies activity parameters.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Adds an activity to the document's history.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

"IActivityInfo" on page 202

"OMActivityConsts Enumeration" on page 690

IProfile::CheckoutPath

The property returns the full path of the checked-out version of a document.

C++ Syntax

```
HRESULT Pages(  BSTR* pbstrPath);
```

Visual Basic Syntax

```
[ pbstrPath = ] Profile.Pages
```

Parameters

pbstrPath	[out, retval] Returns a string that specifies the checked-out path. If the document is not checked-out, the string is empty.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns the check-out path.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

IProfile::Columns

The property returns a collection of all the columns that can be accessed through a profile.

C++ Syntax

```
HRESULT Columns(  IColumns** ppColumns);
```

Visual Basic Syntax

```
[ Set ppColumns = ] Profile.Columns
```

Parameters

ppColumns	[out, retval] Returns a Columns object that represents the collection of profile columns.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The Columns property provides a way to find out information and change values for fields that relate to the document profile. Using the Columns property is the only way to create or "get to" a column object.

Example

The following example changes the name of the profile with number 20.

```
Dim App As New Application
Dim Prof As Profile
Set Prof = App.PrimaryLibrary.GetProfile(20)
If Not Prof Is Nothing Then
If Prof.SelectForEdit(OF_NORMAL + SFE_LOCK_PROFILE) = _
SFER_OK Then
Prof.Columns("$DOCNAME").Value = "New Name"
Prof.Save OF_NORMAL
Prof.ReleaseDoc
If Prof.Name <> "New Name" Then
MsgBox "Document name hasn't changed"
End If
End If
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

"IColumns" on page 285

IProfile::Copy

The method copies the specified version of a document to another profile.

C++ Syntax

```
HRESULT Copy(  IVersion* pVersion,  IProfile** ppProfile);
```

Visual Basic Syntax

```
[ Set ppProfile  = ] Profile.Copy (pVersion)
```

Parameters

pVersion	[in] Required. A Version object that specifies a document's version to be copied.
ppProfile	[out, retval] Returns a Profile object that specifies a newly created profile. If the operation was cancelled or an error occurred, the return value is NULL.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Copies a document.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

IProfile::CreateVersionFromFile

The method creates a new version, subversion, or attachment of a document and uploads a file to this newly created object.

C++ Syntax

```
HRESULT CreateVersionFromFile(  short sFlags,  BSTR bstrPath,
VARIANT vAuthor,  VARIANT vTypist,  VARIANT vLabel,  VARIANT
vComments,  IVersion** ppVersion);
```

Visual Basic Syntax

```
[ Set ppVersion  = ] Profile.CreateVersionFromFile (sFlags,
bstrPath, vAuthor, vTypist, vLabel, vComments)
```

Parameters

sFlags	[in] Required. An integer value that specifies how the method should be executed. The parameter can be a combination of the MethodsFlags and VersionPropertiesFlags values.
bstrPath	[in] Required. A string that specifies the full path to the file to be uploaded.
vAuthor	[in, optional] Optional. A string of the version author's name.
vTypist	[in, optional] Optional. A string of the version typist's name.
vLabel	[in, optional] Optional. A string that specifies the version label. The parameter is used when creating a new subversion or attachment. In this case, for subversions it should contain the number of its parent version (e.g., "1"). For attachments vLabel should specify the attachment label. The attachment label is a three characters long string. The first character should be in the A-Z (uppercase) range. The others should be any uppercase Latin letter or digit.
vComments	[in, optional] Optional. A string that indicates the version comments.
ppVersion	[out, retval] Returns a Version object that specifies the newly created version, subversion, or attachment. If the operation was cancelled or an error occurred, the return value is NULL.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The method is called when a new version, subversion, or attachment is created from an existing external file. It is the only way to create an attachment on a profile.

The method will fail if the document profile is not selected for editing.

Example

The following example creates a new attachment for the document with number 20 on the primary library.

```
Dim App As New Application
Dim Prof As Profile, Ver As Version
Set Prof = App.PrimaryLibrary.GetProfile(20)
If Not Prof Is Nothing Then
Prof.SelectForEdit OF_NORMAL + SFE_LOCK_DOCUMENT
Set Ver = Prof.CreateVersionFromFile( _
OF_SHOWFORM, "C:\MyDoc.doc", , , "TXT")
Prof.ReleaseDoc
If Not Ver Is Nothing Then
MsgBox "New attachment was successfully created."
Else
MsgBox "New attachment wasn't created."
End If
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

"IVersion" on page 625

"MethodsFlags Enumeration" on page 689

"VersionPropertiesFlags Enumeration" on page 705

IProfile::CurrentVersion

The property returns the current version of a document.

C++ Syntax

```
HRESULT CurrentVersion(  IVersion** ppVersion);
```

Visual Basic Syntax

```
[ Set ppVersion = ] Profile.CurrentVersion
```

Parameters

ppVersion	[out, retval] Returns a Version object that specifies the document's current version.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The current version of a document is either its last version or most recently edited version.

Example

The following example logs into the DM Server, creates a new MS Word document on the primary library, and uploads a local file to the document's contents.

```
Dim App As New Application
Dim Prof As Profile, Ver As Version

' Log into the DM Server
If App.DST = "" Then Exit Sub

' Create a new profile on the primary library
Set Prof = App.PrimaryLibrary.SelectProfileForm(OF_NORMAL, _
"MS WORD")
If Not Prof Is Nothing Then
' Display the profile form and save it
Prof.ShowProfile SP_EDIT
If Prof.Save(OF_NORMAL) Then
' Set the current version of the document to
' a local MS Word file
Set Ver = Prof.CurrentVersionVer.LockVersionForEdit OF_NORMAL
FileCopy "C:\MyDoc.doc", Ver.FilePath
```

```
Ver.UploadFile Nothing
Ver.UnlockVersion OF_NORMAL
End If
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

"IVersion" on page 625

IProfile::Delete

The method deletes a document.

C++ Syntax

```
HRESULT Delete(  short sFlags,  VARIANT_BOOL* pbSuccess);
```

Visual Basic Syntax

```
[ pbSuccess  = ] Profile.Delete (sFlags)
```

Parameters

sFlags	[in] Required. An integer value that specifies how the method should be executed. The parameter can be a combination of the MethodsFlags and ProfileDeleteFlags values.
pbSuccess	[out, retval] Returns a Boolean value that specifies whether or not the operation succeeded. True: Document was successfully deleted. False: Document was not deleted.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

After deleting, a Profile object becomes undefined, i.e. no properties or methods can be called on the object before initializing it again with the SetDocument method.

The user must have enough rights to be able to delete a document.

Example

The following example deletes the document with number 20 on the primary library.

```
Dim App As New Application
Dim Prof As Profile
Set Prof = App.PrimaryLibrary.GetProfile(20)
If Not Prof Is Nothing Then
If Prof.Delete(OF_NORMAL) Then
MsgBox "Profile was successfully deleted."
Else
MsgBox "Profile wasn't deleted."
End If
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

"IProfile::SetDocument" on page 455

"MethodsFlags Enumeration" on page 689

"ProfileDeleteFlags Enumeration" on page 694

IProfile::DeleteAllVersions

The method allows to delete all versions of a document.

C++ Syntax

```
HRESULT DeleteAllVersions(  VARIANT_BOOL* pbSuccess);
```

Visual Basic Syntax

```
[ pbSuccess  = ] Profile.DeleteAllVersions
```

Parameters

pbSuccess	[out, retval] Returns a Boolean value that specifies whether or not the operation succeeded. True: Versions of a document were successfully deleted. False: Versions of a document were not deleted.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

IProfile::DocNumber

The property returns the document number of a profile.

C++ Syntax

```
HRESULT DocNumber(  long* plDocNumber);
```

Visual Basic Syntax

```
[ plDocNumber = ] Profile.DocNumber
```

Parameters

plDocNumber	[out, retval] Returns a long that specifies the document number.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The document number will be returned if this is an existing document. The property will return zero (0) if this is a new document that has not been profiled. The document number is unique throughout the document library and uniquely identifies a profile.

Example

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

IProfile::Export

The method exports the contents of a document to a file.

C++ Syntax

```
HRESULT Export(  VARIANT_BOOL* pbSuccess);
```

Visual Basic Syntax

```
[ pbSuccess  = ] Profile.Export
```

Parameters

pbSuccess	[out, retval] Returns a Boolean value that specifies whether or not the operation succeeded. True: The document was successfully exported. False: Operation was cancelled or it was impossible to export the document.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The user will be asked to select a file to export the document through the standard SaveAs file dialog.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

IProfile::Fetch

The method fetches the document data.

C++ Syntax

```
HRESULT Fetch();
```

Visual Basic Syntax

```
Profile.Fetch
```

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Fetches the document data.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

IProfile::FormName

The property returns the name of the form by which a profile was created.

C++ Syntax

```
HRESULT FormName(  BSTR* pbstrName);
```

Visual Basic Syntax

```
[ pbstrName = ] Profile.FormName
```

Parameters

pbstrName	[out, retval] Returns a string with the form name.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns the name of the document form.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

IProfile::Icon

The property returns the icon of a document.

C++ Syntax

```
HRESULT Icon(  long* plIcon);
```

Visual Basic Syntax

```
[ plIcon = ] Profile.Icon
```

Parameters

plIcon	[out, retval] Returns a long that specifies the HICON value of the document's icon. It is caller responsibility to destroy returned icon.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns the document icon.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

IProfile::LastVersion

The property returns the last version of a document.

C++ Syntax

```
HRESULT LastVersion(  IVersion** ppVersion);
```

Visual Basic Syntax

```
[ Set ppVersion = ] Profile.LastVersion
```

Parameters

ppVersion	[out, retval] Returns a Version object that specifies the document's last version.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns the last version of a document.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

"IVersion" on page 625

IProfile::Library

The property returns the Library object on which the profile record exists.

C++ Syntax

```
HRESULT Library(  ILibrary** ppLibrary);
```

Visual Basic Syntax

```
[ Set ppLibrary = ] Profile.Library
```

Parameters

ppLibrary	[out, retval] Returns a Library object that specifies the library the profile is assigned to.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns the library of a document.

Example

The following example makes the parent Library object of a profile to be the current library.

```
Dim Prof As Profile
' Some operations with the profile
...
Prof.Library.MakeCurrent
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

"ILibrary" on page 360

IProfile::LockProfile

The method locks a document profile.

C++ Syntax

```
HRESULT LockProfile(  short sFlags,  DATE dateReturn,  BSTR
bstrComments,  VARIANT_BOOL* pbSuccess);
```

Visual Basic Syntax

```
[ pbSuccess  = ] Profile.LockProfile (sFlags, dateReturn,
bstrComments)
```

Parameters

sFlags	[in] Required. An integer value that specifies how the method should be executed. The parameter can be a combination of the MethodsFlags values.
dateReturn	[in] Required. A date that specifies when the profile is to be checked in.
bstrComments	[in] Required. A string of comments.
pbSuccess	[out, retval] Returns a Boolean value that specifies whether or not the operation succeeded. True: The profile was successfully locked. False: The operation was cancelled or it was impossible to lock the profile.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

If the show form bit is on or the data are insufficient, the check-out dialog pops up. The dialog can be customized and its form should support theIGroupCheckoutForm interface. To display the form, the method calls up theShowGroupCheckoutForm method onIGroupCheckoutForm.

Example

The following example locks the document with number 20 on the primary library.

```
Dim App As New Application
Dim Prof As Profile
Set Prof = App.PrimaryLibrary.GetProfile(20)
If Not Prof Is Nothing Then
If Prof.LockProfile(OF_SHOWFORM, DateAdd("d", 10, Now), "") Then
Dim LockTime As Date
```

```
LockTime = Now

' Some operations with the Prof
...

Prof.UnlockProfile OF_SHOWFORM, _
DateDiff("n", LockTime, Now), "C:\TEMP", UF_ NEWVERSION
End If
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

"IGroupCheckoutForm" on page 881

"IGroupCheckoutForm::ShowGroupCheckoutForm" on page 882

"MethodsFlags Enumeration" on page 689

IProfile::Mail

The method sends a document by e-mail.

C++ Syntax

```
HRESULT Mail(  VARIANT_BOOL* pbSuccess);
```

Visual Basic Syntax

```
[ pbSuccess  = ] Profile.Mail
```

Parameters

pbSuccess	[out, retval] Returns a Boolean value that specifies whether or not the operation succeeded. True: The document was successfully e-mailed. False: The operation was cancelled or it was impossible to e-mail the document.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Sends a document by e-mail.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

IProfile::Name

The property returns the document's name.

C++ Syntax

```
HRESULT Name(  BSTR* pbstrName);
```

Visual Basic Syntax

```
[ pbstrName = ] Profile.Name
```

Parameters

pbstrName	[out, retval] Returns a string with the document's name.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The Name property's value is the value of the DOCNAME column in the Profile table. To assign a document name or change an existing document's name use the columns property to assign a value to a field.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

IProfile::Open

The method opens a document in the associated application in the edit mode.

C++ Syntax

```
HRESULT Open(  long lMode,  VARIANT_BOOL* pbSuccess);
```

Visual Basic Syntax

```
[ pbSuccess  = ] Profile.Open (lMode)
```

Parameters

lMode	[in] Required. A long that specifies the mode in which the document will be opened. The parameter can be a combination of the AccessMode values.
pbSuccess	[out, retval] Returns a Boolean value that specifies whether or not the operation succeeded. True: The operation succeeded. False: The operation failed because of an internal error or the user cancelled the operation.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed. Unexpected error occurs.

Remarks

The document will be opened in its native application (the application specified in its profile record). If the application is not already running, this method will start the application and open the document.

Example

The following example opens a read-only copy of the document with number 20.

```
Dim App As New Application
Dim Prof As Profile
Set Prof = App.PrimaryLibrary.GetProfile(20)
If Not Prof Is Nothing Then
Prof.Open AM_READ + AM_COPY
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

"AccessMode Enumeration" on page 679

IProfile::OpenInCompatibleApp

The method opens a document in an application compatible with its native application.

C++ Syntax

```
HRESULT OpenInCompatibleApp(  short sFlags,  long lMode,  VARIANT
vApp,  VARIANT_BOOL* pbSuccess);
```

Visual Basic Syntax

```
[ pbSuccess  = ] Profile.OpenInCompatibleApp (sFlags, lMode, vApp)
```

Parameters

sFlags	[in] Required. An integer value that specifies how the method should be executed. The parameter can be a combination of the MethodsFlags values.
lMode	[in] Required. A long that specifies the mode in which the document will be opened. The parameter can be a combination of the AccessMode values.
vApp	[in,optional] Optional. An expression that specifies a compatible application used to open the document. If a string expression, vApp should specify the application identifier (e.g., "MS WORD"). If an object, vApp should specify a valid AppInfo object that identifies an application compatible with the application associated with the document.
pbSuccess	[out,retval] Returns a Boolean value that specifies whether or not the operation succeeded. True: The operation succeeded. False: The operation failed because of an internal error or the user cancelled the operation.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The document will be opened in an application compatible with its native application (the application specified in its profile record). If vApp is not specified and OF_SHOWFORM flag is specified in the sFlags parameter the list of compatible applications will pop up allowing the user to select one of them. If the compatible application is

not already running, this method will start the application and open the document.

Example

The following example opens the document with number 20 in an application that the user can choose from the list of compatible applications.

```
Dim App As New Application

Dim Prof As Profile

Set Prof = App.PrimaryLibrary.GetProfile(20)

If Not Prof Is Nothing Then

Prof.OpenInCompatibleApp OF_SHOWFORM, AM_READWRITE

End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

"IAppInfo" on page 220

"MethodsFlags Enumeration" on page 689

"AccessMode Enumeration" on page 679

IProfile::ParentProjects

The property returns a collection of all folders a profile belongs to.

C++ Syntax

```
HRESULT ParentProjects(  IProjects** ppProjects);
```

Visual Basic Syntax

```
[ Set ppProjects = ] Profile.ParentProjects
```

Parameters

ppProjects	[out, retval] Returns a Projects object that represents a collection of profile parent folders.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns a collection of parent projects.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

"IProjects" on page 534

IProfile::Preview

The method views the contents of a document.

C++ Syntax

```
HRESULT Preview(  VARIANT vReserved,  VARIANT_BOOL* pbSuccess);
```

Visual Basic Syntax

```
[ pbSuccess  = ] Profile.Preview (vReserved)
```

Parameters

vReserved	[in, optional] Optional. A reserved parameter that should not be specified.
pbSuccess	[out, retval] Returns a Boolean value that specifies whether or not the operation succeeded. True: The document's contents were successfully displayed by the viewer. False: The operation was cancelled or it was impossible to display the document's contents in the viewer.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

If the application has a defined viewer set in Application Maintenance, then that viewer will be used. Otherwise the DM Extensions viewer will be used to display the contents of the document.

Example

The following example displays the contents of the document with number 20.

```
Dim App As New Application
Dim Prof As Profile
Set Prof = App.PrimaryLibrary.GetProfile(20)
If Not Prof Is Nothing Then
Prof.PreviewEnd If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

IProfile::PrintDoc

The method prints a document.

C++ Syntax

```
HRESULT PrintDoc(  VARIANT_BOOL* pbSuccess);
```

Visual Basic Syntax

```
[ pbSuccess  = ] Profile.PrintDoc
```

Parameters

pbSuccess	[out, retval] Returns a Boolean value that specifies whether or not the operation succeeded. True: The document was successfully printed. False: The operation was cancelled or it was impossible to print the document.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Prints a document.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

IProfile::ReleaseDoc

The method unlocks a document.

C++ Syntax

```
HRESULT ReleaseDoc();
```

Visual Basic Syntax

```
Profile.ReleaseDoc
```

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

A document, which has been locked should be released in order to remove document's locking.

Example

The following example changes the name of the profile with number 20.

```
Dim App As New Application
Dim Prof As Profile
Set Prof = App.PrimaryLibrary.GetProfile(20)
If Not Prof Is Nothing Then
If Prof.SelectForEdit(OF_NORMAL + SFE_LOCK_PROFILE) = _
SFER_OK Then
Prof.Columns("$DOCNAME").Value = "New Name"
Prof.Save OF_NORMAL
Prof.ReleaseDocIf Prof.Name <> "New Name" Then
MsgBox "Document name hasn't changed"
End If
End If
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

IProfile::Save

The method saves a new profile or changes to an existing profile.

C++ Syntax

```
HRESULT Save(  short sFlags,  VARIANT_BOOL* pbSuccess);
```

Visual Basic Syntax

```
[ pbSuccess  = ] Profile.Save (sFlags)
```

Parameters

sFlags	[in] Required. An integer value that specifies how the method should be executed. The parameter can be a combination of the MethodsFlags values.
pbSuccess	[out, retval] Returns a Boolean value that specifies whether or not the operation succeeded. True: Profile data was successfully saved. False: The operation was cancelled or it was impossible to save profile data.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Whenever a new profile is created the information assigned to the profile column values should be saved using the Save method. If changes are made to an existing profile without displaying the profile form then the method should also be used to save those changes to the database. If theShowProfile method is used, then Save does not need to be called.

Example

The following example logs into the DM Server and creates a new profile paper document on the primary library.

```
Dim App As New Application
Dim Prof As Profile

' Log into the DM Server
```

```
If App.DST = "" Then Exit Sub


' Create a new blank profile on the primary library
Set Prof = App.PrimaryLibrary.CreateProfile("DEF_PROF")
' Display the profile form
If Prof.ShowProfile(SP_EDIT) = FR_DATACHANGED Then
' Set the document type to paper document and save the profile
Prof.Columns("$STORAGETYPE").Value = "P"
If Prof.Save(OF_NORMAL) Then
MsgBox "Profile was successfully saved."
Else
MsgBox "Profile wasn't saved."
End If
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

"IProfile::ShowProfile" on page 458

"MethodsFlags Enumeration" on page 689

IProfile::SelectForEdit

The method selects a document for editing.

C++ Syntax

```
HRESULT SelectForEdit(  short sFlags,  short* psResult);
```

Visual Basic Syntax

```
[ psResult  = ] Profile.SelectForEdit (sFlags)
```

Parameters

sFlags	[in] Required. An integer value that specifies how the method should be executed. The parameter can be a combination of the MethodsFlags and SelectForEditFlags values.
psResult	[out, retval] Returns an integer that specifies whether or not the operation succeeded. For possible values see SelectForEditReturnFlags Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The SelectForEdit method must be used to select a document before changing its profile, or changes to the document's trustee list can be saved. The current version of the document is selected by default. When the document is no longer needed use theReleaseDoc method to unlock the document.

There is no need to call up the method when opening a document because it will be locked automatically.

Example

The following example changes the name of the profile with number 1731.

```
Dim App As New Application
Dim Prof As Profile
Set Prof = App.PrimaryLibrary.GetProfile(1731)
If Not Prof Is Nothing Then
If Prof.SelectForEdit(OF_NORMAL + SFE_LOCK_PROFILE) = _
SFER_OK Then
Prof.Columns("$DOCNAME").Value = "New Name"
Prof.Save OF_NORMAL
Prof.ReleaseDoc
If Prof.Name <> "New Name" Then
```

```
MsgBox "Document name hasn't changed"
End If
End If
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

"IProfile::ReleaseDoc" on page 450

"MethodsFlags Enumeration" on page 689

"SelectForEditFlags Enumeration" on page 700

"SelectForEditReturnFlags Enumeration" on page 701

IProfile::SetDocument

The method initializes a Profile object to a particular document.

C++ Syntax

```
HRESULT SetDocument(  VARIANT vLibrary,  VARIANT vDoc);
```

Visual Basic Syntax

```
Profile.SetDocument (vLibrary, vDoc)
```

Parameters

vLibrary	[in] Required. An expression that specifies a library. If a string expression, vLibrary should specify the library name. If an object, vLibrary should specify a valid Library object.
vDoc	[in] Required. An expression that specifies a document profile on vLibrary. If a numeric expression, vDocmust specifies the number of a document. If a string expression, vDoc should specify the document name.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

If the profile was created with the CreateObject function or the Dim statement with the New keyword, the method is to be called up before any other method or property.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

"ILibrary" on page 360

IProfile::ShowAccessForm

The method displays the document's access control form.

C++ Syntax

```
HRESULT ShowAccessForm(  VARIANT_BOOL bAllowEdits,  short*
psResult);
```

Visual Basic Syntax

```
[ psResult  = ] Profile.ShowAccessForm (bAllowEdits)
```

Parameters

bAllowEdits	[in] Required. A Boolean value that specifies whether or not the document's trustee list can be changed. True: The document's trustee list can be changed. False: The document's trustee list cannot be changed.
psResult	[out, retval] Returns an integer that specifies user's action on the form. For possible values see FormsResults Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The document's access control form can be customized and should support the IProfileAccessForm interface. To display the form, the method calls up theShowProfileAccessForm method onIProfileAccessForm.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

"IProfileAccessForm" on page 894

"IProfileAccessForm::ShowProfileAccessForm" on page 895

IProfile::ShowHistoryForm

The method displays the document's history form that presents data from the document's activity log.

C++ Syntax

```
HRESULT ShowHistoryForm();
```

Visual Basic Syntax

```
Profile.ShowHistoryForm
```

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The document's history form can be customized and should support the IHistoryForm interface. To display the form, the method calls up theShowHistoryForm method onIHistoryForm.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

"IHistoryForm" on page 883

"IHistoryForm::ShowHistoryForm" on page 884

IProfile::ShowProfile

The method displays the document's profile form.

C++ Syntax

```
HRESULT ShowProfile(  short sFlags,  short* pResult);
```

Visual Basic Syntax

```
[ pResult  = ] Profile.ShowProfile (sFlags)
```

Parameters

sFlags	[in] Required. An integer value that specifies how the method should be executed. The parameter can be a combination of the ShowProfileFlags values.
pResult	[out, retval] Returns an integer that specifies user's action on the form. For possible values see FormsResults Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

If changes are made to the profile when it is displayed and the user clicks OK, the program does not need to call theSave method because the changes are saved automatically.

To display a document profile the user should have view profile rights. The user cannot change any data on a profile form if he/she does not have edit profile rights.

The profile form can be customized and should support the IProfileForm interface. To display the form, the method calls theShowProfileForm method on IProfileForm.

Example

The following example logs into the DM Server and creates a new profile paper document on the primary library.

```
Dim App As New Application
Dim Prof As Profile

' Log into the DM Server
If App.DST = "" Then Exit Sub

' Create a new blank profile on the primary library
Set Prof = App.PrimaryLibrary.CreateProfile("DEF_PROF")
' Display the profile form
```

```
If Prof.ShowProfile(SP_EDIT) = FR_DATACHANGED Then
' Set the document type to paper document and save the profile
Prof.Columns("$STORAGETYPE").Value = "P"
Prof.Save OF_NORMAL
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

"IProfile::Save" on page 451

"IProfileForm" on page 904

"IProfileForm::ShowProfileForm" on page 905

IProfile::ShowProjectsMembershipForm

The method displays the Where Used form - the list of folders and complex document a document profile contains.

C++ Syntax

```
HRESULT ShowProjectsMembershipForm();
```

Visual Basic Syntax

```
Profile.ShowProjectsMembershipForm
```

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The presented list of folders and complex documents includes items on remote libraries. The form can be customized and should support theIProjectMembershipForm and the IProjectMembershipForm2 interfaces. To display the form, the method calls up theShowProjectMembershipForm method onIProjectMembershipForm.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

"IProjectMembershipForm" on page 922

"IProjectMembershipForm2" on page 924

"IProjectMembershipForm::ShowProjectMembershipForm" on page 923

IProfile::ShowSaveAsOptionsForm

The method displays the custom Save As options form and allows the user to select an option.

C++ Syntax

```
HRESULT ShowSaveAsOptionsForm(  IVersion* pVersion,  IVersion**
ppOut);
```

Visual Basic Syntax

```
[ Set ppOut  = ] Profile.ShowSaveAsOptionsForm (pVersion)
```

Parameters

pVersion	[in] Required. A Version object that specifies a version of the document.
ppOut	[out, retval] The method returns a Version object that specifies the saved version. It may be the same version as pVersion, a new version/ subversion of the same document or a new document. If the operation was canceled or an error occurred, the return value is NULL.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

This is the easy way to allow the user to select Save As options. The options given to the user are: save as new document, save as new version, save as new subversion, and replace original. If the document is not selected for editing the method will fail.

The form can be customized and should support the ISaveAsOptionsForm interface. To display the form, the method calls up theShowSaveAsOptionsForm method onISaveAsOptionsForm.

Example

The following example displays the Save As options form for the document with number 20 on the primary library.

```
Dim App As New Application
Dim Prof As Profile, Ver As Version
Set Prof = App.PrimaryLibrary.GetProfile(20)
If Not Prof Is Nothing Then
Prof.SelectForEdit OF_NORMAL + SFE_LOCK_DOCUMENT
Set Ver = Prof.ShowSaveAsOptionsForm(Prof.CurrentVersion)
Prof.ReleaseDoc
If Not Ver Is Nothing Then
MsgBox "ShowSaveAsOptionsForm succeeded."
End If
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

"IVersion" on page 625

"ISaveAsOptionsForm" on page 931

"ISaveAsOptionsForm::ShowSaveAsOptionsForm" on page 931

IProfile::ShowVersionChooserForm

The method displays the custom form for selecting a version/ attachment.

C++ Syntax

```
HRESULT ShowVersionChooserForm(  short sFlags,  BSTR bstrCaption,
IVersion** ppVersion);
```

Visual Basic Syntax

```
[ Set ppVersion  = ] Profile.ShowVersionChooserForm (sFlags,
bstrCaption)
```

Parameters

sFlags	[in] Required. An integer value that specifies the form appearance. The parameter can be a combination of the VersionChooserFormFlags values.
bstrCaption	[in, optional] Required. A string that specifies the form caption.
ppVersion	[out, retval] Returns a Version object that specifies the version selected by the user. If the user selected no version, the return value is NULL.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The form can be customized and should support the IVersionChooserForm interface. To display the form, the method calls up theShowVersionChooserForm method on IVersionChooserForm.

Example

The following example uploads a local file to the version selected by the user on the document with number 20 from the primary library.

```
Dim App As New Application
Dim Prof As Profile, Ver As Version

Set Prof = App.PrimaryLibrary.GetProfile(20)
If Not Prof Is Nothing Then
If Prof.SelectForEdit(OF_NORMAL + SFE_LOCK_DOCUMENT) = _
SFER_OK Then
' Select a version of the document
Set Ver = Prof.ShowVersionChooserForm(VCF_Default, "Versions")
If Not Ver Is Nothing Then
' Set the version to a local MS Word file
Ver.LockVersionForEdit OF_NORMAL
FileCopy "C:\MyDoc.doc", Ver.FilePath
Ver.UploadFile Nothing
```

```
Ver.UnlockVersion OF_NORMAL
End If
Prof.ReleaseDoc
End If
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

"IVersion" on page 625

"IVersionChooserForm" on page 952

"IVersionChooserForm::ShowVersionChooserForm" on page 953

"VersionChooserFormFlags Enumeration" on page 704

IProfile::Trustees

The property returns a collection of users and groups who have access to a document.

C++ Syntax

```
HRESULT Trustees(  ITrusteeCollection** ppTrustees);
```

Visual Basic Syntax

```
[ Set ppTrustees = ] Profile.Trustees
```

Parameters

ppTrustees	[out, retval] Returns a TrusteeCollection object that represents a collection of document's trustees or NULL if the document is not secured.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns a collection of document trustees.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

"ITrusteeCollection" on page 609

IProfile::UnlockProfile

The method unlocks a document profile.

C++ Syntax

```
HRESULT UnlockProfile(  short sFlags,  long lEditTime,  BSTR
bstrCopyFromPath,  short sAction,  IVersion** ppVersion);
```

Visual Basic Syntax

```
[ Set ppVersion  = ] Profile.UnlockProfile (sFlags, lEditTime,
bstrCopyFromPath, sAction)
```

Parameters

sFlags	[in] Required. An integer value that specifies how the method should be executed. The parameter can be a combination of the MethodsFlags values.
lEditTime	[in] Required. A long that specifies how long the profile has been edited in minutes.
bstrCopyFromPath	[in] Required. A string that specifies the full path (without the file name) to the document version.
sAction	[in] Required. An integer value that specifies what action should be performed. The parameter can be one of the CheckInActions values.
ppVersion	[out,retval] Returns a Version object that specifies the current version of the last unlocked document. If the operation was canceled, an error occurred, or the UF_UNLOCK flag is specified the return value is NULL.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed. Unexpected error occurs.

Remarks

If the show form bit is on or the data are insufficient, the check-in dialog pops up. The dialog can be customized and its form should support theIGroupCheckinForm interface. To display the form, the method calls up theShowGroupCheckinForm method onIGroupCheckinForm.

If necessary (if sAction is not UF_UNLOCK), the file is uploaded onto the server.

Example

The following example locks and then unlocks the document with number 20 on the primary library.

```
Dim App As New Application
Dim Prof As Profile, Ver As Version
Set Prof = App.PrimaryLibrary.GetProfile(20)
If Not Prof Is Nothing Then
If Prof.LockProfile(OF_SHOWFORM, DateAdd("d", 10, Now), "") Then
Dim LockTime As Date
LockTime = Now

' Some operations with the Prof
...

Set Ver = Prof.UnlockProfile(OF_SHOWFORM, _
DateDiff("n", LockTime, Now), "C:\TEMP", UF_ NEWVERSION)
If Not Ver Is Nothing Then
MsgBox "New version was successfully created."
Else
MsgBox "Operation failed."
End If
End If
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

“IProfile” on page 419

“IVersion” on page 625

“IGroupCheckinForm” on page 880

“IGroupCheckoutForm::ShowGroupCheckoutForm” on page 882

“MethodsFlags Enumeration” on page 689

“CheckInActions Enumeration” on page 681

IProfile::Versions

The property returns a collection of all document versions, subversions, and attachments.

C++ Syntax

```
HRESULT Versions(  IVersions** ppVersions);
```

Visual Basic Syntax

```
[ Set ppVersions = ] Profile.Versions
```

Parameters

ppVersions	[out, retval] Returns a Versions object that represents the requested collection.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The property returns a collection of all document versions, subversions, and attachments.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

"IVersions" on page 673

IProfile::WorkingVersion

The property returns the working version of a document.

C++ Syntax

```
HRESULT WorkingVersion(  IVersion** ppVersion);
```

Visual Basic Syntax

```
[ Set ppVersion = ] Profile.WorkingVersion
```

Parameters

ppVersion	[out, retval] Returns a Version object that specifies the document's working version.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The default version of a document is its current version.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

"IVersion" on page 625

IProfile2

Interface IProfile2 extends functionality of Profile object.

Methods

IProfile2 Methods	Description
IProfile2::Publish	Publish document.

Properties

IProfile2 Properties	Description
IProfile2::RelatedItems	Returns related items collection.
IProfile2::WhereUsed	Returns collection where current document is used.
IProfile2::Contents	Returns collection of items that are contained in current document.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import "fsplugin.dll"

IProfile2::Contents

The property returns a collection of items that are used in current Profile. Content is returned for folders. For complex documents all OLE links (documents that are linked into current document) are returned.

C++ Syntax

```
HRESULT Contents(  IDOComponents** ppComponents);
```

Visual Basic Syntax

```
[ Set ppComponents = ] IProfile2.Contents
```

Parameters

ppComponents

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns collection of items that are contained in current document.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IProfile2" on page 470

IProfile2::Publish

The method publish document. Version chooser dialog will be displayed where user is able to select version to publish.

C++ Syntax

```
HRESULT Publish(  IVersion** ppPublishedVersion);
```

Visual Basic Syntax

```
[ Set ppPublishedVersion = ] IProfile2.Publish
```

Parameters

ppPublishedVersion	[out, retval] Returns a published Version object.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Publish document.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IProfile2" on page 470

IProfile2::RelatedItems

The property returns a collection of related items.

C++ Syntax

```
HRESULT RelatedItems(  IDocProfiles** ppDocuments);
```

Visual Basic Syntax

```
[ Set ppDocuments = ] IProfile2.RelatedItems
```

Parameters

ppDocuments	[out, retval] Returns a DocProfiles collection of related items.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns related items collection.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IProfile2" on page 470

IProfile2::WhereUsed

The property returns a collection of where used items (parent folders and complex documents).

C++ Syntax

```
HRESULT WhereUsed(  IDOComponents** ppComponents);
```

Visual Basic Syntax

```
[ Set ppComponents = ] IProfile2.WhereUsed
```

Parameters

ppComponents	[out, retval] Returns a DOComponents collection of items where the current Profile object is used.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns collection where current document is used.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IProfile2" on page 470

IProfileTrustee

The ProfileTrustee object represents a user or group who has access to a profiled item (document, paper profile document or folder). The ProfileTrustee object may be retrieved through the Trustees collection on the document.

Object's properties can be modified only if the user is allowed to control the access rights to the profiled item. When the user who does not have enough rights attempts to make changes in the access rights, an error will occur.

Properties

IProfileTrustee Properties	Description
CanControlAccess	Returns/sets whether a trustee has control access rights.
CanCopy	Returns/sets whether a trustee has copy rights.
CanDelete	Returns/sets whether a trustee has delete rights.
CanEditDocument	Returns/sets whether a trustee has edit content rights.
CanEditProfile	Returns/sets whether a trustee has edit profile rights.
CanRetrieve	Returns/sets whether a trustee has retrieve rights.
CanViewDocument	Returns/sets whether a trustee has view content rights.
CanViewProfile	Returns/sets whether a trustee has view profile rights.
IsGroup	Returns whether an object is a group trustee.
Name	Returns the name of a trustee.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "fsplugin.dll"

See Also

"ITrusteeCollection" on page 609

IProfileTrustee::CanControlAccess

The property returns or sets whether or not a trustee can make modifications to the trustee list for a specific profiled item.

C++ Syntax

```
HRESULT CanControlAccess(  VARIANT_BOOL* pbControl);
HRESULT CanControlAccess(  VARIANT_BOOL bControl);
```

Visual Basic Syntax

```
[ pbControl = ] ProfileTrustee.CanControlAccess
ProfileTrustee.CanControlAccess [ = bControl ]
```

Parameters

pbControl	[out, retval] A Boolean value that specifies whether or not the trustee has control access rights.
bControl	[in] A Boolean value that specifies whether or not the trustee has control access rights. True: The trustee has control access rights. False: The trustee has not control access rights.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

The property checks whether or not this trustee has control access rights to the profiled item. It does not verify whether or not this trustee has rights through another profiled item trustee.

Example

The following example checks whether or not the user "Monica" has control access rights to the document with number 20 on the primary library.

```
Dim App As New Application
Dim Prof As Profile, ProfTrustee As ProfileTrustee
Set Prof = App.PrimaryLibrary.GetProfile(20)
' Check whether the document is secured
If Not Prof Is Nothing And _
Prof.Columns("$DEFAULT_RIGHTS").Value <> "0" Then
Set ProfTrustee = Prof.Trustees.GetEffectiveRights("Monica")
If ProfTrustee.CanControlAccess Then
MsgBox "Monica has control access rights."
```

```
Else
MsgBox "Monica doesn't have control access rights."
End If
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfileTrustee" on page 474

IProfileTrustee::CanCopy

The property returns or sets whether or not a trustee can copy a specific profiled item.

C++ Syntax

```
HRESULT CanCopy(  VARIANT_BOOL* pbCopy);
HRESULT CanCopy(  VARIANT_BOOL bCopy);
```

Visual Basic Syntax

```
[ pbCopy = ] ProfileTrustee.CanCopy
ProfileTrustee.CanCopy [ = bCopy ]
```

Parameters

pbCopy	[out, retval] A Boolean value that specifies whether or not the trustee has copy rights.
bCopy	[in] A Boolean value that specifies whether or not the trustee has copy rights. True: The trustee has copy rights. False: The trustee has not copy rights.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

The property checks whether or not this trustee has copy rights to the profiled item. It does not verify whether or not this trustee has rights through another profiled item trustee.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfileTrustee" on page 474

IProfileTrustee::CanDelete

The property returns or sets whether or not a trustee can delete a specific profiled item.

C++ Syntax

```
HRESULT CanDelete(  VARIANT_BOOL* pbDelete);
HRESULT CanDelete(  VARIANT_BOOL bDelete);
```

Visual Basic Syntax

```
[ pbDelete = ] ProfileTrustee.CanDelete
```

```
ProfileTrustee.CanDelete [ = bDelete ]
```

Parameters

pbDelete	[out, retval] A Boolean value that specifies whether or not the trustee has delete rights.
bDelete	[in] A Boolean value that specifies whether or not the trustee has delete rights. True: The trustee has delete rights. False: The trustee has not delete rights.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

The property checks whether or not this trustee has delete rights to the profiled item. It does not verify whether or not this trustee has rights through another profiled item trustee.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfileTrustee" on page 474

IProfileTrustee::CanEditDocument

The property returns or sets whether or not a trustee can make changes to a specific profiled item.

C++ Syntax

```
HRESULT CanEditDocument(  VARIANT_BOOL* pbEdit);
HRESULT CanEditDocument(  VARIANT_BOOL bEdit);
```

Visual Basic Syntax

```
[ pbEdit = ] ProfileTrustee.CanEditDocument
ProfileTrustee.CanEditDocument [ = bEdit ]
```

Parameters

pbEdit	[out, retval] A Boolean value that specifies whether or not the trustee has edit rights.
bEdit	[in] A Boolean value that specifies whether or not the trustee has edit rights. True: The trustee has edit rights. False: The trustee has not edit rights.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

The property checks whether or not this trustee has edit rights to the profiled item. It does not verify whether or not this trustee has rights through another profiled item trustee.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfileTrustee" on page 474

IProfileTrustee::CanEditProfile

The property returns or sets whether or not a trustee can make changes to the profile of a specific profiled item.

C++ Syntax

```
HRESULT CanEditProfile(  VARIANT_BOOL* pbEdit);
```

```
HRESULT CanEditProfile(  VARIANT_BOOL bEdit);
```

Visual Basic Syntax

```
[ pbEdit = ] ProfileTrustee.CanEditProfile
ProfileTrustee.CanEditProfile [ = bEdit ]
```

Parameters

pbEdit	[out, retval] A Boolean value that specifies whether or not the trustee has edit profile rights.
bEdit	[in] A Boolean value that specifies whether or not the trustee has edit profile rights. True: The trustee has edit profile rights. False: The trustee has not edit profile rights.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

The property checks whether or not this trustee has edit profile rights to the profiled item. It does not verify whether or not this trustee has rights through another profiled item trustee.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfileTrustee" on page 474

IProfileTrustee::CanRetrieve

The property returns or sets whether or not a trustee can retrieve the document's contents.

C++ Syntax

```
HRESULT CanRetrieve(  VARIANT_BOOL* pbRetrieve);

HRESULT CanRetrieve(  VARIANT_BOOL bRetrieve);
```

Visual Basic Syntax

```
[ pbRetrieve = ] ProfileTrustee.CanRetrieve

ProfileTrustee.CanRetrieve [ = bRetrieve ]
```

Parameters

pbRetrieve	[out, retval] A Boolean value that specifies whether or not the trustee has retrieve rights.
bRetrieve	[in] A Boolean value that specifies whether or not the trustee has retrieve rights. True: The trustee has retrieve rights. False: The trustee has not retrieve rights.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

The property checks whether or not this trustee has retrieve rights to the document. It does not verify whether or not this trustee has rights through another profiled item trustee.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfileTrustee" on page 474

IProfileTrustee::CanViewDocument

The property returns or sets whether or not a trustee can view a specific profiled item.

C++ Syntax

```
HRESULT CanViewDocument(  VARIANT_BOOL* pbView);
HRESULT CanViewDocument(  VARIANT_BOOL bView);
```

Visual Basic Syntax

```
[ pbView = ] ProfileTrustee.CanViewDocument
ProfileTrustee.CanViewDocument [ = bView ]
```

Parameters

pbView	[out, retval] A Boolean value that specifies whether or not the trustee has view rights.
bView	[in] A Boolean value that specifies whether or not the trustee has view rights. True: The trustee has view rights. False: The trustee has not view rights.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

The property checks whether or not this trustee has view rights to the profiled item. It does not verify whether or not this trustee has rights through another profiled item trustee.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfileTrustee" on page 474

IProfileTrustee::CanViewProfile

The property returns or sets whether or not a trustee is able to see the profile of a specific profiled item.

C++ Syntax

```
HRESULT CanViewProfile(  VARIANT_BOOL* pbView);
HRESULT CanViewProfile(  VARIANT_BOOL bView);
```

Visual Basic Syntax

```
[ pbView = ] ProfileTrustee.CanViewProfile
ProfileTrustee.CanViewProfile [ = bView ]
```

Parameters

pbView	[out, retval] A Boolean value that specifies whether or not the trustee has view profile rights.
bView	[in] A Boolean value that specifies whether or not the trustee has view profile rights. True: The trustee has view profile rights. False: The trustee has not view profile rights.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

The property checks whether or not this trustee has view profile rights to the profiled item. It does not verify whether or not this trustee has rights through another profiled item trustee.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfileTrustee" on page 474

IProfileTrustee::IsGroup

The property returns whether or not a trustee is a group.

C++ Syntax

```
HRESULT IsGroup(  VARIANT_BOOL* pbGroup);
```

Visual Basic Syntax

```
[ pbGroup = ] ProfileTrustee.IsGroup
```

Parameters

pbGroup	[out, retval] Returns a Boolean value that specifies whether or not the trustee is a group. True: The trustee is a group. False: The trustee is not a group.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

The IsGroup property is useful in determining which Trustees methods to use with a Trustee.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfileTrustee" on page 474

"ITrusteeCollection" on page 609

IProfileTrustee::Name

The property returns the DM trustee's username or group name.

C++ Syntax

```
HRESULT Name(  BSTR* pbstrName);
```

Visual Basic Syntax

```
[ pbstrName = ] ProfileTrustee.Name
```

Parameters

pbstrName	[out, retval] Returns a string that specifies the trustee's name.

Return Values

S_OK	The operation succeeded.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the name of a trustee.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfileTrustee" on page 474

IProfileTrustee2

Interface IProfileTrustees2 extends functionality of ProfileTrustee object.

Properties

IProfileTrustee2 Properties	Description
AccessControlAccess	Returns/sets trustee's control access access.
AccessCopy	Returns/sets trustee's copy access.
AccessDelete	Returns/sets trustee's delete access.
AccessEditDocument	Returns/sets trustee's edit content access.
AccessEditProfile	Returns/sets trustee's edit profile access.
AccessRetrieve	Returns/sets trustee's retrieve access.
AccessViewDocument	
AccessViewProfile	Returns/sets trustee's view profile access.
AccessViewPublishedOnly	Returns/sets trustee's view profile access.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import "fsplugin.dll"

IProfileTrustee2::AccessControlAccess

The property returns or sets whether or not a trustee can make modifications to the trustee list for a specific profiled item.

C++ Syntax

```
HRESULT AccessControlAccess(  short* psControl);

HRESULT AccessControlAccess(  short sControl);
```

Visual Basic Syntax

```
[ Set psControl = ] IProfileTrustee2.AccessControlAccess

IProfileTrustee2.AccessControlAccess [ = sControl ]
```

Parameters

psControl	[out, retval] An AccessType value that specifies whether or not the trustee has control access rights.
sControl	[in] An AccessType value that specifies whether or not the trustee has control access rights.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

The property checks whether or not this trustee has control access rights to the profiled item. It doesn't verify whether or not this trustee has rights through another profiled item trustee.

Example

The following example checks whether or not the user "Monica" has control access rights to the document with number 20 on the primary library.

```
Dim App As New Application
 Dim Prof As Profile, ProfTrustee As ProfileTrustee
 Set Prof = App.PrimaryLibrary.GetProfile(20)
 ' Check whether the document is secured
 If Not Prof Is Nothing And _
 Prof.Columns("$DEFAULT_RIGHTS").Value <> "0" Then
 Set ProfTrustee = Prof.Trustees("Monica")
 If Not (ProfTrustee Is Nothing) Then
 If ProfTrustee.AccessControlAccess = AT_ALLOW Then
 MsgBox "Monica has control access rights."
 ElseIf ProfTrustee.AccessControlAccess = AT_NONE Then
 MsgBox "Monica doesn't have control access rights."
 ElseIf ProfTrustee.AccessControlAccess = AT_DENY Then
 MsgBox "Control access is denied for Monica "
 End If
 End If
```

```
End If
```

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IProfileTrustee2" on page 486

IProfileTrustee2::AccessCopy

The property returns or sets whether or not a trustee can copy a specific profiled item.

C++ Syntax

```
HRESULT AccessCopy(  short* psCopy);
HRESULT AccessCopy(  short sCopy);
```

Visual Basic Syntax

```
[ Set psCopy = ] IProfileTrustee2.AccessCopy
IProfileTrustee2.AccessCopy [ = sCopy ]
```

Parameters

psCopy	[out, retval] An AccessType value that specifies whether or not the trustee has copy rights.
sCopy	[in] An AccessType value that specifies whether or not the trustee has copy rights.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

The property checks whether or not this trustee has copy rights to the profiled item. It doesn't verify whether or not this trustee has rights through another profiled item trustee.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IProfileTrustee2" on page 486

IProfileTrustee2::AccessDelete

The property returns or sets whether or not a trustee can delete a specific profiled item.

C++ Syntax

```
HRESULT AccessDelete(  short* psDelete);
HRESULT AccessDelete(short sDelete);
```

Visual Basic Syntax

```
[ Set psDelete = ] IProfileTrustee2.AccessDelete
IProfileTrustee2.AccessDelete [ = sDelete ]
```

Parameters

psDelete	[out, retval] An AccessType value that specifies whether or not the trustee has delete rights.
sDelete	[in] An AccessType value that specifies whether or not the trustee has delete rights.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

The property checks whether or not this trustee has delete rights to the profiled item. It doesn't verify whether or not this trustee has rights through another profiled item trustee.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IProfileTrustee2" on page 486

IProfileTrustee2::AccessEditDocument

The property returns or sets whether or not a trustee can make changes to a specific profiled item.

C++ Syntax

```
HRESULT AccessEditDocument(  short* psEdit);
HRESULT AccessEditDocument(  short sEdit);
```

Visual Basic Syntax

```
[ Set psEdit = ] IProfileTrustee2.AccessEditDocument
IProfileTrustee2.AccessEditDocument [ = sEdit ]
```

Parameters

psEdit	[out, retval] An AccessType value that specifies whether or not the trustee has edit rights.
sEdit	[in] An AccessType value that specifies whether or not the trustee has edit rights.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

The property checks whether or not this trustee has edit rights to the profiled item. It doesn't verify whether or not this trustee has rights through another profiled item trustee.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IProfileTrustee2" on page 486

IProfileTrustee2::AccessEditProfile

The property returns or sets whether or not a trustee can make changes to the profile of a specific profiled item.

C++ Syntax

```
HRESULT AccessEditProfile(  short* psEdit);
HRESULT AccessEditProfile(  short sEdit);
```

Visual Basic Syntax

```
[ Set psEdit = ] IProfileTrustee2.AccessEditProfile
IProfileTrustee2.AccessEditProfile [ = sEdit ]
```

Parameters

psEdit	[out, retval] An AccessType value that specifies whether or not the trustee has edit profile rights.
sEdit	[in] An AccessType value that specifies whether or not the trustee has edit profile rights.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

The property checks whether or not this trustee has edit profile rights to the profiled item. It doesn't verify whether or not this trustee has rights through another profiled item trustee.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IProfileTrustee2" on page 486

IProfileTrustee2::AccessRetrieve

The property returns or sets whether or not a trustee can retrieve the document's contents.

C++ Syntax

```
HRESULT AccessRetrieve(  short* psRetrieve);HRESULT
AccessRetrieve(  short sRetrieve);
```

Visual Basic Syntax

```
[ Set psRetrieve = ] IProfileTrustee2.AccessRetrieve
IProfileTrustee2.AccessRetrieve [ = sRetrieve ]
```

Parameters

psRetrieve	[out, retval] An AccessType value that specifies whether or not the trustee has retrieve rights.
sRetrieve	[in] An AccessType value that specifies whether or not the trustee has retrieve rights.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

The property checks whether or not this trustee has retrieve rights to the document. It doesn't verify whether or not this trustee has rights through another profiled item trustee.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IProfileTrustee2" on page 486

IProfileTrustee2::AccessViewDocument

The property returns or sets whether or not a trustee can view a specific profiled item.

C++ Syntax

```
HRESULT AccessViewDocument(  short* psView);
HRESULT AccessViewDocument(  short sView);
```

Visual Basic Syntax

```
[ Set psView = ] IProfileTrustee2.AccessViewDocument
IProfileTrustee2.AccessViewDocument [ = sView ]
```

Parameters

psView	[out, retval] An AccessType value that specifies whether or not the trustee has view rights.
sView	[in] An AccessType value that specifies whether or not the trustee has view rights.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

The property checks whether or not this trustee has view rights to the profiled item. It doesn't verify whether or not this trustee has rights through another profiled item trustee.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IProfileTrustee2" on page 486

IProfileTrustee2::AccessViewProfile

The property returns or sets whether or not a trustee is able to see the profile of a specific profiled item.

C++ Syntax

```
HRESULT AccessViewProfile(  short* psView);

HRESULT AccessViewProfile(  short sView);
```

Visual Basic Syntax

```
[ Set psView = ] IProfileTrustee2.AccessViewProfile

IProfileTrustee2.AccessViewProfile [ = sView ]
```

Parameters

psView	[out, retval] An AccessType value that specifies whether or not the trustee has view profile rights.
sView	[in] An AccessType value that specifies whether or not the trustee has view profile rights.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

The property checks whether or not this trustee has view profile rights to the profiled item. It doesn't verify whether or not this trustee has rights through another profiled item trustee.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IProfileTrustee2" on page 486

IProfileTrustee2::AccessViewPublishedOnly

The property returns or sets whether or not a trustee is able to see view only published versions of a specific document.

C++ Syntax

```
HRESULT AccessViewPublishedOnly(  short* psView);
HRESULT AccessViewPublishedOnly(  short sView);
```

Visual Basic Syntax

```
[ Set psView = ] IProfileTrustee2.AccessViewPublishedOnly
IProfileTrustee2.AccessViewPublishedOnly [ = sView ]
```

Parameters

psView	[out, retval] An AccessType value that specifies whether or not the trustee has view published versions only rights.
sView	[in] An AccessType value that specifies whether or not the trustee has view published versions only rights.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

The property checks whether or not this trustee has view published versions only to the document. It doesn't verify whether or not this trustee has rights through another profiled item trustee. If this property is set to AT_ALLOW only published versions are available.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IProfileTrustee2" on page 486

IProject

The Project object represents a DM profiled folder. Folders allow the user to group documents and other folders used for the same effort. A single document may be included in multiple projects. A Folder may even be included in a larger folder, becoming a sub-folder to the larger one.

The Project object inherits its definition from the IOMErrInfo interface so its implementation also includes all the properties of IOMErrInfo in addition to the ones described below. The Project object supports IProfile/IProfile2 interfaces.

When to Use

A Project object may be created with the CreateObject function or the Dim statement with the New keyword. To initialize the object the SetProject method is to be called up before any other method or property. The following example shows the Visual Basic syntax for creating a DOCSObjects.Project object:

```
Sub Form_Load()
 ...
 Dim Prj As Object
 ' Create a DOCSObjects.Project object
 Set Prj = CreateObject("DOCSObjects.Project")
 Prj.SetProject "Test", "My Project"
 ...
 End Sub
```

Methods

IProject Methods	Description
IProject::SetProject	Initializes a folder.
IProject::Save	Saves a folder.
IProject::ShowAccessForm	Shows a folder access form.
IProject::DisplayModal	Displays a folder form.
IProject::AddDocument	Adds a document to a folder.
IProject::RemoveDocument	Removes a document from a folder.
IProject::AddSubProject	Adds a sub-folder to a folder.
IProject::RemoveSubProject	Removes a sub-folder from a folder.
IProject::Delete	Deletes a folder.

Properties

IProject Properties	Description
IProject::Name	Returns/sets the name of a folder.
IProject::ProjectID	Returns the folder identifier.
IProject::Iconized	Returns/sets whether a folder is iconized on to desktop.
IProject::Library	Returns the folder library.
IProject::SubDocuments	Returns a collection of all documents included in a folder.
IProject::SubProjects	Returns a collection of all sub-folders included in a folder.
IProject::SelectedDocuments	Returns/sets selected documents.
IProject::SelectedSubProjects	Returns/sets selected sub-folders.
IProject::Caption	Returns/sets the folder caption.
IProject::Trustees	Returns a collection of folder trustees.
IProject::Icon	Returns the folder icon.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "fsplugin.dll"

See Also

"IOMErrInfo" on page 412

IProject::AddDocument

The method adds a document to a folder.

C++ Syntax

```
HRESULT AddDocument(  VARIANT vDocument,  VARIANT_BOOL*
pbSuccess);
```

Visual Basic Syntax

```
[ pbSuccess  = ] Project.AddDocument (vDocument)
```

Parameters

vDocument	[in] Required. An expression that specifies a document profile. If an object, vDocument should specify a valid Profile object. If a numeric expression, vDocument should specify the number of a document on the current library. If a string expression, vDocument should specify the name of a document on the current library.
pbSuccess	[out, retval] Returns a Boolean value that specifies whether or not a document was successfully added to the folder. True: A document was successfully added to the folder. False: A document was not added to the folder.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Adds a document to a folder.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProject" on page 497

"IProfile" on page 419

IProject::AddSubProject

The method adds a sub-folder to the folder.

C++ Syntax

```
HRESULT AddSubProject(  VARIANT vSubproject,  VARIANT_BOOL*
pbSuccess);
```

Visual Basic Syntax

```
[ pbSuccess  = ] Project.AddSubProject (vSubproject)
```

Parameters

vSubproject	[in] Required. An expression that specifies a folder to be added as a sub-folder to the current folder. If an object, vSubproject should specify a valid Project object. If a numeric expression, vSubproject should specify the identifier of a folder on the current library. If a string expression, vSubproject should specify the name of a folder on the current library.
pbSuccess	[out, retval] Returns a Boolean value that specifies whether or not a sub-folder was successfully added to the folder. True: A sub-folder was successfully added to the folder. False: A sub-folder was not added to the folder.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The folder to be added must already exist.

Example

The following example adds all the folders selected by the user to another folder.

```
Dim App As New Application
Dim Proj, Subproj As Project
Dim Projs As Projects

' Select a folder sub-folders will be added to
Set Projs = App.ShowProjectChooserForm(PCF_ALL)
If Not Projs Is Nothing Then
Set Proj = Projs(1)
```

```
' Select sub-folder that will be added to Proj
Set Projs = App.ShowProjectChooserForm( _
PCF_ALLOWMULTISEL + PCF_ALL)
If Not Projs Is Nothing Then
      For Each Subproj In Projs
         Proj.AddSubProject Subproj
      Next
   End If
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProject" on page 497

IProject::Caption

The property returns or sets the text of the title bar in the folder form.

C++ Syntax

```
HRESULT Caption(  BSTR* pbstrCaption);
HRESULT Caption(  BSTR bstrCaption);
```

Visual Basic Syntax

```
[ pbstrCaption = ] Project.Caption
Project.Caption [ = bstrCaption ]
```

Parameters

pbstrCaption	[out, retval] A string that specifies the folder form's caption.
bstrCaption	[in] A string that specifies the folder form's caption.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The folder caption can be set for the current session. If the caption needs to be changed, it should be set every time the user runs the program.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProject" on page 497

IProject::Delete

The method deletes a folder.

C++ Syntax

```
HRESULT Delete(  short sFlags,  VARIANT_BOOL* pbSuccess);
```

Visual Basic Syntax

```
[ pbSuccess  = ] Project.Delete (sFlags)
```

Parameters

sFlags	[in] Required. An integer value that specifies how the method should be executed. The parameter can be a combination of the MethodsFlags values.
pbSuccess	[out, retval] Returns a Boolean value that specifies whether or not the operation succeeded. True: A folder was successfully deleted. False: A folder was not deleted.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

After deleting, a Project object becomes undefined, i.e. no properties or methods can be called on the object before initializing it again with the SetProject method.

The user should have enough rights to be able to delete a folder. Folder should not be a member of another folder.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProject" on page 497

"IProject::SetProject" on page 516

"MethodsFlags Enumeration" on page 689

IProject::DisplayModal

The method displays the folder form to the user.

C++ Syntax

```
HRESULT DisplayModal(  VARIANT_BOOL bAllowEdits,  VARIANT_BOOL*
psResult);
```

Visual Basic Syntax

```
[ psResult  = ] Project.DisplayModal (bAllowEdits)
```

Parameters

bAllowEdits	[in] Required. A Boolean value that specifies whether or not folder properties can be changed. True: Folder properties can be changed. False: Folder properties cannot be changed.
psResult	[out, retval] Returns a Boolean value that specifies whether or not folder data was changed. True: Changes were made to the folder. False: No changes were made to the folder.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

If changes are made to the folder when displaying its form and the user clicks OK, the program does not need to call the Save method because the changes are saved automatically.

The folder form can be customized and should support the IProjectForm interface. To display the form, the method calls up theShowProjectForm method on IProjectForm.

The folder form is displayed modally, meaning that the window has exclusive focus in the application until it has been closed or an option has been chosen.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProject" on page 497

"IProject::Save" on page 514

"IProjectForm" on page 921

"IProjectForm::ShowProjectForm" on page 921

IProject::Icon

The property returns the icon of a folder.

C++ Syntax

```
HRESULT Icon(  long* plIcon);
```

Visual Basic Syntax

```
[ plIcon = ] Project.Icon
```

Parameters

pllcon	[out, retval] Returns a long that specifies the HICON value of the folder's icon. It is caller responsibility to destroy returned icon.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns the folder icon.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProject" on page 497

IProject::Iconized

The property returns or sets whether or not a folder appears under the Public Folders node of the DM Extensions tree.

C++ Syntax

```
HRESULT Iconized(  VARIANT_BOOL* pbIconized);
HRESULT Iconized(  VARIANT_BOOL bIconized);
```

Visual Basic Syntax

```
[ pbIconized = ] Project.Iconized
Project.Iconized [ = bIconized ]
```

Parameters

pbIconized	[out, retval] A Boolean value that specifies whether or not a folder is a Public Folder.
bIconized	[in] A Boolean value that specifies whether or not a folder is a Public Folder. True: Folder is a Public Folder. False: Folder isn't a Public Folder.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns/sets whether a project is iconized on the desktop.

Example

The following example creates a new folder on the primary library.

```
Dim App As New Application
Dim QR As New QuickRetrieve, Prj As Project

' Create a new folder on the primary library
Set Prj = App.PrimaryLibrary.CreateProject
If Not Prj Is Nothing Then
Prj.Iconized = True
```

```
If Prj.DisplayModal(True) Then
Prj.Save OF_NORMAL
End If
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProject" on page 497

IProject::Library

The property returns a Library object on which a folder exists.

C++ Syntax

```
HRESULT Library(  ILibrary** ppLibrary);
```

Visual Basic Syntax

```
[ Set ppLibrary = ] Project.Library
```

Parameters

pbCopy	[out, retval] Returns a Library object that specifies the library the folder is assigned to.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns the folder library.

Example

The following example makes the parent Library object of a folder to be the current library.

```
Dim Prj As Project
' Some operations with the folder
...
Prj.Library.MakeCurrent
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProject" on page 497

"ILibrary" on page 360

IProject::Name

The property returns or sets display name for folder in container - another folder or in Public Folder node.

C++ Syntax

```
HRESULT Name(  BSTR* pbstrName);
HRESULT Name(  BSTR bstrName);
```

Visual Basic Syntax

```
[ pbstrName = ] Project.Name
Project.Name [ = bstrName ]
```

Parameters

pbstrName	[out, retval] A string that specifies the display name.
bstrName	[in] A string that specifies the display name.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns/sets the name of a project.

Example

The following example creates a new project on the primary library.

```
Dim App As New Application
Dim QR As New QuickRetrieve, Prj As Project
```

```
' Create a new project on the primary library
Set Prj = App.PrimaryLibrary.CreateProject
If Not Prj Is Nothing Then
Prj.Name = "My Project"
Prj.Iconized = True
Prj.Save OF_NORMAL
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProject" on page 497

IProject::ProjectID

The property returns the identifier of a folder (profiled document number).

C++ Syntax

```
HRESULT ProjectID(  long* plID);
```

Visual Basic Syntax

```
[ plID = ] Project.ProjectID
```

Parameters

pllD	[out, retval] Returns a long that uniquely identifies the folder.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The folder identifier (profile document number) will be returned if this is an existing folder. The property will return zero (0) if this is a new folder that has not been saved. The folder identifier is unique within a library, i.e. two folders on the same library have different identifiers.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProject" on page 497

IProject::RemoveDocument

The method removes a document from the folder.

C++ Syntax

```
HRESULT RemoveDocument(  VARIANT vDocument,  VARIANT_BOOL*
pbSuccess);
```

Visual Basic Syntax

```
[ pbSuccess  = ] Project.RemoveDocument (vDocument)
```

Parameters

vDocument	[in] Required. An expression that specifies a document profile from the folder. If an object, vDocument should specify a valid Profile object. If a numeric expression, vDocument should specify the number of a document. If a string expression, vDocument should specify the document name.
pbSuccess	[out, retval] Returns a Boolean value that specifies whether or not a document was successfully removed from the folder. True: A document was successfully removed from the folder. False: A document was not removed from the folder.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The method does not delete the document from the system, it simply revokes the document's membership in the folder. The document will still exist on the library.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProject" on page 497

"IProfile" on page 419

IProject::RemoveSubProject

The method removes a sub-folder from the folder.

C++ Syntax

```
HRESULT RemoveSubProject(  VARIANT vSubproject,  VARIANT_BOOL*
pbSuccess);
```

Visual Basic Syntax

```
[ pbSuccess  = ] Project.RemoveSubProject (vSubproject)
```

Parameters

vSubproject	[in] Required. An expression that specifies a sub-folder to be removed from the folder. If an object, vSubproject should specify a valid Project object. If a numeric expression, vSubproject should specify the identifier of a sub-folder. If a string expression, vSubproject should specify the sub-folder name.
pbSuccess	[out, retval] Returns a Boolean value that specifies whether or not a sub-folder was successfully removed from the folder. True: A sub-folder was successfully removed from the folder. False: A sub-folder was not removed from the folder.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The method does not delete the folder from the system; it simply revokes the sub-folder's membership in the folder. The folder will still exist on the library.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProject" on page 497

IProject::Save

The method saves a new folder or changes to an existing folder.

C++ Syntax

```
HRESULT Save(  short sFlags,  VARIANT_BOOL* pbSuccess);
```

Visual Basic Syntax

```
[ pbSuccess  = ] Project.Save (sFlags)
```

Parameters

sFlags	[in] Required. An integer value that specifies how the method should be executed. The parameter can be a combination of the MethodsFlags values.
pbSuccess	[out, retval] Returns a Boolean value that specifies whether or not the operation succeeded. True: A folder was successfully saved. False: A folder was not saved.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Only the name and iconized/favorite property changes are saved. Changes made to the documents and sub-folders in the folder are saved immediately. If the caption is changed, it will need to be reset when the user begins the next session.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProject" on page 497

"MethodsFlags Enumeration" on page 689

IProject::SelectedDocuments

The property returns or sets a collection of highlighted or selected documents on the folder form.

C++ Syntax

```
HRESULT SelectedDocuments(  IDocProfiles** ppDocs);
HRESULT SelectedDocuments(  IDocProfiles* pDocs);
```

Visual Basic Syntax

```
[ Set ppDocs = ] Project.SelectedDocuments
Set Project.SelectedDocuments [ =  ]
```

Parameters

ppDocs	[out, retval] A DocProfiles object that represents the collection of selected documents.
pDocs	[in] A DocProfiles object that represents the collection of selected documents.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns/sets selected documents.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProject" on page 497

"IDocProfiles" on page 292

IProject::SelectedSubProjects

The property returns or sets a collection of highlighted or selected sub-folders on the folder form.

C++ Syntax

```
HRESULT SelectedSubProjects(  IProjects** ppSubprojects);
HRESULT SelectedSubProjects(  IProjects* pSubprojects);
```

Visual Basic Syntax

```
[ Set ppSubprojects = ] Project.SelectedSubProjects
Set Project.SelectedSubProjects [ =  ]
```

Parameters

ppSubprojects	[out, retval] A Projects object that represents the collection of selected sub-folders.
pSubprojects	[in] A Projects object that represents the collection of selected sub-folders.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns/sets selected sub-folders.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProject" on page 497

"IProjects" on page 534

IProject::SetProject

The method initializes a Project object to a particular folder.

C++ Syntax

```
HRESULT SetProject(  VARIANT vLibrary,  VARIANT vProject);
```

Visual Basic Syntax

```
Project.SetProject (vLibrary, vProject)
```

Parameters

vLibrary	[in] Required. An expression that specifies a library. If a string expression, vLibrary should specify the library name. If an object, vLibrary should specify a valid Library object.
vProject	[in] Required. An expression that specifies a folder on vLibrary. If a numeric expression, vProject should specify the identifier of a folder. If a string expression, vProject should specify the folder name.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

If the folder was created with the CreateObject function or the Dim statement with the New keyword, the method is to be called up before any other method or property.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProject" on page 497

"ILibrary" on page 360

IProject::ShowAccessForm

The method displays the folder's access control form.

C++ Syntax

```
HRESULT ShowAccessForm(  VARIANT_BOOL bAllowEdits,  short*
psResult);
```

Visual Basic Syntax

```
[ psResult  = ] Project.ShowAccessForm (bAllowEdits)
```

Parameters

bAllowEdits	[in] Required. A Boolean value that specifies whether or not the folder's trustee list can be changed. True: The trustee list can be changed. False: The trustee list cannot be changed.
psResult	[out, retval] Returns an integer that specifies user's action on the form. For available values see FormsResults Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The folder's access control form can be customized and should support the IProfileAccessForm interface. To display the form, the method calls up theShowProfileAccessForm method onIProfileAccessForm.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProject" on page 497

"IProfileAccessForm" on page 894

"IProfileAccessForm::ShowProfileAccessForm" on page 895

"FormsResults Enumeration" on page 686

IProject::SubDocuments

The property returns a collection of document profiles belonging to a folder.

C++ Syntax

```
HRESULT SubDocuments(  IDocProfiles** ppDocs);
```

Visual Basic Syntax

```
[ Set ppDocs = ] Project.SubDocuments
```

Parameters

ppDocs	[out, retval] Returns a DocProfiles object that represents the collection of document profiles.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns a collection of all documents included in a folder.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProject" on page 497

"IDocProfiles" on page 292

IProject::SubProjects

The property returns a collection of all sub-folders in a folder.

C++ Syntax

```
HRESULT SubProjects(  IProjects** ppProjects);
```

Visual Basic Syntax

```
[ Set ppProjects = ] Project.SubProjects
```

Parameters

ppProjects	[out, retval] Returns a Projects object that represents a collection of sub-folders.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns a collection of all sub-folders included in a project.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProject" on page 497

"IProjects" on page 534

IProject::Trustees

The property returns a collection of users and groups having access to a folder.

C++ Syntax

```
HRESULT Trustees(  ITrusteeCollection** ppTrustees);
```

Visual Basic Syntax

```
[ Set ppTrustees = ] Project.Trustees
```

Parameters

ppTrustees	[out, retval] Returns a TrusteeCollection object that represents a collection of folder's trustees.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Returns a collection of folder trustees.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProject" on page 497

"ITrusteeCollection" on page 609

IProject2

The IProject2 interface extends Project object functionality.

The interface inherits its definition from IProject.

Methods

IProject2 Methods	Description
IProject2::InsertDocument	Inserts a document to a project into specified position.
IProject2::InsertSubProject	Inserts a subproject to a project into specified position.

Properties

IProject2 Properties	Description
IProject2::DocumentOrder	Returns/sets the document order inside project.
IProject2::SubProjectOrder	Returns/sets the subproject order inside project.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "fsplugin.dll"

See Also

"IProject" on page 497

IProject2::DocumentOrder

Returns/sets the document order inside a folder.

C++ Syntax

```
HRESULT DocumentOrder(  VARIANT vDocument,  long* pnOrder);
HRESULT DocumentOrder(  VARIANT vDocument,  long nOrder);
```

VBScript Syntax

```
[ pnOrder = ] Project2.DocumentOrder (vDocument)
Project2.DocumentOrder (vDocument) [ = nOrder ]
```

Parameters

vDocument	[in] Specifies the document whose order should be set/returned.
pnOrder	[out, retval] Specifies an address where an order number should be returned.
nOrder	[in] Specifies an order number, which should be assigned.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

In the case when an order is set, all subsequent components will be reordered. If returned value is S_FALSE methods of IOMErrInfo should be used for error description.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <fsplugin.dll>

See Also

"IProject2" on page 521

IProject2::InsertDocument

The method adds the document to a folder into the specified position.

C++ Syntax

```
HRESULT InsertDocument(  VARIANT vDocument,  long nOrder,
VARIANT_BOOL* pbSuccess);
```

VBScript Syntax

```
[ pbSuccess  = ] Project2.InsertDocument (vDocument, nOrder)
```

Parameters

vDocument	[in] Specifies the document to insert.
nOrder	[in] Specifies the position where a new document should be inserted.
pbSuccess	[out, retval] Specifies an address where an operation result should be returned.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

If insertion succeeded all subsequent components will be reordered. If returned value is S_FALSE methods of IOMErrInfo should be used for error description. If specified document already exists in this folder, only its order will be changed. If nOrder is negative or greater or equal than components count in a folder, it will be inserted as last component in a folder.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <fsplugin.dll>

See Also

"IProject2" on page 521

"IProfile" on page 419

IProject2::InsertSubProject

The method adds a sub-folder to the folder into the specified position.

C++ Syntax

```
HRESULT InsertSubProject(  VARIANT vSubProject,  long nOrder,
VARIANT_BOOL* pbSuccess);
```

VBScript Syntax

```
[ pbSuccess  = ] Project2.InsertSubProject (vSubProject, nOrder)
```

Parameters

vSubProject	[in] Specifies a sub-folder to insert.
nOrder	[in] Specifies the position where a new sub-folder should be inserted.
pbSuccess	[out, retval] Specifies an address where an operation result should be returned.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

If insertion succeeded all subsequent components will be reordered. If returned value is S_FALSE methods of IOMErrInfo should be used for error description. If specified sub-folder already exists in this folder, only its order will be changed. If nOrder negative or greater or equal than components count in folder, it will be inserted as a last component in the folder.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <fsplugin.dll>

See Also

"IProject2" on page 521

"IProfile" on page 419

IProject2::SubProjectOrder

Returns/sets the sub-folder order inside a folder.

C++ Syntax

```
HRESULT SubProjectOrder(  VARIANT vSubProject,  long* pnOrder);
HRESULT SubProjectOrder(  VARIANT vSubProject,  long nOrder);
```

VBScript Syntax

```
[ pnOrder = ] Project2.SubProjectOrder (vSubProject)
Project2.SubProjectOrder (vSubProject) [ = nOrder ]
```

Parameters

vDocument	[in] Specifies the sub-folder whose order should be set/returned.
pnOrder	[out, retval] Specifies an address where an order number should be returned.
nOrder	[in] Specifies an order number, which should be assigned.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

In the case when an order is set, all subsequent components will be reordered. If returned value is S_FALSE methods of IOMErrInfo should be used for error description.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <fsplugin.dll>

See Also

"IProject2" on page 521

IProject3

Methods

IProject3 Methods	Description
IProject3::InsertComponent	Inserts new component into folder.
IProject3::ChangeVersionTypeForm	Displays Change Version Type form for folder components.
IProject3::ChangeDisplayNameForm	Displays Change Display Name form for folder component.

Properties

IProject3 Properties	Description
IProject3::ProfiledDocument	Returns the Profile object for folder.
IProject3::Favorite	Returns/sets whether a project is in Favorites Folders.
IProject3::DocumentVersionType	Returns/sets the component version type inside folder.
IProject3::ComponentDisplayName	Returns/sets the component display name inside folder.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import "fsplugin.dll"

IProject3::ChangeDisplayNameForm

The method inserts new component into folder.

C++ Syntax

```
HRESULT ChangeDisplayNameForm(  sort sFlags,  IProfile* pDocument,
VARIANT_BOOL* pbSuccess);
```

VBScript Syntax

```
[ pbSuccess = ] Project3.ChangeDisplayNameForm ( sFlags, pDocument
)
```

Parameters

sFlags	[in] Reserved. Should be 0.
pDocument	[in] An expression that specifies profiled item.
pbSuccess	[out, retval] Specifies an address where an operation result should be returned. True: Display name of component(s) was successfully changed. False: Display name of component(s) was not changed.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IProject3" on page 526

IProject3::ChangeVersionTypeForm

The method inserts new component into folder.

C++ Syntax

```
HRESULT ChangeVersionTypeForm(  sort sFlags,  VARIANT vDocument,
VARIANT_BOOL* pbSuccess);
```

VBScript Syntax

```
[ pbSuccess = ] Project3.ChangeVersionTypeForm ( sFlags, vDocument
)
```

Parameters

sFlags	[in] Reserved. Should be 0.
vDocument	[in] Specifies the document to insert.
pbSuccess	[out, retval] Specifies an address where an operation result should be returned. True: Version type of component(s) was successfully changed. False: Version type of component(s) was not changed.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IProject3" on page 526

IProject3::ComponentDisplayName

Returns/set the whether the FolderProfile object for folder.

C++ Syntax

```
HRESULT ComponentDisplayName(  VARIANT vDocument,  BSTR*
pbstrDisplayName);

HRESULT ComponentDisplayName(  VARIANT vDocument,  BSTR
bstrDisplayName);
```

VBScript Syntax

```
[ pbstrDisplayName = ] Project3.ComponentDisplayName (vDocument)

Project3.ComponentDisplayName (vDocument) [ = bstrDisplayName ]
```

Parameters

vDocument	[in] Specifies the document whose order should be set/returned.
pbstrDisplayName	[out, retval] Returns a display name of component.
bstrDisplayName	[in] Sets a display name of component..

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IProject3" on page 526

IProject3::DocumentVersionType

Returns/set the whether the Folde Profile object for folder.

C++ Syntax

```
HRESULT DocumentVersionType(  VARIANT vDocument,  VARIANT*
pvDocumentVersionType);

HRESULT DocumentVersionType(  VARIANT vDocument,  VARIANT
vDocumentVersionType);
```

VBScript Syntax

```
[ pvDocumentVersionType = ] Project3.DocumentVersionType
(vDocument)

Project3.DocumentVersionType (vDocument) [ = vDocumentVersionType
]
```

Parameters

vDocument	[in] Specifies the document whose order should be set/returned.
pvDocumentVersionType	[out, retval] Returns a version type of component. R: Current (most recent) version P: Published (last published) version VersionLabel: Specific version
vDocumentVersionType	[in] Sets a version type of component.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IProject3" on page 526

IProject3::Favorite

Returns/sets whether a project is in Favorites Folders.

C++ Syntax

```
HRESULT Favorite(  VARIANT_BOOL* pbFavorite);

HRESULT Favorite(  VARIANT_BOOL bFavorite);
```

VBScript Syntax

```
[ pbFavorite = ] Project3.Favorite

Project3.Favorite [ = bFavorite ]
```

Parameters

pbFavorite	[out, retval] Returns a Boolean value that indicate whether the folder is a favorite folder.
bFavorite	[in] Sets a Boolean value that indicate whether the folder is a favorite folder.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IProject3" on page 526

IProject3::InsertComponent

The method inserts new component into folder.

C++ Syntax

```
HRESULT InsertComponent(  VARIANT vDocument,  long nOrder,
VARIANT vVersion,  BSTR strDisplayName,  VARIANT_BOOL* pbSuccess);
```

VBScript Syntax

```
[ pbSuccess  = ] Project3.InsertComponent (vDocument, nOrder,
vVersion, strDisplayName)
```

Parameters

vDocument	[in] Specifies the document to insert.
nOrder	[in] Specifies the position where a new document should be inserted.
vVersionType	[in] Specifies version type of inserted component.
strDisplayName	[in] Specify component display name.
pbSuccess	[out, retval] Specifies an address where an operation result should be returned. True: Component was successfully added. False: Component was not added.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IProject3" on page 526

IProject3::ProfiledDocument

Returns the Profile object for folder.

C++ Syntax

```
HRESULT ProfiledDocument(  IProfile** ppProfiledDocument);
```

VBScript Syntax

```
[ Set ppProfiledDocument = ] Project3.ProfiledDocument
```

Parameters

ppProfiledDocument	[out, retval] Specifies address where Profile object should be returned.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Profile object also can be retrieve via QueryInterface method.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IProject3" on page 526

IProjects

The Projects object represents a collection of folders.

When to Use

A Projects object may be created with the CreateObject function or the Dim statement with the New keyword. The following example shows the Visual Basic syntax for creating a DOCSObjects.Projects object:

```
Sub Form_Load()
 ...
 Dim Prjs As Object
 ' Create a DOCSObjects.Projects object
 Set Prjs = CreateObject("DOCSObjects.Projects")
 ...
 End Sub
```

Methods

IProjects Methods	Description
IProjects::Add	The method adds a new item to a collection.
IProjects::Clear	The method removes all items from a collection.
IProjects::Remove	The method removes a specific item from a collection.

Properties

IProjects Properties	Description
IProjects::Count	The property returns the number of items in a collection.
IProjects::Item	The property returns a specific member of a collection either by position or by key.
IProjects::_NewEnum	Returns an object that supports IEnumVARIANT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "fsplugin.dll"

See Also

"IProject" on page 497

IProjects::Add

The method adds a new item to a collection.

C++ Syntax

```
HRESULT Add(  IProject* pProject);
```

VBScript Syntax

```
Projects.Add (pProject)
```

Parameters

pProject	[in] Required. A Project object that specifies the item to add to a collection.

Return Values

S_OK	The operation succeeded.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProjects" on page 534

"IProfile" on page 419

IProjects::Clear

The method removes all items from a collection.

C++ Syntax

```
HRESULT Clear();
```

VBScript Syntax

```
Projects.Clear
```

Return Values

S_OK	The operation succeeded.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProjects" on page 534

"IProfile" on page 419

IProjects::Count

The property returns the number of items in a collection.

C++ Syntax

```
HRESULT Count(  long* plCount);
```

VBScript Syntax

```
[ plCount = ] Projects.Count
```

Parameters

plCount	[out, retval] Returns a long that specifies the number of items in a collection.

Return Values

S_OK	The operation succeeded.
E_UNEXPECTED	Unexpected error occurs.

Access:

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProjects" on page 534

IProjects::Item

The property returns the specific member of a collection either by position or by key.

C++ Syntax

```
HRESULT Item(  VARIANT vIndex,  IProject** ppProject);
```

VBScript Syntax

```
[ Set ppProject = ] Projects.Item (vIndex)
```

Parameters

vIndex	[in] Required. An expression that specifies the member position in a collection. If a numeric expression, vIndex should be a number from 1 to the value of the collection's Count property. If a string expression, vIndex should correspond to a folder name.
ppProject	[out, retval] Returns a Project object that specifies the requested item.

Return Values

S_OK	The operation succeeded.
E_UNEXPECTED	Unexpected error occurs.

Access:

Read Only

Remarks

If the value provided as vIndex does not match any existing member of the collection, an error occurs.

Item is the default property for a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProjects" on page 534

IProjects::Remove

The method removes the specific item from a collection.

C++ Syntax

```
HRESULT Remove(  VARIANT vIndex);
```

VBScript Syntax

```
Projects.Remove (vIndex)
```

Parameters

vIndex	[in] Required. A variant expression specifying the name or index in a collection of the object to be accessed.

Return Values

S_OK	The operation succeeded.
E_INAVLIDARG	Invalid parameter has been passed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProjects" on page 534

"IProfile" on page 419

IProjects::_NewEnum

The _NewEnum property returns the IEnumVARIANT interface on an object that can be used to enumerate the collection.

C++ Syntax

```
HRESULT _NewEnum(LPUNKNOWN* ppEnum);
```

Parameters

ppEnum	[out, retval] An address of the IUnknown* pointer variable that receives the interface pointer to the enumeration object. Each time the collection receives a successful call to _NewEnum, it should increase the reference count on the *ppEnum pointer the property returns. It is the caller's responsibility to call IUnknown::Release when it is done with the pointer. If an error occurs, the implementation should set *ppEnum to NULL

Return Values

S_OK	The operation succeeded.
E_OUTOFMEMORY	There is not enough memory to complete the operation.
E_INVALIDARG	ppEnum is not a valid pointer.
E_UNEXPECTED	An unexpected error occurred.

Remarks

It is the standard property of a collection. It returns an enumeration object, which implements the IEnumVariant interface. This property should have the DISPID_NEWENUM identifier. The user can use the enumeration interface to step through the objects in the collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProjects" on page 534

IProjectTrustee

Obsolete. IProfileTrustee object is used to access folder trustees.

The ProjectTrustee object represents a user or group who has access to a project. The ProjectTrustee object may be retrieved through the Trustees collection on the project.

Object's properties can be modified only if the user has edit access to the project. When the user who does not have enough rights attempts to make changes in the access rights, an error will occur.

Properties

IProjectTrustee Properties	Description
CanDeleteProject	The property returns or sets whether or not a trustee can delete a specific project.
CanEditProject	The property returns or sets whether or not a trustee can edit a specific project.
CanViewProject	The property returns or sets whether or not a trustee can see a specific project and view its contents.
IsGroup	The property returns whether or not a trustee is a group.
Name	The property returns the DM trustee's username or group name.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "fsplugin.dll"

See Also

"IProject" on page 497

IProjectTrustee::CanDeleteProject

The property returns or sets whether or not the trustee can delete a specific project.

C++ Syntax

```
HRESULT CanDeleteProject(  VARIANT_BOOL* pbDelete);
HRESULT CanDeleteProject(  VARIANT_BOOL bDelete);
```

VBScript Syntax

```
[ pbDelete = ] ProjectTrustee.CanDeleteProject
ProjectTrustee.CanDeleteProject [ = bDelete ]
```

Parameters

pbDelete	[out, retval] A Boolean value that specifies whether or not the trustee has rights to delete a project.
bDelete	[in] A Boolean value that specifies whether or not the trustee has rights to delete a project. True: The trustee has rights to delete the project.

Return Values

S_OK	The operation succeeded.
E_UNEXPECTED	Unexpected error occurs.

Access:

Read/Write

Remarks

A trustee cannot have delete access without also having edit and view access.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProjectTrustee" on page 540

IProjectTrustee::CanEditProject

The property returns or sets whether or not the trustee can edit a specific project.

C++ Syntax

```
HRESULT CanEditProject(  VARIANT_BOOL* pbEdit);
HRESULT CanEditProject(  VARIANT_BOOL bEdit);
```

VBScript Syntax

```
[ pbEdit = ] ProjectTrustee.CanEditProject
ProjectTrustee.CanEditProject [ = bEdit ]
```

Parameters

pbEdit	[out, retval] A Boolean value that specifies whether or not the trustee has rights to edit a project.
bEdit	[in] A Boolean value that specifies whether or not the trustee has rights to edit a project. True: The trustee has rights to edit a project.

Return Values

S_OK	The operation succeeded.
E_UNEXPECTED	Unexpected error occurs.

Access:

Read/Write

Remarks

If the trustee has been granted edit rights on a project he/she can change the name and iconized properties of the project as well as add and remove documents and subprojects from the project and modify project's trustee list. A user cannot have edit access without also having view access.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProjectTrustee" on page 540

IProjectTrustee::CanViewProject

The property returns or sets whether or not the trustee can see a specific project and view its contents.

C++ Syntax

```
HRESULT CanViewProject(  VARIANT_BOOL* pbView);
HRESULT CanViewProject(  VARIANT_BOOL bView);
```

VBScript Syntax

```
[ pbView = ] ProjectTrustee.CanViewProject
ProjectTrustee.CanViewProject [ = bView ]
```

Parameters

pbView	[out, retval] A Boolean value that specifies whether or not the trustee has rights to view a project and its contents.
bView	[in] A Boolean value that specifies whether or not the trustee has rights to view a project and its contents. True: The trustee has rights to view a project and its contents.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Access:

Read/Write

Remarks

If a trustee is given access to view a project, he/she will still not be able to retrieve, view, or change any document to which he has not been granted access. If the library uses profile level security, those documents that the trustee has been restricted from seeing will still be hidden from him. If a trustee does not have view access then he/she cannot also have delete or edit access to the project.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProjectTrustee" on page 540

IProjectTrustee::IsGroup

The property returns whether or not the trustee is a group.

C++ Syntax

```
HRESULT IsGroup(  VARIANT_BOOL* pbGroup);
```

VBScript Syntax

```
[ pbGroup = ] ProjectTrustee.IsGroup
```

Parameters

pbGroup	[out, retval] Returns a Boolean value that specifies whether or not the trustee is a group. True: The trustee is a group.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Access:

Read Only

Remarks

The IsGroup property is useful for determining which Trustees methods to use with a Trustee.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProjectTrustee" on page 540

IProjectTrustee::Name

The property returns the DM trustee's username or group name.

C++ Syntax

```
HRESULT Name(  BSTR* pbstrName);
```

VBScript Syntax

```
[ pbstrName = ] ProjectTrustee.Name
```

Parameters

pbstrName	[out, retval] Returns a string that specifies the trustee's name.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Access:

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProjectTrustee" on page 540

IProperties

The Properties object represents a collection of all persistent properties supported by another object, e.g., Library. The collection may be retrieved with the Properties property on that object.

Properties

IProperties Properties	Description
Count	The property returns the number of items in a collection.
Item	The property returns the specific member of a collection either by position or by key.
_NewEnum	Returns an object that supports IEnumVARIANT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "fsplugin.dll"

See Also

"IProperty" on page 551

IProperties::Count

The property returns the number of items in a collection.

C++ Syntax

```
HRESULT Count(  long* plCount);
```

VBScript Syntax

```
[ plCount = ] Properties.Count
```

Parameters

plCount	[out, retval] Returns a long that specifies the number of items in a collection.

Return Values

S_OK	The operation succeeded.
E_UNEXPECTED	Unexpected error occurs.

Access:

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProperty" on page 551

"IProperties" on page 546

IProperties::Item

The property returns the specific member of a collection either by position or by key.

C++ Syntax

```
HRESULT Item(  VARIANT vIndex,  IProperty** ppProp);
```

VBScript Syntax

```
[ Set ppProp = ] Properties.Item (vIndex)
```

Parameters

vIndex	[in] Required. An expression that specifies the member position in a collection. If a numeric expression, vIndex should be a number from 1 to the value of the collection's Count property. If a string expression, vIndex should correspond to a persistent property's name.
ppProp	[out, retval] Returns a Property object that specifies the requested item.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_INAVLIDARG	An invalid parameter has been passed.
E_UNEXPECTED	Unexpected error occurs.

Access:

Read Only

Remarks

If the value provided as vIndex does not match any existing member of a collection, an error occurs.

Item is the default property for a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProperty" on page 551

"IProperties" on page 546

IProperties::_NewEnum

The _NewEnum property returns the IEnumVARIANT interface on an object that can be used to enumerate the collection.

C++ Syntax

```
HRESULT _NewEnum(LPUNKNOWN* ppEnum);
```

Parameters

ppEnum	[out, retval] An address of the IUnknown* pointer variable that receives the interface pointer to the enumeration object. Each time the collection receives a successful call to _NewEnum, it should increase the reference count on the *ppEnum pointer the property returns. It is the caller's responsibility to call IUnknown::Release when it is done with the pointer. If an error occurs, the implementation should set *ppEnum to NULL

Return Values

S_OK	The operation succeeded.
E_OUTOFMEMORY	There is not enough memory to complete the operation.
E_INVALIDARG	ppEnum is not a valid pointer.
E_UNEXPECTED	An unexpected error occurred.

Remarks

It is the standard property of a collection. It returns an enumeration object, which implements the IEnumVariant interface. This property should have the DISPID_NEWENUM identifier. The user can use the enumeration interface to step through the objects in the collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProperties" on page 546

IProperty

The Property object represents a persistent property of another object, e.g., Library. For the list of all available persistent properties see the corresponding parent object.

Properties

IProperty Properties	Description
Name	The property returns the name of a Property object.
ReadOnly	The property returns whether or not the persistent property's value can be changed.
Type	The property returns the data type of a persistent property.
Value	The property returns or sets the value of a Property object.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "fsplugin.dll"

See Also

"IProject" on page 497

IProperty::Name

The property returns the name of a Property object.

C++ Syntax

```
HRESULT Name(  BSTR* pbstrName);
```

VBScript Syntax

```
[ pbstrName = ] Property.Name
```

Parameters

pbstrName	[out, retval] Returns a string that specifies the persistent property's unique name.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Access:

Read Only

Remarks

The name can be used to identify the object in a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProperty" on page 551

IProperty::ReadOnly

The property returns whether or not the persistent property's value can be changed.

C++ Syntax

```
HRESULT ReadOnly(  VARIANT_BOOL* pbReadOnly);
```

VBScript Syntax

```
[ pbReadOnly = ] Property.ReadOnly
```

Parameters

pbReadOnly	[out, retval] Returns a Boolean value that specifies whether or not the persistent property is read-only. True: The persistent property is read-only.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Access:

Read Only

Remarks

If the persistent property is read-only, it cannot be assigned a value by the Value property.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProperty" on page 551

IProperty::Type

The property returns the data type of a persistent property.

C++ Syntax

```
HRESULT Type(  short* psType);
```

VBScript Syntax

```
[ psType = ] Property.Type
```

Parameters

psType	[out, retval] Returns an integer that specifies the persistent property's type. It can be one of the DataTypes values.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Access:

Read Only

Remarks

The persistent property's types are defined through DM and not through the database engine even though the database may have similar field types. This frees the programmer from worrying about the various data types on different database servers, since DM handles the data types.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProperty" on page 551

IProperty::Value

The property returns or sets the value of a Property object.

C++ Syntax

```
HRESULT Value(  VARIANT* pvValue);HRESULT Value(  VARIANT vValue);
```

VBScript Syntax

```
[ pvValue = ] Property.Value
```

Parameters

pvValue	[out, retval] An expression that specifies the persistent property's value.
vValue	[in] An expression that specifies the persistent property's value. The type of this parameter should conform to the persistent property's type returned by the Type property.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Access:

Read/Write

Remarks

The persistent property's value cannot be changed if it is read-only, i.e. if its ReadOnly property returns True.

The Value property is the default property for a persistent property object.

Example

The following example indicates whether or not the user can delete documents on the primary library.

```
Dim App As New Application
Dim bCanDelete As Boolean
bCanDelete = App.PrimaryLibrary.Properties("User.DocDelete")
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProperty" on page 551

IQuickRetrieve

A QuickRetrieve object represents the DM Extensions Quick Retrieve form.

When to Use

A QuickRetrieve object may be created with the CreateObject function or the Dim statement with the New keyword. The following example shows the Visual Basic syntax for creating a DOCSObjects.QuickRetrieve object:

```
Sub Form_Load()
 ...
 Dim QR As Object
 ' Create a DOCSObjects.QuickRetrieve object
 Set QR = CreateObject("DOCSObjects.QuickRetrieve")
 ...
 End Sub
```

A QuickRetrieve object inherits its definition from the IOMErrInfo interface so its implementation also includes all the properties of IOMErrInfo in addition to ones described below.

Methods

IQuickRetrieve Methods	Description
IQuickRetrieve::DisplayModal	The method displays the Quick Retrieve form.

Properties

IQuickRetrieve Properties	Description
IQuickRetrieve::Application	The property returns the Application object to which a QuickRetrieve object belongs.
IQuickRetrieve::Caption	The property returns or sets the text of the title bar in the Quick Retrieve form.
IQuickRetrieve::DocumentCount	The property returns the number of documents currently contained in the Quick Retrieve form.
IQuickRetrieve::Documents	The property returns a collection of document profiles in the current list.
IQuickRetrieve::FormName	The property returns the name of the Quick Retrieve form.
IQuickRetrieve::Search	The property returns or sets a Search object whose results the Quick Retrieve form will display.
IQuickRetrieve::SelectedDocuments	The property returns or sets a collection of highlighted or selected documents in the current list.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "fsplugin.dll"

See Also

"IProject" on page 497

IQuickRetrieve::Application

The property returns the Application object to which a QuickRetrieve object belongs.

C++ Syntax

```
HRESULT Application(  IApplication** ppApp);
```

VBScript Syntax

```
[ Set ppApp = ] QuickRetrieve.Application
```

Parameters

ppApp	[out, retval] Returns the Application object that specifies the object's parent.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access:

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IQuickRetrieve" on page 556

IQuickRetrieve::Caption

The property returns or sets the text of the title bar in the Quick Retrieve form.

C++ Syntax

```
HRESULT Caption(  BSTR* pbstrCaption);
HRESULT Caption(  BSTR bstrCaption);
```

VBScript Syntax

```
[ pbstrCaption = ] QuickRetrieve.Caption
QuickRetrieve.Caption [ = bstrCaption ]
```

Parameters

pbstrCaption	[out, retval] A string that specifies the form caption.
bstrCaption	[in] A string that specifies the form caption.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access:

Read/Write

Remarks

The Quick Retrieve form's caption can be set for the current session. If the caption needs to be changed, it should be set every time the user runs the program.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IQuickRetrieve" on page 556

IQuickRetrieve::DisplayModal

The method displays the Quick Retrieve form.

C++ Syntax

```
HRESULT DisplayModal(  short sFlags,  short* psResult);
```

VBScript Syntax

```
[ psResult  = ] QuickRetrieve.DisplayModal (sFlags)
```

Parameters

sFlags	[in, optional] Optional. An integer value that specifies the form appearance. The parameter can be one of the OMFlagsQR values. By default the form allows multiple selections.
psResult	[out, retval] Returns an integer that specifies user's action on the form. For possible values see FormsResults Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The Quick Retrieve form can be customized and should support the IQuickRetrieveForm interface. To display the form, the method calls up the ShowQuickRetrieveForm method on IQuickRetrieveForm.

The Quick Retrieve form is displayed modally, meaning that the window has exclusive focus in the application until it has been closed or an option has been chosen.

Example

The following example creates a new project on the primary library, displays the Quick Retrieve form, and adds a document the user selected to the folder.

```
Set Prj = App.PrimaryLibrary.CreateProject
If Not Prj Is Nothing Then
    If Prj.DisplayModal(True) Then
        QR.DisplayModal QR_SINGLE_SELECTION
        If QR.SelectedDocuments.Count > 0 Then
            ' Add the selected document to the folder
            Prj.AddDocument QR.SelectedDocuments(1)
        End If
    End If
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IQuickRetrieve" on page 556

"IProfile" on page 419

IQuickRetrieve::DocumentCount

The property returns the number of documents currently contained in the Quick Retrieve form.

C++ Syntax

```
HRESULT DocumentCount(  long* plCount);
```

VBScript Syntax

```
[ plCount = ] QuickRetrieve.DocumentCount
```

Parameters

plCount	[out, retval] Returns a long that represents the number of documents meeting object's search criteria.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access:

Read Only

Remarks

Use this property to find out the total number of documents in the Quick Retrieve form.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IQuickRetrieve" on page 556

IQuickRetrieve::Documents

The property returns a collection of document profiles in the current list.

C++ Syntax

```
HRESULT Documents(  IDocProfiles** ppDocs);
```

VBScript Syntax

```
[ Set ppDocs = ] QuickRetrieve.Documents
```

Parameters

ppDocs	[out, retval] Returns a DocProfiles object that represents the collection of document profiles.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access:

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IQuickRetrieve" on page 556

IQuickRetrieve::FormName

The property returns the name of the Quick Retrieve form.

C++ Syntax

```
HRESULT FormName(  BSTR* pbstrName);
```

VBScript Syntax

```
[ pbstrName = ] QuickRetrieve.FormName
```

Parameters

pbstrName	[out, retval] Returns a string with the form name.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access:

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IQuickRetrieve" on page 556

IQuickRetrieve::Search

The property returns or sets a Search object whose results the Quick Retrieve form will display.

C++ Syntax

```
HRESULT Search(  ISearch** ppSearch);

HRESULT Search(  ISearch* pSearch);
```

VBScript Syntax

```
[ Set ppSearch = ] QuickRetrieve.Search

Set QuickRetrieve.Search [ = pSearch ]
```

Parameters

ppSearch	[out, retval] A Search object that represents the Quick Retrieve form's search.
pSearch	[in] A Search object that represents the Quick Retrieve form's search.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access:

Read/Write

Remarks

This search is run either upon calling up the DisplayModal method or upon the attempt to retrieve the list of documents or the number of documents in the Quick Retrieve list. The property is set to the Recently Edited Documents search by default.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IQuickRetrieve" on page 556

IQuickRetrieve::SelectedDocuments

The property returns or sets a collection of highlighted or selected documents in the current list.

C++ Syntax

```
HRESULT SelectedDocuments(  IDocProfiles** ppDocs);
HRESULT SelectedDocuments(  IDocProfiles* pDocs);
```

VBScript Syntax

```
[ Set ppDocs = ] QuickRetrieve.SelectedDocuments
QuickRetrieve.SelectedDocuments = [ pDocs ]
```

Parameters

ppDocs	[out, retval] A DocProfiles object that represents the collection of selected documents.
pDocs	[in] A DocProfiles object that represents the collection of selected documents.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access:

Read/Write

Example

The following example creates a new project on the primary library, displays the Quick Retrieve form, and adds a document the user selected to the folder.

```
Dim App As New Application
Dim QR As New QuickRetrieve, Prj As Project

' Create a new project on the primary library
Set Prj = App.PrimaryLibrary.CreateProject
If Not Prj Is Nothing Then
If Prj.DisplayModal(True) Then
If Prj.Save(OF_NORMAL) Then
QR.DisplayModal QR_SINGLE_SELECTION
If QR.SelectedDocuments.Count > 0 Then
' Add the selected document to the folder
Prj.AddDocument QR.SelectedDocuments(1)
End If
End If
End If
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IQuickRetrieve" on page 556

IQuickRetrieveCallback

This interface extends functionality of a QuickRetrieve object. This object should be connected to a QuickRetrieve object using connection point before the IQuickRetrieve::DisplayModal method.

The interface inherits its definition from IUnknown.

Methods

IQuickRetrieveCallback Methods	Description
IQuickRetrieveCallback::OnOk	Called when a user click OK on the QuickRetrieve form.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "fsplugin.dll"

See Also

"IQuickRetrieve" on page 556

IQuickRetrieveCallback::OnOk

The method is called when a user clicks OK on the QuickRetrieve form.

C++ Syntax

```
HRESULT OnOk(  long hWnd,  IDocProfiles* pSelectedDocs,  long*
pResult);
```

VBScript Syntax

```
[ pResult  = ] QuickRetrieveCallback.OnOk (hWnd, pSelectedDocs)
```

Parameters

hWnd	[in] A window handle (HWND) of the currently displayed QuickRetrieve form.
pSelectedDocs	[in] Documents that are currently selected.
pResult	[out] For possible results see QRCReturnFlags Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <fsplugin.dll>

See Also

"IQuickRetrieveCallback" on page 566

"IQuickRetrieve" on page 556

ISearch

The Search object represents a DM Extensions quick search, an easy way to find particular documents by providing the information the user knows about those documents.

Search object criteria are specified in the search criteria form. The form depends on the primary group of the user working with the system.

When to Use

A Search object may be created with the CreateObject function or the Dim statement with the New keyword. To initialize the object the SetSearch method is to be called up before any other method or property.

The following example shows the Visual Basic syntax for creating a DOCSObjects.Search object:

```
Sub Form_Load()
```

```
...
Dim QS As Object
' Create a DOCSObjects.Search object
Set QS = CreateObject("DOCSObjects.Search")
QS.SetSearch "Test", "My Search"
...
End Sub
```

The Search object allow to set some properties via IBCObject2::SetProperty method.

Name	Type	Description
SearchLibraries	String	Libraries that should be used to search documents. Default search libraries will be ignored when this property set.
DisableSave	Boolean	User will not be able to save search into database. Ability to save the search into database will be available via API method ISearch::Save() only.
DisableEdit	Boolean	User will not be able to edit search (change criteria, change access, change search form, narrow the search). Ability to change the search will be available via API methods only.
DisableDelete	Boolean	User will not be able to delete search. Ability to save the search into database will be available via API method ISearch::Save() only.

The following example shows the Visual Basic syntax for using these properties:

```
Private Sub Form_Load()
    Dim pApp As New Application
    Dim pSearch As Search
    Dim pObj2 As IBCObject2

    Set pSearch = pApp.PrimaryLibrary.CreateSearch("DEF_QBE")
    Set pObj2 = pSearch
```

```
    pObj2.SetProperty "", "", "DisableSave", True
    pObj2.SetProperty "", "", "SearchLibraries",
"Library1;Librart2"
End Sub
```

The Search object inherits its definition from the IOMErrInfo interface so its implementation also includes all the properties of IOMErrInfo in addition to the ones described below.

Methods

ISearch Methods	Description
ISearch::Delete	The method deletes a search.
ISearch::Save	The method saves a new search or changes of an existing search.
ISearch::SetColumns	The method specifies document profile's columns to be retrieved by performing a search.
ISearch::SetCriteriaForm	The method sets the name of the search criteria form a search is assigned to.
ISearch::SetSearch	The method initializes a Search object to a particular quick search.
ISearch::ShowAccessForm	The method displays the search access control form.
ISearch::ShowEditCriteriaForm	The method displays the search criteria form.
ISearch::ShowSearchOptionsForm	The method displays the search form.

Properties

ISearch Properties	Description
ISearch::Canned	The property returns whether or not a search is the predetermined system quick search.
ISearch::Columns	The property returns a collection of all the columns that can be accessed through the search criteria form.
ISearch::Documents	The property returns a collection of document profiles meeting search criteria.
ISearch::DocumentsCount	The property returns the number of documents meeting search criteria.

ISearch Properties	Description
ISearch::FormName	The property returns the name of the search criteria form.
ISearch::Icon	The property returns the icon of a search.
ISearch::Iconized	Returns/sets whether a search is iconized on to desktop.
ISearch::Library	The property returns the Library object on which a search exists.
ISearch::Name	The property returns or sets the name or description of a search.
ISearch::SearchID	The property returns the identifier of a search.
ISearch::Trustees	The property returns a collection of users and groups having access to a search.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "fsplugin.dll"

See Also

"IOMErrInfo" on page 412

ISearch::Canned

The property returns whether or not a search is the predetermined system quick search.

C++ Syntax

```
HRESULT Canned(  VARIANT_BOOL* pbCanned);
```

VBScript Syntax

```
[ pbCanned = ] Search.Canned
```

Parameters

pbCanned	[out, retval] Returns a Boolean value that specifies whether or not a search is canned. True: A search is canned. False: A search is not canned.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access:

Read Only

Remarks

There are the following types of system quick searches: recently edited documents, template documents, checked-out documents, and restored documents. A search is canned if it is one of these quick searches.

The SearchID property of a system quick search is always negative.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ISearch" on page 567

ISearch::Columns

The property returns a collection of all the columns that can be accessed through the search criteria form.

C++ Syntax

```
HRESULT Columns(  IColumns** pColumns);
```

VBScript Syntax

```
[ Set pColumns = ] Search.Columns
```

Parameters

pColumns	[out, retval] Returns a Columns object that represents a collection of search columns.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access:

Read Only

Remarks

The Columns property provides a way to find out information and change values for fields that relate to the search criteria. Using the Columns property is the only way to create or “get to” a column object.

Example

The following example displays the name and value of each column of a search.

```
Dim App As New Application
Dim QS As Search
Dim Col As Column

Set QS = App.PrimaryLibrary.Searches("My Search")
If Not QS Is Nothing Then
For Each Col In QS.Columns
MsgBox "Column: " + Col.Name + Chr(13) + "Value: " + Col.Value
Next
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

“ISearch” on page 567

ISearch::Delete

The method deletes a search.

C++ Syntax

```
HRESULT Delete(  short sFlags,  VARIANT_BOOL* pbSuccess);
```

VBScript Syntax

```
[ pbSuccess  = ] Search.Delete (sFlags)
```

Parameters

sFlags	[in] Required. An integer value. Specifies how the method should be executed. The parameter can be a combination of the MethodsFlags values.
pbSuccess	[out, retval] Returns a Boolean value that specifies whether or not the operation succeeded. True: A search was successfully deleted. False: A search was not deleted.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

After deleting, a Search object becomes undefined, i.e. no properties or methods can be called on the object before initializing it again with the SetSearch method.

The user must have enough rights to be able to delete a search.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ISearch" on page 567

"ISearch::SetSearch" on page 585

"MethodsFlags Enumeration" on page 689

ISearch::Documents

The property returns a collection of document profiles meeting search criteria.

C++ Syntax

```
HRESULT Documents(  IDocProfiles** pDocs);
```

VBScript Syntax

```
[ Set pDocs = ] Search.Documents
```

Parameters

pDocs	[out, retval] Returns a DocProfiles object that represents a collection of document profiles.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access:

Read Only

Remarks

While retrieving a collection the search is being performed (if it has not been done before) and the resultant collection is returned to the client. Documents are retrieved only from searchable libraries, whose the SearchByDefault property is True.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ISearch" on page 567

ISearch::DocumentsCount

The property returns the number of documents meeting search criteria.

C++ Syntax

```
HRESULT DocumentsCount(  long* plCount);
```

VBScript Syntax

```
[ plCount = ] Search.DocumentsCount
```

Parameters

plCount	[out, retval] Returns a long that represents the number of documents meeting search criteria.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access:

Read Only

Remarks

While retrieving the property's value the search is always being performed anew. Documents are retrieved only from searchable libraries, whose the SearchByDefault property is True.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ISearch" on page 567

ISearch::FormName

The property returns the name of the search criteria form.

C++ Syntax

```
HRESULT FormName(  BSTR* pbstrName);
```

VBScript Syntax

```
[ pbstrName = ] Search.FormName
```

Parameters

pbstrName	[out, retval] Returns a string with the form name.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access:

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ISearch" on page 567

ISearch::Icon

The property returns the icon of a search.

C++ Syntax

```
HRESULT Icon(  long* plIcon);
```

VBScript Syntax

```
[ plIcon = ] Search.Icon
```

Parameters

pIIcon	[out, retval] Returns a long that specifies the HICON value of the search icon.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access:

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ISearch" on page 567

ISearch::Iconized

Returns/sets whether a search is iconized on the desktop.

C++ Syntax

```
HRESULT Iconized(  VARIANT_BOOL* pbIconized);
HRESULT Iconized(  VARIANT_BOOL bIconized);
```

Visual Basic Syntax

```
[ pbIconized = ] Search.Iconized
Search.Iconized [ = bIconized ]
```

Parameters

pbIconized	[out, retval] Indicates whether a search is iconized on the desktop.
bIconized	[in] Indicates whether a search is iconized on the desktop.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access:

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ISearch" on page 567

ISearch::Library

The property returns the Library object on which a search exists.

C++ Syntax

```
HRESULT Library(  ILibrary** ppLibrary);
```

VBScript Syntax

```
[ Set ppLibrary = ] Search.Library
```

Parameters

ppLibrary	[out, retval] Returns a Library object that specifies the library a search is assigned to.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access:

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ISearch" on page 567

ISearch::Name

The property returns or sets the name or description of a search.

C++ Syntax

```
HRESULT Name(  BSTR* pbstrName);
HRESULT Name(  BSTR bstrName);
```

VBScript Syntax

```
[ pbstrName = ] Search.Name
Search.Name [ = bstrName ]
```

Parameters

pbstrName	[out, retval] A string that specifies the search name.
bstrName	[in] A string that specifies the search name.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access:

Read/Write

Example

The following example creates a new search on the primary library.

```
Dim App As New Application
Dim QS As Search

' Create a new search on the primary library
Set QS = App.PrimaryLibrary.CreateSearch("DEF_QBE")
If Not QS Is Nothing Then
QS.ShowEditCriteriaForm True
QS.Name = "My Search"
QS.Save OF_NORMAL
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ISearch" on page 567

ISearch::Save

The method saves a new search or changes of an existing search.

C++ Syntax

```
HRESULT Save(  short sFlags,  VARIANT_BOOL* pbSuccess);
```

VBScript Syntax

```
[ pbSuccess  = ] Search.Save (sFlags)
```

Parameters

sFlags	[in] Required. An integer value. Specifies how the method should be executed. The parameter can be a combination of the MethodsFlags values.
pbSuccess	[out, retval] Returns a Boolean value that specifies whether or not the operation succeeded. True: A search was successfully saved. False: A search was not saved.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

Only the name property changes are saved. Changes made to the search criteria are saved immediately.

Example

The following example creates a new search on the primary library.

```
Dim App As New Application
Dim QS As Search

' Create a new search on the primary library
Set QS = App.PrimaryLibrary.CreateSearch("DEF_QBE")
If Not QS Is Nothing Then
QS.ShowEditCriteriaForm True
QS.Name = "My Search"
QS.Save OF_NORMAL
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ISearch" on page 567

"ISearch::SetSearch" on page 585

"MethodsFlags Enumeration" on page 689

ISearch::SearchID

The property returns the identifier of a search.

C++ Syntax

```
HRESULT SearchID(  long* plID);
```

VBScript Syntax

```
[ plID = ] Search.SearchID
```

Parameters

pllD	[out, retval] Returns a long that uniquely identifies a search. The QuickSearchesID values will be returned for new and canned quick searches.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access:

Read Only

Remarks

The search identifier will be returned if it is an existing folder. The search identifier is unique within a library, i.e. two searches on the same library have different identifiers.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ISearch" on page 567

ISearch::SetColumns

The method specifies document profile's columns to be retrieved by performing a search.

C++ Syntax

```
HRESULT SetColumns(  BSTR bstrColumns);
```

VBScript Syntax

```
Search.SetColumns (bstrColumns)
```

Parameters

bstrColumns	[in] Required. A string that specifies names of all document columns being retrieved separated by a semicolon.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ISearch" on page 567

ISearch::SetCriteriaForm

The method sets the name of the search criteria form a search is assigned to.

C++ Syntax

```
HRESULT SetCriteriaForm(  BSTR bstrName);
```

VBScript Syntax

```
Search.SetCriteriaForm (bstrName)
```

Parameters

bstrName	[in] Required. A string that specifies the name of the search criteria form.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

When assigning a new form that does not coincide with the current one, a new column collection is built, which replaces the previous one. All criteria of the old form are copied into the new one (if possible).

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ISearch" on page 567

ISearch::SetSearch

The method initializes a Search object to a particular quick search.

C++ Syntax

```
HRESULT SetSearch(  VARIANT vLibrary,  VARIANT vSearch);
```

VBScript Syntax

```
Search.SetSearch (vLibrary, vSearch)
```

Parameters

vLibrary	[in] Required. An expression that specifies a library. If a string expression, vLibrary should specify the library name. If an object, vLibrary should specify a valid Library object.
vSearch	[in] Required. An expression that specifies a quick search on vLibrary. If a numeric expression, vSearch should specify the identifier of a search. If a string expression, vSearch should specify the search name.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

If the search was created with the CreateObject function or the Dim statement with the New keyword, the method is to be called up before any other method or property.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ISearch" on page 567

ISearch::ShowAccessForm

The method displays the search's access control form.

C++ Syntax

```
HRESULT ShowAccessForm(  VARIANT_BOOL bAllowEdits,  short*
psResult);
```

VBScript Syntax

```
[ psResult  = ] Search.ShowAccessForm (bAllowEdits)
```

Parameters

bAllowEdits	[in] Required. A Boolean value that specifies whether or not the search's trustee list can be changed. True: The trustee list can be changed. False: The trustee list cannot be changed.
psResult	[out, retval] Returns an integer that specifies user's action on the form. For possible values see FormsResults Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The search's access control form can be customized and should support the ISearchAccessForm interface. To display the form, the method calls up the ShowSearchAccessForm method on ISearchAccessForm.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ISearch" on page 567

ISearch::ShowEditCriteriaForm

The method displays the search criteria form.

C++ Syntax

```
HRESULT ShowEditCriteriaForm(  VARIANT_BOOL bAllowEdits,  short*
psResult);
```

VBScript Syntax

```
[ psResult  = ] Search.ShowEditCriteriaForm (bAllowEdits)
```

Parameters

bAllowEdits	[in] Required. A Boolean value that specifies whether or not the search criteria can be changed. True: The search criteria can be changed. False: The search criteria cannot be changed.
psResult	[out, retval] Returns an integer that specifies user's action on the form. For possible values see FormsResults Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The search criteria form can be customized and should support the ISearchCriteriaForm interface. To display the form, the method calls up the ShowSearchCriteriaForm method on ISearchCriteriaForm.

Example

The following example creates a new search on the primary library, retrieves documents meeting the search criteria, and displays the name of the first document.

```
Dim App As New Application
Dim QS As Search, Prof As Profile

' Create a new search on the primary library
Set QS = App.PrimaryLibrary.CreateSearch("DEF_QBE")
If Not QS Is Nothing Then
QS.ShowEditCriteriaForm True
For Each Prof In QS.Documents
MsgBox QS.Documents(1).Columns("DOCNAME").Value
Next
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ISearch" on page 567

ISearch::ShowSearchOptionsForm

The method displays the search form.

C++ Syntax

```
HRESULT ShowSearchOptionsForm(  VARIANT_BOOL bAllowEdits,
VARIANT_BOOL* pbResult);
```

VBScript Syntax

```
[ pbResult  = ] Search.ShowSearchOptionsForm (bAllowEdits)
```

Parameters

bAllowEdits	[in] Required. A Boolean value that specifies whether or not search properties can be changed. True: Search properties can be changed. False: Search properties cannot be changed.
pbResult	[out, retval] Returns an integer that specifies user's action on the form. For possible values see FormsResults Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

If changes are made to the search when displaying its form and the user clicks OK, the program does not need to call the Save method because the changes are saved automatically.

The search form can be customized and should support the ISearchOptionsForm interface. To display the form, the method calls up the ShowSearchOptionsForm method on ISearchOptionsForm.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ISearch" on page 567

ISearch::Trustees

The property returns a collection of users and groups having access to a search.

C++ Syntax

```
HRESULT Trustees(  ITrusteeCollection** ppTrustees);
```

VBScript Syntax

```
[ Set ppTrustees = ] Search.Trustees
```

Parameters

ppTrustees	[out, retval] Returns a TrusteeCollection object that represents a collection of search's trustees.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access:

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ISearch" on page 567

ISearches

The Searches object represents a collection of all searches from a library including both saved and new searches. The collection may be retrieved with the Searches property on a library.

Properties

ISearches Properties	Description
Count	The property returns the number of items in a collection.
Item	The property returns the specific member of a collection either by position or by key.
_NewEnum	Returns an object that supports IEnumVARIANT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "fsplugin.dll"

See Also

"ISearch" on page 567

ISearches::Count

The property returns the number of items in a collection.

C++ Syntax

```
HRESULT Count(  long* plCount);
```

VBScript Syntax

```
[ plCount = ] Searches.Count
```

Parameters

plCount	[out, retval] Returns a long that specifies the number of items in a collection.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Access:

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ISearch" on page 567

"ISearches" on page 591

ISearches::Item

The property returns the specific member of a collection either by position or by key.

C++ Syntax

```
HRESULT Item(  VARIANT vIndex,  ISearch** ppSearch);
```

VBScript Syntax

```
[ Set ppSearch = ] Searches.Item (vIndex)
```

Parameters

vIndex	[in] Required. An expression that specifies the member position in a collection. If a numeric expression, vIndex should be a number from 1 to the value of the collection's Count property. If a string expression, vIndex should correspond to a search name.
ppSearch	[out, retval] Returns a Search object that specifies the requested item.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_INAVLIDARG	An invalid parameter has been passed.
E_UNEXPECTED	Unexpected error occurs.

Access:

Read Only

Remarks

If the value provided as vIndex does not match any existing member of a collection, an error occurs.

Item is the default property for a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ISearch" on page 567

"ISearches" on page 591

ISearches::_NewEnum

The _NewEnum property returns the IEnumVARIANT interface on an object that can be used to enumerate the collection.

C++ Syntax

```
HRESULT _NewEnum(LPUNKNOWN* ppEnum);
```

Parameters

ppEnum	[out, retval] An address of the IUnknown* pointer variable that receives the interface pointer to the enumeration object. Each time the collection receives a successful call to _NewEnum, it should increase the reference count on the *ppEnum pointer the property returns. It is the caller's responsibility to call IUnknown::Release when it is done with the pointer. If an error occurs, the implementation should set *ppEnum to NULL

Return Values

S_OK	The operation succeeded.
E_OUTOFMEMORY	There is not enough memory to complete the operation.
E_INVALIDARG	ppEnum is not a valid pointer.
E_UNEXPECTED	An unexpected error occurred.

Remarks

It is the standard property of a collection. It returns an enumeration object, which implements the IEnumVariant interface. This property should have the DISPID_NEWENUM identifier. The user can use the enumeration interface to step through the objects in the collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ISearches" on page 591

ISearchTrustee

The SearchTrustee object represents a user or group who has access to a search. The SearchTrustee object may be retrieved through the Trustees collection on a search.

Object's properties can be modified only if the user has edit rights to the search. When the user who does not have enough rights attempts to make changes in the access rights, an error will occur.

Properties

ISearchTrustee Properties	Description
CanDeleteSearch	The property returns or sets whether or not a trustee can delete a specific search.
CanEditSearch	The property returns or sets whether or not a trustee can edit a specific search.
CanViewSearch	The property returns or sets whether or not a trustee can see a specific search and launch it.
IsGroup	The property returns whether or not a trustee is a group.
Name	The property returns the DM trustee's username or group name.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "fsplugin.dll"

See Also

"ISearch" on page 567

ISearchTrustee::CanDeleteSearch

The property returns or sets whether or not the trustee can delete a specific search.

C++ Syntax

```
HRESULT CanDeleteSearch(  VARIANT_BOOL* pbDelete);
HRESULT CanDeleteSearch(  VARIANT_BOOL bDelete);
```

VBScript Syntax

```
[ pbDelete = ] SearchTrustee.CanDeleteSearch
SearchTrustee.CanDeleteSearch [ = bDelete ]
```

Parameters

pbDelete	[out, retval] A Boolean value that specifies whether or not the trustee has rights to delete a search.
bDelete	[in] A Boolean value that specifies whether or not the trustee has rights to delete a search. True: The trustee has rights to delete the search. False: The trustee does not have rights to delete the search

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Access:

Read/Write

Remarks

A trustee cannot have delete rights without also having edit and view rights.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ISearchTrustee" on page 595

ISearchTrustee::CanEditSearch

The property returns or sets whether or not the trustee can edit a specific search.

C++ Syntax

```
HRESULT CanEditSearch(  VARIANT_BOOL* pbEdit);
HRESULT CanEditSearch(  VARIANT_BOOL bEdit);
```

VBScript Syntax

```
[ pbEdit = ] SearchTrustee.CanEditSearch
SearchTrustee.CanEditSearch [ = pbEdit ]
```

Parameters

pbEdit	[out, retval] A Boolean value that specifies whether or not the trustee has rights to edit a search.
bEdit	[in] A Boolean value that specifies whether or not the trustee has rights to edit a search. True: The trustee has rights to edit the search. False: The trustee has not rights to edit the search.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Access:

Read/Write

Remarks

If the trustee has been granted edit rights on a search he/she can change the name property of the search as well as change the search criteria and form and modify search's trustee list. A user cannot have edit rights without also having view rights.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ISearchTrustee" on page 595

ISearchTrustee::CanViewSearch

The property returns or sets whether or not the trustee can see a specific search and launch it.

C++ Syntax

```
HRESULT CanViewSearch(  VARIANT_BOOL* pbView);
HRESULT CanViewSearch(  VARIANT_BOOL bView);
```

VBScript Syntax

```
[ pbView = ] SearchTrustee.CanViewSearch
SearchTrustee.CanViewSearch [ = bView ]
```

Parameters

pbView	[out, retval] A Boolean value that specifies whether or not the trustee has rights to view a search and its results.
bView	[in] A Boolean value that specifies whether or not the trustee has rights to view a search and its results. True: The trustee has rights to view the search and its results. False: The trustee has not rights to view the search and its results.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Access:

Read/Write

Remarks

If the trustee has rights to view a search, he/she will still not be able to retrieve, view, or change any document from the search result list to which he/she has not been granted access. If the library uses the profile level security, those documents that the trustee has been restricted from seeing will still be hidden from him. If the trustee does not have view access then he/she cannot also have delete or edit rights to a search.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ISearchTrustee" on page 595

ISearchTrustee::IsGroup

The property returns whether or not the trustee is a group.

C++ Syntax

```
HRESULT IsGroup(  VARIANT_BOOL* pbGroup);
```

VBScript Syntax

```
[ pbGroup = ] SearchTrustee.IsGroup
```

Parameters

pbGroup	[out, retval] Returns a Boolean value that specifies whether or not the trustee is a group. True: The trustee is a group. False: The trustee is not a group.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Access:

Read Only

Remarks

The IsGroup property is useful for determining which Trustees methods to use with a Trustee.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ISearchTrustee" on page 595

ISearchTrustee::Name

The property returns the DM trustee's username or group name.

C++ Syntax

```
HRESULT Name(  BSTR* pbstrName);
```

VBScript Syntax

```
[ pbstrName = ] SearchTrustee.Name
```

Parameters

pbstrName	[out, retval] Returns a string that specifies the trustee's name.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Access:

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ISearchTrustee" on page 595

ISearchTrustee2

The ISearchTrustee2 interface extends SearchTrustee object functionality.

Object's properties can be modified only if the user has edit access to the search. When the user who doesn't have enough rights attempts to make changes in the access rights, an error will occur.

Properties

ISearchTrustee2 Properties	Description
AccessDeleteSearch	The property returns or sets whether or not a trustee can delete a specific search.
AccessEditSearch	The property returns or sets whether or not a trustee can edit a specific search.
AccessViewSearch	The property returns or sets whether or not a trustee can see a specific search and launch it.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import "fsplugin.dll"

ISearchTrustee2::AccessDeleteSearch

The property returns or sets whether or not a trustee can delete a specific search.

C++ Syntax

```
HRESULT AccessDeleteSearch(  short* psDelete);

HRESULT AccessDeleteSearch(  short sDelete);
```

VBScript Syntax

```
[ psDelete = ] SearchTrustee2.AccessDeleteSearch

SearchTrustee2.AccessDeleteSearch [ = sDelete ]
```

Parameters

psDelete	[out, retval] An AccessType value that specifies whether or not the trustee has rights to delete the search.
sDelete	[in] An AccessType value that specifies whether or not the trustee has rights to delete the search.

Return Values

S_OK	The operation succeeded.
E_POINTER	Parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Access:

Read/Write

Remarks

A trustee can't have delete access without also having edit and view access.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"ISearchTrustee2" on page 601

ISearchTrustee2::AccessEditSearch

The property returns or sets whether or not a trustee can edit a specific search.

C++ Syntax

```
HRESULT AccessEditSearch(  short* psEdit);
HRESULT AccessEditSearch(  short sEdit);
```

VBScript Syntax

```
[ psEdit = ] SearchTrustee2.AccessEditSearch
SearchTrustee2.AccessEditSearch [ = sEdit ]
```

Parameters

psEdit	[out, retval] An AccessType value that specifies whether or not the trustee has rights to edit the search.
sEdit	[in] An AccessType value that specifies whether or not the trustee has rights to edit the search.

Return Values

S_OK	The operation succeeded.
E_POINTER	Parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Access:

Read/Write

Remarks

If a trustee has been granted edit rights on a search he/she can change the name property of the search as well as change the search criteria and form and modify search's trustee list. A user can't have edit access without also having view access.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"ISearchTrustee2" on page 601

ISearchTrustee2::AccessViewSearch

The property returns or sets whether or not a trustee can see a specific search and launch it.

C++ Syntax

```
HRESULT AccessViewSearch(  short* psView);
HRESULT AccessViewSearch(  short sView);
```

VBScript Syntax

```
[ psView = ] SearchTrustee2.AccessViewSearch
```

```
SearchTrustee2.AccessViewSearch [ = sView ]
```

Parameters

psView	[out, retval] An AccessType value that specifies whether or not the trustee has rights to view the search and its results.
sView	[in] An AccessType value that specifies whether or not the trustee has rights to view the search and its results.

Return Values

S_OK	The operation succeeded.
E_POINTER	Parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Access:

Read/Write

Remarks

If a trustee is given access to view a search, he/she will still not be able to retrieve, view, or change any document from search result list to which he/she hasn't been granted access. If the library is using profile level security, those documents that the trustee has been restricted from seeing will still be hidden from him. If a trustee doesn't have view access then he/she can't also have delete or edit access to the search.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"ISearchTrustee2" on page 601

ITerminology

The ITerminology interface is used to substitute replacement names for terms used in in strings, windows, and menus in DM Extensions. Terms are values from the TERM column in the SQL database

TERMINOLOGY table. The replacement name is the string value in the REPLACE_VALUE column for the specified TERM string. Terminology information is extracted from the login library.

Properties

ITerminology Methods	Description
ITerminology::FormatAllWindowControls	Replaces the specified terms in all of the window controls.
ITerminology::FormatMenu	Replaces the specified terms in all of the menu items.
ITerminology::FormatTerms	Replaces the specified terms with their corresponding values from the REPLACE_VALUE column of the TERMINOLOGY table.
ITerminology::FormatWindowText	Replaces the terms in the text of the specified window.

Remarks

To avoid delays when replacement values are substituted for their associated terms, the entire TERMINOLOGY table is read into system memory at the start of each user session. As a result, changes to text values in the TERMINOLOGY table will not be reflected in a user session that is in progress when the changes are made. Terminology values for a given user session are those that were current when the session first started.

Requirements

DM: Requires DM Extensions 5.1 or later.

Header: #import "fsplugin.dll"

ITerminology::FormatAllWindowControls

This method substitutes data from the REPLACE_VALUE column of the TERMINOLOGY table for each of the terms in each of the window controls in the current window.

C++ Syntax

```
HRESULT FormatAllWindowControls(  long lWndHandle);
```

VBScript Syntax

```
ITerminology.FormatAllWindowControls( lWndHandle )
```

Parameters

lWndHandle	[in] A long integer that identifies the window that is to have all controls, as shown in the TERM column of the TERMINOLOGY table, replaced with the associated REPLACE_VALUE term in the TERMINOLOGY table.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.1 or later.

Header: #import <fsplugin.dll>

See Also

"ITerminology" on page 604

ITerminology::FormatMenu

This method substitutes data from the REPLACE_VALUE column of the TERMINOLOGY table for each of the terms in the specified menu.

C++ Syntax

```
HRESULT FormatMenu(  long lMenuHandle);
```

VBScript Syntax

```
ITerminology.FormatMenu( lMenuHandle )
```

Parameters

lMenuHandle	[in] A long integer that identifies the menu that is to have all terms, as shown in the TERM column of the TERMINOLOGY table, replaced with the associated REPLACE_VALUE term in the TERMINOLOGY table.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.1 or later.

Header: #import <fsplugin.dll>

See Also

"ITerminology" on page 604

ITerminology::FormatTerms

This method replaces the specified terms with their corresponding values from the REPLACE_VALUE column of the TERMINOLOGY table. The input terms must be enclosed within square brackets and percent signs in order for the FormatTerms method to recognize them. For example, if "MyTerm" was the input value, the following string would need to be passed to FormatTerms:

C++ Syntax

```
HRESULT FormatTerms(  BSTR bstrFormat,  BSTR bstrProduct);
```

VBScript Syntax

```
[ bstrProduct = ] Terminology.FormatTerms( bstrFormat  )
```

Parameters

bstrFormat	[in] A properly formatted string that contains one or more terms to be replaced. A properly formatted input string has terms contained within square brackets and percent signs, as shown above.
bstrProduct	[out, retval] The string with the replacement term(s) associated with the input string.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

For each properly formatted term in the input string, it's replacement value will be substituted in the output string. This input string contains two properly formatted input values: "The %[object]% is %[color]%." The output string for this example might become: "The house is white."

Requirements

DM: Requires DM Extensions 5.1 or later.

Header: #import <fsplugin.dll>

See Also

"ITerminology" on page 604

ITerminology::FormatWindowText

This method substitutes data from the REPLACE_VALUE column of the TERMINOLOGY table for each of the terms in the specified window.

C++ Syntax

```
HRESULT FormatWindowText(  long lWndHandle);
```

VBScript Syntax

```
ITerminology.FormatWindowText( lWndHandle )
```

Parameters

lWndHandle	[in] A long integer that identifies the window that is to have all terms, as shown in the TERM column of the TERMINOLOGY table, replaced with the associated REPLACE_VALUE term in the TERMINOLOGY table.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.1 or later.

Header: #import <fsplugin.dll>

See Also

"ITerminology" on page 604

ITrusteeCollection

The TrusteeCollection object represents a collection of users and groups who have access to a particular object, such asProfile, Project, orSearch. The collection may be retrieved with the Trustees property on that object.

Methods

ITrusteeCollection Methods	Description
ITrusteeCollection::AddGroup	The method adds a new group to object's trustees.
ITrusteeCollection::AddPerson	The method adds a new user to object's trustees.
ITrusteeCollection::BeginEdit	The method begins a transaction for modifying the object's trustee list.
ITrusteeCollection::FinishEdit	The method ends a transaction for modifying the object's trustee list.

ITrusteeCollection Methods	Description
ITrusteeCollection::GetEffectiveRights method	The method returns the user's or group access rights, including access rights of the groups the user belongs to, to an object by the user's or group name.
ITrusteeCollection::RemoveGroup	The method removes a group from the object's trustee list.
ITrusteeCollection::RemovePerson	The method removes a user from the object's trustee list.

Properties

ITrusteeCollection Properties	Description
ITrusteeCollection::Count	The property returns the number of items in a collection.
ITrusteeCollection::Item	The property returns the specific member of a collection either by position or by key.
ITrusteeCollection::_NewEnum	Returns an object that supports IEnumVARIANT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "fsplugin.dll"

See Also

"IProject" on page 497

ITrusteeCollection::AddGroup

The method adds a new group to object's trustees.

C++ Syntax

```
HRESULT AddGroup(  VARIANT vGroupID,  long lRights);
```

VBScript Syntax

```
TrusteeCollection.AddGroup (vGroupID, lRights)
```

Parameters

vGroupID	[in] Required. An expression that specifies the group being assigned rights. If a string expression, vGroupID should specify a group name. If an object expression, vGroupID should specify a valid group object of the type ProfileTrustee, ProjectTrustee, or SearchTrustee.
lRights	[in] Required. A long that specifies the numeric value of rights to be assigned. The rights to use for a trustee depend on whether the object is a document, project or quick search. For rights and their values see ProfileRights Enumeration, ProjectRights Enumeration and QuickSearchRights Enumeration.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_INAVLIDARG	An invalid parameter has been passed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

After modifying the document's trustee list, the profile should be saved with the Save method.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ITrusteeCollection" on page 609

"IProfile" on page 419

ITrusteeCollection::AddPerson

The method adds a new user to object's trustees.

C++ Syntax

```
HRESULT AddPerson(  VARIANT vPersonID,  long lRights);
```

VBScript Syntax

```
TrusteeCollection.AddPerson (vPersonID, lRights)
```

Parameters

vPersonID	[in] Required. An expression that specifies the user being assigned rights. If a string expression, vPersonID should specify a user's name. If an object expression, vPersonID should specify a valid user object of the type ProfileTrustee, ProjectTrustee, or SearchTrustee.
lRights	[in] Required. A long that specifies the numeric value of rights to be assigned. The rights to use for a trustee depend on whether the object is a document, folder or quick search. For rights and their values see ProfileRights Enumeration, ProjectRights Enumeration and QuickSearchRights Enumeration. To calculate a composite rights value, just add the appropriate individual rights together.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_INAVLIDARG	An invalid parameter has been passed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

After modifying the document's trustee list, the profile should be saved with the Save method.

Example

The following example adds the user "Monica" to the trustee list of the document with number 20 on the primary library.

```
Dim App As New Application
Dim Prof As Profile
Set Prof = App.PrimaryLibrary.GetProfile(20)
```

```
' Check whether the document is secured
If Not Prof Is Nothing And _
Prof.Columns("$DEFAULT_RIGHTS").Value <> "0" Then
Prof.SelectForEdit OF_NORMAL + SFE_LOCK_PROFILE
Prof.Trustees.AddPerson "Monica", _
RIGHTS_PROFILE_VIEW + RIGHTS_PROFILE_EDIT
Prof.Save OF_NORMAL
Prof.ReleaseDoc
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ITrusteeCollection" on page 609

"IProfile" on page 419

ITrusteeCollection::BeginEdit

The method begins the transaction for modifying the object's trustee list.

C++ Syntax

```
HRESULT BeginEdit();
```

VBScript Syntax

```
TrusteeCollection.BeginEdit
```

Return Values

S_OK	The operation succeeded.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Transactions make it possible to commit or rollback all changes to the object's trustee list made within a transaction block between BeginEdit and FinishEdit. A transaction will be rolled back if its block does not end with the FinishEdit method.

Example

The following example adds the user “Monica” to the trustee list of the document with number 20 on the primary library.

```
Dim App As New Application
Dim Prof As Profile, Trustees As TrusteeCollection
Set Prof = App.PrimaryLibrary.GetProfile(20)
' Check whether the document is secured
If Not Prof Is Nothing And _
Prof.Columns("$DEFAULT_RIGHTS").Value <> "0" Then
Prof.SelectForEdit OF_NORMAL + SFE_LOCK_PROFILE
Set Trustees = Prof.Trustees
Trustees.BeginEdit
Trustees.AddPerson "Monica", _
RIGHTS_PROFILE_VIEW + RIGHTS_PROFILE_EDIT
Trustees.FinishEdit True
Prof.Save OF_NORMAL
Prof.ReleaseDoc
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

“ITrusteeCollection” on page 609

“IProfile” on page 419

ITrusteeCollection::Count

The property returns the number of items in a collection.

C++ Syntax

```
HRESULT Count(  long* plCount);
```

VBScript Syntax

```
[ plCount = ] TrusteeCollection.Count
```

Parameters

plCount	[out, retval] Returns a long that specifies the number of items in a collection.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Access:

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ITrusteeCollection" on page 609

ITrusteeCollection::FinishEdit

The method ends the transaction for modifying the object's trustee list.

C++ Syntax

```
HRESULT FinishEdit(  VARIANT_BOOL bSave);
```

VBScript Syntax

```
TrusteeCollection.FinishEdit (bSave)
```

Parameters

bSave	[in] Required. A Boolean value that specifies whether or not the transaction should be committed. True: Commits the transaction. All changes made to the trustee list will be saved. False: Rollbacks the transaction. All changes made to the trustee list will be undone.

Return Values

S_OK	The operation succeeded.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Transactions make it possible to commit or rollback all changes to the object's trustee list made within a transaction block between BeginEdit and FinishEdit. A transaction will be rolled back if its block does not end with the FinishEdit method.

Example

The following example adds the user "Monica" to the trustee list of the document with number 20 on the primary library.

```
Dim App As New Application
Dim Prof As Profile, Trustees As TrusteeCollection
Set Prof = App.PrimaryLibrary.GetProfile(20)
' Check whether the document is secured
If Not Prof Is Nothing And _
Prof.Columns("$DEFAULT_RIGHTS").Value <> "0" Then
Prof.SelectForEdit OF_NORMAL + SFE_LOCK_PROFILE
Set Trustees = Prof.Trustees
Trustees.BeginEdit
Trustees.AddPerson "Monica", _
RIGHTS_PROFILE_VIEW + RIGHTS_PROFILE_EDIT
Trustees.FinishEdit True
Prof.Save OF_NORMAL
Prof.ReleaseDoc
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ITrusteeCollection" on page 609

"IProfile" on page 419

ITrusteeCollection::GetEffectiveRights method

The method returns the user or group access rights, including access rights of the groups the user belongs to, to an object by the user's or group's name.

C++ Syntax

```
HRESULT GetEffectiveRights(  BSTR bstrUserName,  LPDISPATCH*
ppTrustee);
```

VBScript Syntax

```
[ ppTrustee = ] TrusteeCollection.GetEffectiveRights
(bstrUserName)
```

Parameters

bstrUserName	[in] Required. A string with the user's or group's name.
ppTrustee	[out, retval] Returns an object of the type ProfileTrustee, ProjectTrustee, or SearchTrustee that specifies the user or group access rights.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_INAVLIDARG	An invalid parameter has been passed.
E_UNEXPECTED	Unexpected error occurs.

Example

The following example checks whether or not the user "Monica" has control access rights to the document with number 20 on the primary library.

```
Dim App As New Application
Dim Prof As Profile, ProfTrustee As ProfileTrustee
Set Prof = App.PrimaryLibrary.GetProfile(20)
' Check whether the document is secured
If Not Prof Is Nothing And _
Prof.Columns("$DEFAULT_RIGHTS").Value <> "0" Then
Set ProfTrustee = Prof.Trustees.GetEffectiveRights("Monica")
If ProfTrustee.CanControlAccess Then
MsgBox "Monica has control access rights."
Else
MsgBox "Monica doesn't have control access rights."
End If
End If
```

Requirements

DM: Requires PowerDOCS 3.5.1 (for example) or later.

Header: #import <fsplugin.dll>

See Also

"ITrusteeCollection" on page 609

"IProfile" on page 419

ITrusteeCollection::Item

The property returns the specific member of a collection either by position or by key.

C++ Syntax

```
HRESULT Item(  VARIANT vIndex,  LPDISPATCH* ppTrustee);
```

VBScript Syntax

```
[ ppTrustee = ] TrusteeCollection.Item (vIndex)
```

Parameters

vIndex	[in] Required. An expression that specifies the member position in a collection. If a numeric expression, vIndex should be a number from 1 to the value of the collection's Count property. If a string expression, vIndex should correspond to the key argument specified when the member referred to was added to a collection.
ppTrustee	[out, retval] Returns an object that specifies the requested item. It may be a ProfileTrustee, ProjectTrustee, or SearchTrustee object.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_INAVLIDARG	An invalid parameter has been passed.
E_UNEXPECTED	Unexpected error occurs.

Access:

Read Only

Remarks

If the value provided as vIndex does not match any existing member of a collection, an error occurs.

Item is the default property for a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ITrusteeCollection" on page 609

ITrusteeCollection::RemoveGroup

The method removes a group from the object's trustee list.

C++ Syntax

```
HRESULT RemoveGroup(  VARIANT vGroupID);
```

VBScript Syntax

```
TrusteeCollection.RemoveGroup (vGroupID)
```

Parameters

vGroupID	[in] Required. An expression that specifies a group being removed from the trustee list. If a string expression, vGroupID should specify a group name. If an object expression, vGroupID should specify a valid group object of the type ProfileTrustee, ProjectTrustee, or SearchTrustee.

Return Values

S_OK	The operation succeeded.
E_INAVLIDARG	An invalid parameter has been passed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

After modifying the document's trustee list, the profile should be saved with the Save method.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ITrusteeCollection" on page 609

"IProfile" on page 419

ITrusteeCollection::RemovePerson

The method removes a user from the object's trustee list.

C++ Syntax

```
HRESULT RemovePerson(  VARIANT vPersonID);
```

VBScript Syntax

```
TrusteeCollection.RemovePerson (vPersonID)
```

Parameters

vPersonID	[in] Required. An expression that specifies a user being assigned rights. If a string expression, vPersonID should specify a user's name. If an object expression, vPersonID should specify a valid user object of the type ProfileTrustee, ProjectTrustee, or SearchTrustee.

Return Values

S_OK	The operation succeeded.
E_INAVLIDARG	An invalid parameter has been passed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

After modifying the document's trustee list, the profile should be saved with the Save method.

Example

The following example removes the user "Monica" from the trustee list of the document with number 20 on the primary library.

```
Dim App As New Application
Dim Prof As Profile
Set Prof = App.PrimaryLibrary.GetProfile(20)
' Check whether the document is secured
If Not Prof Is Nothing And _
Prof.Columns("$DEFAULT_RIGHTS").Value <> "0" Then
Prof.SelectForEdit OF_NORMAL + SFE_LOCK_PROFILE
Prof.Trustees.RemovePerson "Monica"
Prof.Save OF_NORMAL
Prof.ReleaseDoc
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ITrusteeCollection" on page 609

"IProfile" on page 419

ITrusteeCollection::_NewEnum

The _NewEnum property returns the IEnumVARIANT interface on an object that can be used to enumerate the collection.

C++ Syntax

```
HRESULT _NewEnum(LPUNKNOWN* ppEnum);
```

Parameters

ppEnum	[out, retval] An address of the IUnknown* pointer variable that receives the interface pointer to the enumeration object. Each time the collection receives a successful call to _NewEnum, it should increase the reference count on the *ppEnum pointer the property returns. It is the caller's responsibility to call IUnknown::Release when it is done with the pointer. If an error occurs, the implementation should set *ppEnum to NULL

Return Values

S_OK	The operation succeeded.
E_OUTOFMEMORY	There is not enough memory to complete the operation.
E_INVALIDARG	ppEnum is not a valid pointer.
E_UNEXPECTED	Unexpected error occurred.

Remarks

It is the standard property of a collection. It returns an enumeration object, which implements the IEnumVariant interface. This property should have the DISPID_NEWENUM identifier. The user can use the enumeration interface to step through the objects in the collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ITrusteeCollection" on page 609

IVersion

The Version object represents a DM document's version, subversion, or attachment.

The Version object inherits its definition from the IOMErrInfo interface so its implementation also includes all the properties of IOMErrInfo in addition to ones described below.

Methods

IVersion Methods	Description
IVersion::Open	The method opens a version in the associated application in the edit mode.
IVersion::Preview	The method views the contents of a document version.
IVersion::PrintVersion	The method prints a document version.
IVersion::Export	The method exports the contents of a version to a file.
IVersion::OpenInCompatibleApp	The method opens a document version in an application compatible with its native application.
IVersion::DownloadFile	The method downloads the contents of a document version to the file specified by the FilePath property of the object.
IVersion::UploadFile	The method uploads the file specified by the FilePath property of the object to a document version.
IVersion::CanSaveAsNewSubversion	The method indicates whether or not the user can save changes to a version as a new subversion.
IVersion::CanSaveAsNewVersion	The method indicates whether or not the user can save changes to a version as a new version.

IVersion Methods	Description
IVersion::SaveAsNewSubversion	The method saves a version as a new subversion of the same document.
IVersion::SaveAsNewVersion	The method saves a version as a new version of the same document.
IVersion::CancelTransfer	The method cancels downloading or uploading a version.
IVersion::LockVersionForEdit	The method locks a document's version for editing.
IVersion::UnlockVersion	The method unlocks a document's version.
IVersion::CheckOut	The method checks out a version.
IVersion::ShowSaveAsOptionsForm	The method displays a custom Save As options form and allows the user to select an option.
IVersion::Update	The method updates the version record in the database.
IVersion::Delete	The method deletes a document's version.
IVersion::ShowProperties	The method displays the customizable property form for a version.

Properties

IVersion Properties	Description
IVersion::Author	The property returns or sets the version author.
IVersion::Typist	The property returns or sets the version typist.
IVersion::VersionLabel	The property returns the version label.
IVersion::Profile	The property returns the Profile object on which a version exists.
IVersion::Comment	The property returns or sets the version comments.
IVersion::VersionID	The property returns the version identifier.
IVersion::LastEditDate	The property returns the date and time of the day a document version was last edited.
IVersion::FilePath	The property returns the full path to the local file with a copy of a version's contents.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "fsplugin.dll"

See Also

"ISearch" on page 567

IVersion::Author

The property returns or sets the version author.

C++ Syntax

```
HRESULT Author(  BSTR* pbstrAuthor);
HRESULT Author(  BSTR bstrAuthor);
```

VBScript Syntax

```
[ pbstrAuthor = ] Version.Author
```

```
Version.Author [ = bstrAuthor ]
```

Parameters

pbstrAuthor	[out, retval] A string that specifies the version author's name.
bstrAuthor	[in] A string that specifies the version author's name.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access:

Read/Write

Remarks

After changing the version author it is necessary to call the Update method on the object to update the record. Otherwise, all changes will be lost.

Example

The following example changes the author of the current version of the document with number 20.

```
Dim App As New Application
Dim Ver As Version
Set Ver = App.CurrentLibrary.GetProfile(20).CurrentVersion
Ver.Author = "OlegGa"
Ver.Update
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

IVersion::CancelTransfer

The method cancels downloading or uploading a version.

C++ Syntax

```
HRESULT CancelTransfer(  long lTransferID);
```

VBScript Syntax

```
Version.CancelTransfer (lTransferID)
```

Parameters

lTransferID	[in] Required. A long that specifies the process identifier. This identifier may be obtained from the previous call to the DownloadFile and UploadFile methods.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

"IProfile" on page 419

IVersion::CanSaveAsNewSubversion

The method indicates whether or not the user can save changes to the version as a new subversion.

C++ Syntax

```
HRESULT CanSaveAsNewSubversion(  VARIANT_BOOL* pbCanSave);
```

VBScript Syntax

```
[ pbCanSave  = ] Version.CanSaveAsNewSubversion
```

Parameters

pbCanSave	[out,retval] Returns a Boolean value that specifies whether or not the user can save changes to the current version as a new subversion. True: The user can save edits to the version as a new subversion. False: The user cannot save edits to the version as a new subversion.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The CanSaveAsNewSubversion method determines whether or not the user can create more subversions if they are based on the user's primary group rights, the document's Document Type parameters, and the System Parameters Settings for the library.

The method always returns False for attachments.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

"IProfile" on page 419

IVersion::CanSaveAsNewVersion

The method indicates whether or not the user can save changes to the version as a new version.

C++ Syntax

```
HRESULT CanSaveAsNewVersion(  VARIANT_BOOL* pbCanSave);
```

VBScript Syntax

```
[ pbCanSave  = ] Version.CanSaveAsNewVersion
```

Parameters

pbCanSave	[out,retval] Returns a Boolean value that specifies whether or not the user can save changes to the current version as a new version of the same document. True: The user can save edits to the version as a new version. False: The user cannot save edits to the version as a new version.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The CanSaveAsNewVersion method determines whether or not the user can create more versions if they are based on the user's primary group rights, the document's Document Type parameters, and the System Parameters Settings for the library.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

"IProfile" on page 419

IVersion::CheckOut

The method checks out a version.

C++ Syntax

```
HRESULT CheckOut(  DATE dateReturn,  BSTR bstrComments,  BSTR
bstrPath,  VARIANT_BOOL* pbSuccess);
```

VBScript Syntax

```
[ pbSuccess  = ] Version.CheckOut (dateReturn, bstrComments,
bstrPath)
```

Parameters

dateReturn	[in] Required. A date that specifies when the profile is to be checked in.
bstrComments	[in] Required. A string of comments.
bstrPath	[in] Required. A string that specifies the directory where the downloaded file is to be copied.
pbSuccess	[out,retval] Returns a Boolean value that specifies whether or not the operation succeeded. True: The version was successfully checked out. False: The operation was canceled or it was impossible to check out the version.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Example

The following example checks out the current version of the document with number 20 on the primary library.

```
Dim App As New Application
Dim Prof As Profile
Set Prof = App.PrimaryLibrary.GetProfile(20)
If Not Prof Is Nothing Then
Prof.CurrentVersion.CheckOut DateAdd("d", 10, Now), "", "C:\"
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

"IProfile" on page 419

IVersion::Comment

The property returns or sets the version comments.

C++ Syntax

```
HRESULT Comment(  BSTR* pbstrComment);
HRESULT Comment(  BSTR bstrComment);
```

VBScript Syntax

```
[ pbstrComment = ] Version.Comment
Version.Comment [ = bstrComment ]
```

Parameters

pbstrComment	[out, retval] A string that specifies the version comments.
bstrComment	[in] A string that specifies the version comments.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access:

Read/Write

Remarks

After changing the version comments it is necessary to call the Update method on the object to update the record. Otherwise, all changes will be lost.

Example

The following example changes the comments of the last version of the document with number 20.

```
Dim App As New Application
Dim Ver As Version
Set Ver = App.CurrentLibrary.GetProfile(20).LastVersion
Ver.Comment = "This is the last version"
Ver.Update
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

IVersion::Delete

The method deletes a document's version.

C++ Syntax

```
HRESULT Delete(  VARIANT_BOOL* pbSuccess);
```

VBScript Syntax

```
[ pbSuccess  = ] Version.Delete
```

Parameters

pbSuccess	[out,retval] Returns a Boolean value that specifies whether or not the operation succeeded. True: A version was successfully deleted. False: The operation was canceled or it was impossible to delete the version.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

After deleting, a Version object becomes undefined, i.e. no other properties or methods can be called on the object.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

"IProfile" on page 419

IVersion::DownloadFile

The method downloads the contents of a document version to the file specified by the FilePath property of an object.

C++ Syntax

```
HRESULT DownloadFile(  IDocTransferNotify* pNotify,  long*
plTransferID);
```

VBScript Syntax

```
[ plTransferID  = ] Version.DownloadFile (pNotify)
```

Parameters

pNotify	[in, optional] Optional. A DocTransferNotify object that should receive notifications about download progress.
plTransferID	[out, retval] Returns a long that specifies the process identifier.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

If the most up to date copy of the version already exists on the local disk, nothing is downloaded. If the pNotify parameter was not specified, the method returns only after finishing downloading. Otherwise, the return value may be specified as a parameter of the CancelTransfer method to cancel downloading.

Example

The following example downloads the version selected by the user on the document with number 20 from the primary library and copies it to a local file.

```
Dim App As New Application
```

```
Dim Prof As Profile, Ver As Version
Set Prof = App.PrimaryLibrary.GetProfile(20)
If Not Prof Is Nothing Then
' Select a version of the document
Set Ver = Prof.ShowVersionChooserForm(VCF_Default, "Versions")
If Not Ver Is Nothing Then
' Copy the version to a local file
Ver.DownloadFile Nothing
FileCopy Ver.FilePath, "C:\MyDoc.doc"
End If
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

"IProfile" on page 419

IVersion::Export

The method exports the contents of a version to a file.

C++ Syntax

```
HRESULT Export();
```

VBScript Syntax

```
Version.Export
```

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The user will be asked to select a file to export the version through the standard SaveAs file dialog.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

"IProfile" on page 419

IVersion::FilePath

The property returns the full path to the local file with a copy of version's contents.

C++ Syntax

```
HRESULT FilePath(  BSTR* pbstrFilePath);
```

VBScript Syntax

```
[ pbstrFilePath = ] Version.FilePath
```

Parameters

pbstrFilePath	[out, retval] Returns a string that specifies the path to the version copy.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access:

Read Only

Remarks

The path is determined automatically and does not require version downloading.

Example

The following example logs into the DM Server, creates a new MS Word document on the primary library, and uploads a local file to the document's contents.

```
Dim App As New Application
Dim Prof As Profile, Ver As Version
```

```
' Log into the DM Server
If App.DST = "" Then Exit Sub

' Create a new profile on the primary library
Set Prof = App.PrimaryLibrary.SelectProfileForm(OF_NORMAL, _
"MS WORD")
If Not Prof Is Nothing Then
' Display the profile form and save it
Prof.ShowProfile SP_EDIT
If Prof.Save(OF_NORMAL) Then
' Set the current version of the document to
' a local MS Word file
Set Ver = Prof.CurrentVersion
Ver.LockVersionForEdit OF_NORMAL
FileCopy "C:\MyDoc.doc", Ver.FilePath
Ver.UploadFile Nothing
Ver.UnlockVersion OF_NORMAL
End If
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

IVersion::LastEditDate

The property returns the date and time of the day a document version has been last edited.

C++ Syntax

```
HRESULT LastEditDate(  DATE* pdateLastEdit);
```

VBScript Syntax

```
[ pdateLastEdit = ] Version.LastEditDate
```

Parameters

pdateLastEdit	[out, retval] Returns a date that specifies the requested date and time.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access:

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

IVersion::LockVersionForEdit

The method locks a document's version for editing.

C++ Syntax

```
HRESULT LockVersionForEdit(  short sFlags,  VARIANT_BOOL*
pbSuccess);
```

VBScript Syntax

```
[ pbSuccess  = ] Version.LockVersionForEdit (sFlags)
```

Parameters

sFlags	[in] Required. An integer value that specifies how the method should be executed. The parameter can be a combination of the MethodsFlags.
pbSuccess	[out,retval] Returns a Boolean value that specifies whether or not the operation succeeded. True: The operation succeeded. False: The operation failed because of an internal error or the user canceled the operation.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The LockVersionForEdit method should be used to lock a document's version before changing it. When the document is no longer needed use the UnlockVersion method to unlock the version.

There is no need to call up the method when opening a document because it will be locked automatically.

Example

The following example uploads a local file to the version selected by the user on the document with number 20 from the primary library.

```
Dim App As New Application
Dim Prof As Profile, Ver As Version

Set Prof = App.PrimaryLibrary.GetProfile(20)
If Not Prof Is Nothing Then
If Prof.SelectForEdit(OF_NORMAL + SFE_LOCK_DOCUMENT) = _
SFER_OK Then
' Select a version of the document
Set Ver = Prof.ShowVersionChooserForm(VCF_Default, "Versions")
If Not Ver Is Nothing Then
' Set the version to a local MS Word file
```

```
Ver.LockVersionForEdit OF_NORMAL
FileCopy "C:\MyDoc.doc", Ver.FilePath
Ver.UploadFile Nothing
Ver.UnlockVersion OF_NORMAL
End If
Prof.ReleaseDoc
End If
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

"IProfile" on page 419

IVersion::Open

The method opens a version in the associated application in the edit mode.

C++ Syntax

```
HRESULT Open(  long lMode,  VARIANT_BOOL* pbSuccess);
```

VBScript Syntax

```
[ pbSuccess  = ] Version.Open (lMode)
```

Parameters

lMode	[in] Required. A long that specifies the mode in which the version will be opened. The parameter can be a combination of the AccessMode values.
pbSuccess	[out,retval] Returns a Boolean value that specifies whether or not the operation succeeded. True: The operation succeeded. False: The operation failed because of an internal error or the user canceled the operation.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The version will be opened in its native application (the application specified in its profile's record). If the application is not already running, this method will start the application and open the document.

Example

The following example opens the user selected document (with number 20) version with from the primary library.

```
Dim App As New Application
Dim Prof As Profile, Ver As Version
Set Prof = App.PrimaryLibrary.GetProfile(20)
If Not Prof Is Nothing Then
Set Ver = Prof.ShowVersionChooserForm(VCF_Default, "Versions")
If Not Ver Is Nothing Then
Ver.Open AM_READWRITE
End If
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

"IProfile" on page 419

IVersion::OpenInCompatibleApp

The method opens a document version in an application compatible with its native application.

C++ Syntax

```
HRESULT OpenInCompatibleApp( short sFlags, long lMode, VARIANT vApp, VARIANT_BOOL* pbSuccess);
```

VBScript Syntax

```
[ pbSuccess = ] Version.OpenInCompatibleApp (sFlags, lMode, vApp)
```

Parameters

sFlags	[in] Required. An integer value that specifies how the method should be executed. The parameter can be a combination of the MethodsFlags values.
lMode	[in] Required. A long that specifies the mode in which the version will be opened. The parameter can be a combination of the AccessMode values.
vApp	[in,optional] Optional. An expression that specifies a compatible application used to open the version. If a string expression, vApp should specify the application identifier (e.g., “MS WORD”). If an object, vApp should specify a valid AppInfo object that identifies an application compatible with the application associated with the version’s parent document.
pbSuccess	[out,retval] Returns a Boolean value that specifies whether or not the operation succeeded. True: The operation succeeded. False: The operation failed because of an internal error or the user canceled the operation.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The version will be opened in an application compatible with its native application (the application specified in its profile’s record). If application is not specified the list of compatible applications will pop up allowing the user to select one of them. If the compatible application

is not already running, this method will start the application and open the version.

Example

The following example opens the version, the user selected on the document with number 20, in an compatible application.

```
Dim App As New Application

Dim Prof As Profile, Ver As Version

Set Prof = App.PrimaryLibrary.GetProfile(20)

If Not Prof Is Nothing Then

Set Ver = Prof.ShowVersionChooserForm(VCF_Default, "Versions")

If Not Ver Is Nothing Then

Ver.OpenInCompatibleApp OF_SHOWFORM, AM_READWRITE

End If

End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

"IProfile" on page 419

IVersion::Preview

The method views the contents of a document version.

C++ Syntax

```
HRESULT Preview(  VARIANT vReserved,  VARIANT_BOOL* pbSuccess);
```

VBScript Syntax

```
[ pbSuccess  = ] Version.Preview (vReserved)
```

Parameters

vReserved	[in, optional] Optional. A reserved parameter that should not be specified.
pbSuccess	[out,retval] Returns a Boolean value that specifies whether or not the operation succeeded. True: The version's contents were successfully displayed in the viewer. False: The operation was canceled or it was impossible to display the version's contents in the viewer.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

If the application has the defined viewer set in Application Maintenance, then that viewer will be used. Otherwise the DM Extensions viewer will be used to display the contents of the version.

Example

The following example displays the contents of the current version of the document with number 20 on the primary library.

```
Dim App As New Application
Dim Prof As Profile
Set Prof = App.PrimaryLibrary.GetProfile(20)
If Not Prof Is Nothing Then
Prof.CurrentVersion.Preview
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

"IProfile" on page 419

IVersion::PrintVersion

The method prints a document version.

C++ Syntax

```
HRESULT PrintVersion(  VARIANT_BOOL* pbSuccess);
```

VBScript Syntax

```
[ pbSuccess  = ] Version.PrintVersion
```

Parameters

vReserved	[in, optional] Optional. A reserved parameter that should not be specified.
pbSuccess	[out,retval] Returns a Boolean value that specifies whether or not the operation succeeded. True: The version was successfully printed. False: The operation was canceled or it was impossible to print the version.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

"IProfile" on page 419

IVersion::Profile

The property returns the Profile object on which a version exists.

C++ Syntax

```
HRESULT Profile(  IProfile** ppProfile);
```

VBScript Syntax

```
[ Set ppProfile = ] Version.Profile
```

Parameters

ppProfile	[out, retval] Returns a Profile object that specifies the parent profile for a version.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access:

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

IVersion::SaveAsNewSubversion

The method saves a version as a new subversion of the same document.

C++ Syntax

```
HRESULT SaveAsNewSubversion(  short sFlags,  BSTR bstrComments,
VARIANT vAuthor,  VARIANT vTypist,  long lTypingTime,  long
lKeystrokes,  IVersion** ppVersion);
```

VBScript Syntax

```
[ Set ppVersion  = ] Version.SaveAsNewSubversion (sFlags,
bstrComments, vAuthor, vTypist, lTypingTime, lKeystrokes)
```

Parameters

sFlags	[in] Required. An integer value that specifies how the method should be executed. The parameter can be a combination of the MethodsFlags values.
bstrComments	[in] Required. A string that specifies the subversion comments.
vAuthor	[in] Required. A string of the subversion author's name.
vTypist	[in] Required. A string of the subversion typist's name.
lTypingTime	[in] Required. A long that specifies the time in seconds for typing. If -1 is passed for this parameter, then the values obtained by DM Extensions activity monitoring will be used.
lKeystrokes	[in] Required. A long that specifies the number of keystrokes. If -1 is passed for this parameter, then the values obtained by DM Extensions' activity monitoring will be used.
ppVersion	[out,retval] Returns a Version object that specifies the newly created subversion. If the operation was canceled or an error occurred, the return value is NULL.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The user would not be able to save a new subversion if the user's Primary Group version control parameters are set to prevent additional subversions, or if the Document Type of this document prevents additional subversions, or if System parameters are set to prevent additional subversions. A new subversion cannot be created for an attachment.

The method will fail if the document profile is not selected for editing.

Example

The following example creates a new subversion for the document with number 20 on the primary library.

```
Dim App As New Application
Dim Prof As Profile, Ver As Version
Dim TimeStart As Date
Set Prof = App.PrimaryLibrary.GetProfile(20)
If Not Prof Is Nothing Then
Prof.SelectForEdit OF_NORMAL + SFE_LOCK_DOCUMENT
Set Ver = Prof.CurrentVersion
TimeStart = Now
' Some operations with the current version
...
Ver.SaveAsNewSubversion OF_SHOWFORM, "", "", "", _
DateDiff("n", TimeStart, Now), -1
Prof.ReleaseDoc
End If
```

—

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

"IProfile" on page 419

IVersion::SaveAsNewVersion

The method saves a version as a new version of the same document.

C++ Syntax

```
HRESULT SaveAsNewVersion( short sFlags, BSTR bstrComments,
VARIANT vAuthor, VARIANT vTypist, long lTypingTime, long
lKeystrokes, IVersion** ppVersion);
```

VBScript Syntax

```
[ Set ppVersion = ] Version.SaveAsNewVersion (sFlags,
bstrComments, vAuthor, vTypist, lTypingTime, lKeystrokes)
```

Parameters

sFlags	[in] Required. An integer value that specifies how the method should be executed. The parameter can be a combination of the MethodsFlags values.
bstrComments	[in] Required. A string that specifies the subversion comments.
vAuthor	[in] Required. A string of the subversion author's name.
vTypist	[in] Required. A string of the subversion typist's name.
lTypingTime	[in] Required. A long that specifies the time in seconds for typing. If -1 is passed for this parameter, then the values obtained by DM Extensions' activity monitoring will be used.
lKeystrokes	[in] Required. A long that specifies the number of keystrokes. If -1 is passed for this parameter, then the values obtained by DM Extensions' activity monitoring will be used.
ppVersion	[out,retval] Returns a Version object that specifies the newly created subversion. If the operation was canceled or an error occurred, the return value is NULL.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The user would not be able to save a new version if the user's Primary Group version control parameters are set to prevent additional versions, or if the Document Type of this document prevents additional versions, or if System parameters are set to prevent additional versions.

The method will fail if the document profile is not selected for editing.

Example

The following example creates a new version for the document with number 20 on the primary library.

```
Dim App As New Application
Dim Prof As Profile, Ver As Version
Dim TimeStart As Date
Set Prof = App.PrimaryLibrary.GetProfile(20)
If Not Prof Is Nothing Then
Prof.SelectForEdit OF_NORMAL + SFE_LOCK_DOCUMENT
Set Ver = Prof.CurrentVersion
TimeStart = Now

' Some operations with the current version
...

Ver.SaveAsNewVersion OF_SHOWFORM, "", "", "", _
DateDiff("n", TimeStart, Now), -1
Prof.ReleaseDoc
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

"IProfile" on page 419

IVersion::ShowProperties

The method displays the customizable property form for a version.

C++ Syntax

```
HRESULT ShowProperties(  short sFlags,  short* psResult);
```

VBScript Syntax

```
[ psResult  = ] Version.ShowProperties (sFlags)
```

Parameters

sFlags	[in] Required. Returns an integer value that specifies the form appearance. The parameter can be a combination of the VersionPropertiesFlags values.
psResult	[out, retval] Returns an integer that specifies user's action on the form. For possible values see FormsResults Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The method is called when there is a need to display the version's properties dialog to edit the version author, typist, label, and comments. The dialog can be customized and its form should support the IPropertiesForm interface. To display the form, the method calls up the Show method on IPropertiesForm.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

"IProfile" on page 419

IVersion::ShowSaveAsOptionsForm

The method displays a custom Save As options form and allows the user to select an option.

C++ Syntax

```
HRESULT ShowSaveAsOptionsForm(  short* psFlags,  IVersion**
ppVersion);
```

VBScript Syntax

```
[ ppVersion  = ] Version.ShowSaveAsOptionsForm (psFlags)
```

Parameters

psFlags	[in, out] Required. An integer value that specifies how the method should be executed. The parameter can be zero or a combination of the SAOFFlags values. On return this parameter specifies user's action on the form. For possible values see FormsResults Enumeration.
ppVersion	[out,retval] Returns a Version object that specifies the saved version. It may be the same version, a new version/subversion of the same document or a new document. If the operation was canceled or an error occurred, the return value is NULL. True: The version's contents were successfully displayed in the viewer. False: The operation was canceled or it was impossible to display the version's contents in the viewer.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

This is the easy way to allow the user to select Save As options. The options given to the user are: save as new document, save as new version, save as new subversion, and replace original. If the document is not selected for editing the method will fail.

The form can be customized and should support the ISaveAsOptionsForm interface. To display the form, the method calls up the ShowSaveAsOptionsForm method on ISaveAsOptionsForm.

Example

The following example creates a copy of a version on the document with number 20 on the primary library.

```
Dim App As New Application
Dim Prof As Profile,
Dim Ver, NewVer As Version
Dim flags As Integer

Set Prof = App.PrimaryLibrary.GetProfile(20)
If Not Prof Is Nothing Then
Prof.SelectForEdit OF_NORMAL Or SFE_LOCK_DOCUMENT
' Select a version of the document
Set Ver = Prof.ShowVersionChooserForm(VCF_Default, "Versions")
If Not Ver Is Nothing Then
flags = SAOF_NO_NEW_DOC Or SAOF_DISABLE_REPLACE
Set NewVer = Ver.ShowSaveAsOptionsForm(flags)
If Not NewVer Is Nothing Then
MsgBox "ShowSaveAsOptionsForm succeeded."
End If
End If
Prof.ReleaseDoc
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

"IProfile" on page 419

IVersion::Typist

The property returns or sets the version typist.

C++ Syntax

```
HRESULT Typist(  BSTR* pbstrTypist);
HRESULT Typist(  BSTR bstrTypist);
```

VBScript Syntax

```
[ pbstrTypist = ] Version.Typist
Version.Typist [ = bstrTypist ]
```

Parameters

pbstrTypist	[out, retval] A string that specifies the version typist's name.
bstrTypist	[in] A string that specifies the version typist's name.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access:

Read/Write

Remarks

After changing the version typist it is necessary to call the Update method on the object to update the record. Otherwise, all changes will be lost.

Example

The following example changes the typist of the current version of the document with number 20.

```
Dim App As New Application
Dim Ver As Version
Set Ver = App.CurrentLibrary.GetProfile(20).CurrentVersion
Ver.Typist = "OlegGa"
Ver.Update
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

IVersion::UnlockVersion

The method unlocks a document's version.

C++ Syntax

```
HRESULT UnlockVersion(  short sFlags,  VARIANT_BOOL* pbSuccess);
```

VBScript Syntax

```
[ pbSuccess  = ] Version.UnlockVersion (sFlags)
```

Parameters

sFlags	[in] Required. An integer value that specifies how the method should be executed. The parameter can be a combination of the MethodsFlags values.
pbSuccess	[out,retval] Returns a Boolean value that specifies whether or not the operation succeeded. True: The operation succeeded. False: The operation failed because of an internal error or the user canceled the operation.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

A document's version that has been locked should be unlocked in order to remove locking the version.

Example

The following example uploads a local file to the version selected by the user of the document with number 20 from the primary library..

```
Dim App As New Application
Dim Prof As Profile, Ver As Version

Set Prof = App.PrimaryLibrary.GetProfile(20)
If Not Prof Is Nothing Then
If Prof.SelectForEdit(OF_NORMAL + SFE_LOCK_DOCUMENT) = _
```

```
SFER_OK Then

' Select a version of the document

Set Ver = Prof.ShowVersionChooserForm(VCF_Default, "Versions")

If Not Ver Is Nothing Then

' Set the version to a local MS Word file

Ver.LockVersionForEdit OF_NORMAL

FileCopy "C:\MyDoc.doc", Ver.FilePath

Ver.UploadFile Nothing

Ver.UnlockVersion OF_NORMAL

End If

Prof.ReleaseDoc

End If

End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

"IProfile" on page 419

IVersion::Update

The method updates the version record in the database.

C++ Syntax

```
HRESULT Update(  VARIANT_BOOL* pbSuccess);
```

VBScript Syntax

```
[ pbSuccess  = ] Version.Update
```

Parameters

pbSuccess	[out,retval] Returns a Boolean value that specifies whether or not the operation succeeded. True: The version was successfully updated. False: It was impossible to update the version.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The method should follow any call to a property that affects the version data, such as Author, Typist, and Comment.

Example

The following example changes the author of the current version of the document with number 20.

```
Dim App As New Application
Dim Ver As Version
Set Ver = App.CurrentLibrary.GetProfile(20).CurrentVersion
Ver.Author = "OlegGa"
Ver.Update
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

"IProfile" on page 419

IVersion::UploadFile

The method uploads the file specified by the FilePath property of the object to a document version.

C++ Syntax

```
HRESULT UploadFile(  IDocTransferNotify* pNotify,  long*
plTransferID);
```

VBScript Syntax

```
[ plTransferID  = ] Version.UploadFile (pNotify)
```

Parameters

pNotify	[in, optional] Optional. A DocTransferNotify object that should receive notifications about upload progress.
plTransferID	[out,retval] Returns a long that specifies the process identifier.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

If the pNotify parameter was not specified, the method returns only after finishing uploading. Otherwise, the return value may be specified as a parameter of the CancelTransfer method to cancel uploading. Before uploading, the SelectForEdit and LockVersionForEdit methods should be called on the Profile and Version objects respectively.

Example

The following example uploads a local file to the version selected by the user of the document with number 20 from the primary library.

```
Dim App As New Application
Dim Prof As Profile, Ver As Version

Set Prof = App.PrimaryLibrary.GetProfile(20)
If Not Prof Is Nothing Then
If Prof.SelectForEdit(OF_NORMAL + SFE_LOCK_DOCUMENT) = _
SFER_OK Then
' Select a version of the document
Set Ver = Prof.ShowVersionChooserForm(VCF_Default, "Versions")
If Not Ver Is Nothing Then
' Set the version to a local MS Word file
```

```
Ver.LockVersionForEdit OF_NORMAL
FileCopy "C:\MyDoc.doc", Ver.FilePath
Ver.UploadFile Nothing
Ver.UnlockVersion OF_NORMAL
End If
Prof.ReleaseDoc
End If
End If
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

"IProfile" on page 419

IVersion::VersionID

The property returns the version identifier.

C++ Syntax

```
HRESULT VersionID(  long* plID);
```

VBScript Syntax

```
[ plID = ] Version.VersionID
```

Parameters

plID	[out, retval] Returns a long that uniquely identifies the version.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access:

Read Only

Remarks

The version identifier is unique within a library, i.e. two versions of documents on the same library have different identifiers.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

IVersion::VersionLabel

The property returns the version label.

C++ Syntax

```
HRESULT VersionLabel(  BSTR* pbstrLabel);
```

VBScript Syntax

```
[ pbstrLabel = ] Version.VersionLabel
```

Parameters

pbstrLabel	[out, retval] Returns a string that specifies the version label.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access:

Read Only

Remarks

The label for a version indicates its number and consists of an integer and letter combination (e.g., "1" or "1a"). The attachment label is specified when creating an attachment with the CreateVersionFromFile method on Profile. The attachment label is a three characters long string. The first character should be in the A-Z (uppercase) range. The others must be any uppercase Latin letter or digit.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

IVersion2

The interface is implemented by the version object, inherits its definition from IVersion, but allows additionally external objects to work with the version's RIFF that contains search term highlighting information for the Fulcrum FulView ActiveX control.

The interface inherits its definition from IVersion, so its implementation should also include all properties and methods of IVersion.

Methods

IVersion2 Methods	Description
IVersion2::Download	Downloads a version to a local file.

Properties

IVersion2 Properties	Description
IVersion2::LocalFile	Returns the path to the version's local file.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

IVersion2::Download

The method downloads a version to a local file.

C++ Syntax

```
HRESULT Download(VARIANT vFlags, VARIANT vNotify, long*
pdwTransferID);
```

Visual Basic Syntax

```
[ pdwTransferID = ] Version2.Download ( [vFlags], vNotify)
```

Parameters

vFlags	[in, optional] Specifies where the user wants to download the version. The parameter can be either of the type VT_ERROR (DISP_E_PARAMNOTFOUND) or a numerical value that specifies one of the FileTypes enumeration flags. If VT_ERROR or LFT_NATIVE, the methods downloads version to the local file (the same operation as IVersion::DownloadFile). If LFT_RIFF, the method downloads the version to a temporary RIFF file containing search term highlighting information.
vNotify	[in] Specifies an object that will receive notifications about download progress. The parameter can be either of the type VT_ERROR (DISP_E_PARAMNOTFOUND) or a pointer to the IDocTransferNotify interface to be used in this operation. If this parameter is not NULL the operation should be performed asynchronously.
pdwTransferID	[out] An address of a long pointer variable that receives a pointer to the download identifier. Upon successful return, the identifier can be used to cancel asynchronous downloading by the IVersion::CancelTransfer method.

Return Values

S_OK	The version was downloaded successfully.
S_FALSE	The version was downloaded successfully. The LFT_RIFF flag was specified but no search term highlighting information is available.

E_OUTOFMEMORY	There is not enough memory to retrieve the path.
E_UNEXPECTED	An unexpected error occurred.
E_INVALIDARG	pdwTransfer is not a valid pointer. The OM_INVALID_ARGUMENTS error code should be also set.
CTL_E_INVALIDPROPERTYVALUE	vFlags has an invalid value.
DISP_E_TYPEMISMATCH	vFlags or vNotify is not of valid type.

Remarks

Downloads a version to a local file.

This method superseded IVersion::DownloadFile and can be used to download RIFF as well as the version's native file.

If the most up to date copy of the file already exists on the local disk, nothing is downloaded. If the vNotify parameter was not specified or NULL, the method returns only after finishing downloading. Otherwise, the operation is performed asynchronously and pdwTransferID may be specified as a parameter of the IVersion::CancelTransfer method to cancel downloading the version.

The method can download the version even if no search term highlighting information is available but the LFT_RIFF flag was specified; for example, if no context search was executed to get the document's profile. In that case, the method operates like IVersion::DownloadFile but returns S_FALSE when successful.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

"IVersion2" on page 663

"IDocTransferNotify" on page 327

IVersion2::LocalFile

The property returns the path to the version's local file or temporary RIFF file.

C++ Syntax

```
HRESULT LocalFile(VARIANT vFlags, BSTR* pbstrPath)
```

Visual Basic Syntax

```
[ pbstrPath = ] Version2.LocalFile [ (vFlags) ]
```

Parameters

vFlags	[in, optional] Specifies a file path the user wants to retrieve. The parameter can be either of the type VT_ERROR (DISP_E_PARAMNOTFOUND) or a numerical value that specifies one of the FileTypes enumeration flags. If VT_ERROR or LFT_NATIVE, the property returns the path to the version's local file in its pbstrPath parameter (the same as IVersion::FilePath). If LFT_RIFF, the property returns the path to a temporary RIFF file containing search term highlighting information.
pbstrPath	[out, retval] An address of the BSTR pointer variable that receives a pointer to the local file path string. Upon successful return, this string is a Basic wide character string (two bytes per character) containing the path to the version's local file (as specified by vFlags). This path must coincide with IVersion::FilePath if no search term highlighting information is available or vFlags is LFT_NATIVE. The implementation should allocate the string returned in pbstrPath, and the caller is responsible for freeing it. If an error occurs, the implementation should set *pbstrPath to NULL.

Return Values

S_OK	The path to either the version's local file or RIFF was retrieved successfully.

S_FALSE	The path to the version's local file was retrieved successfully. The LFT_RIFF flag was specified but no search term highlighting information is available.
E_OUTOFMEMORY	There is not enough memory to retrieve the path.
E_UNEXPECTED	An unexpected error occurred.
E_INVALIDARG	pbstrPath is not a valid pointer. The OM_INVALID_ARGUMENTS error code should be also set.
CTL_E_INVALIDPROPERTYVALUE	vFlags has an invalid value.
DISP_E_TYPEMISMATCH	vFlags is not of valid type.

Remarks

Returns the path to the version's local file.

The property superseded the IVersion::FilePath property. The property can be used to retrieve the path to the version's local file where the version is downloaded by IVersion2::Download or wherefrom the version can be uploaded.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion2" on page 663

IVersion3

The interface extends functionaloity of Version object.

The interface inherits its definition from IVersion2, so its implementation must also include all properties and methods ofIVersion2.

Methods

IVersion3 Methods	Description
IVersion3::UploadFileEx	The method uploads the file specified by the FilePath property of the object to a document version.

Properties

IVersion3 Properties	Description
IVersion3::VersionStatus	Returns/sets version status.
IVersion3::CanChangeStatus	Determines whether the version status can be changed.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import "fsplugin.dll"

IVersion3::CanChangeStatus

Returns/sets version status.

C++ Syntax

```
HRESULT CanChangeStatus(  short sWhatToChange,  VARIANT_BOOL*
pbStatus);
```

VBScript Syntax

```
[ pbStatus = ] Version3.CanChangeStatus ( sWhatToChange )
```

Parameters

sWhatToChange	[in] Specifies what status can be changed. For possible values see CanChangeVersionStatus Enumeration.
pbStatus	[out, retval] Returns result: True: Version status can be changed to specified value. False: Version status cannot be changed to specified value.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IVersion3" on page 669

IVersion3::UploadFileEx

The method uploads the file specified by the FilePath property of the object to a document version.

C++ Syntax

```
HRESULT UploadFileEx(  long nFlags,  IDocTransferNotify* pNotify,
IActivityInfo* pActivityInfo,  long* plTransferID);
```

VBScript Syntax

```
[ plTransferID = ] Version3.UploadFileEx ( nFlags, [ pNotify ] , [
pActivityInfo ] )
```

Parameters

nFlags	[in] Upload flags. For possible values see UploadFileExFlags Enumeration.
pNotify	[in, optional] A DocTransferNotify object that should receive notifications about upload progress.
pActivityInfo	[in, optional] An ActivityInfo object that contains cost recovery information that should be added to "Edit" history record.
plTransferID	[out, retval]Returns a long that specifies the process identifier.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IVersion3" on page 669

IVersion3::VersionStatus

Returns/sets version status.

C++ Syntax

```
HRESULT VersionStatus(  long* plStatus);

HRESULT VersionStatus(  long lStatus);
```

VBScript Syntax

```
[ plStatus = ] Version3.VersionStatus

Version3.VersionStatus [ = lStatus ]
```

Parameters

plStatus	[out, retval] Returns version status for possible values see VersionStatus Enumeration.
lStatus	[in] Sets version status for possible values see VersionStatus Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <fsplugin.dll>

See Also

"IVersion3" on page 669

IVersions

The Versions object represents a collection of document's versions.

When to Use

A Versions object may be created with the CreateObject function or the Dim statement with the New keyword. The following example shows the Visual Basic syntax for creating a DOCSObjects.Versions object:

```
Sub Form_Load()
 ...
 Dim Vers As Object
 ' Create a DOCSObjects.Versions object
 Set Vers = CreateObject("DOCSObjects.Versions")
 ...
 End Sub
```

Methods

IVersions Methods	Description
IVersions::Add	Adds an item to a collection.
IVersions::Remove	Removes a specific item from a collection.
IVersions::Clear	Removes all items from a collection.

Properties

IVersions Properties	Description
IVersions::Item	Returns a specific item of a collection either by position or by key.
IVersions::Count	Returns the number of items in a collection.
IVersions::_NewEnum	Returns an object that supports IEnumVARIANT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

"IVersion2" on page 663

IVersions::Add

The method adds a new item to a collection.

C++ Syntax

```
HRESULT Add(IVersion* pVersion);
```

Visual Basic Syntax

```
Versions.Add (pVersion)
```

Parameters

pVersion	[in] Required. A Version object that specifies the item to add to a collection.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Adds an item to a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersions" on page 673

IVersions::Clear

The method removes all items from a collection.

C++ Syntax

```
HRESULT Clear();
```

Visual Basic Syntax

```
Versions.Clear
```

Return Values

S_OK	The operation succeeded.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Removes all items from a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersions" on page 673

IVersions::Count

The property returns the number of items in a collection.

C++ Syntax

```
HRESULT Count(long* plCount);
```

Visual Basic Syntax

```
[ plCount = ] Versions.Count
```

Parameters

plCount	[out, retval] Returns a long that specifies the number of items in the collection.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the number of items in a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersions" on page 673

IVersions::Item

The property returns the specific member of a collection either by position or by key.

C++ Syntax

```
HRESULT Item(VARIANT vIndex, IVersion** ppVersion);
```

Visual Basic Syntax

```
[ Set ppVersion = ] Versions.Item (vIndex)
```

Parameters

vIndex	[in] Required. An expression that specifies the member position in a collection. If a numeric expression, vIndex should be a number from 1 to the value of the collection's Count property. If a string expression, vIndex should correspond to a version label.
ppVersion	[out, retval] Returns an Version object that specifies the requested item

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_INAVLIDARG	An invalid parameter has been passed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Returns the specific item of a collection either by position or by key.

If the value provided as vIndex does not match any existing member of a collection, an error occurs. Item is the default property for a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersions" on page 673

"IVersion" on page 625

IVersions::Remove

The method removes the specific item from a collection.

C++ Syntax

```
HRESULT Remove(VARIANT vIndex);
```

Visual Basic Syntax

```
Versions.Remove (vIndex)
```

Parameters

vIndex	Required. A variant expression specifying the name or index in a collection of the object to be accessed.

Return Values

S_OK	The operation succeeded.
E_INAVLIDARG	An invalid parameter has been passed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

Removes a specific item from a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersions" on page 673

IVersions::_NewEnum

The _NewEnum property returns the IEnumVARIANT interface on an object that can be used to enumerate the collection.

C++ Syntax

```
HRESULT _NewEnum(LPUNKNOWN* ppEnum);
```

Parameters

ppEnum	[out, retval] On return, an IEnumVARIANT object reference that can be used to enumerate the collection.

Remarks

Returns an object that supports IEnumVARIANT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersions" on page 673

Object Model Enumerations

AccessMode Enumeration

This enumeration specifies the available access mode when opening a document or version.

Elements

AccessMode Elements	Description
AM_READ	Indicates whether a document is opened in the read mode.
AM_WRITE	Indicates whether a document is opened in the write mode.
AM_READWRITE	Indicates whether a document is opened in the read-write mode.
AM_COPY	Indicates whether a copy of a document should be opened.
AM_ASYNC	Indicates whether an operation should be asynchronous.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

AccessType Enumeration

This enumeration specifies values that are returned/set by IProfileTrustee2/ISearchTrustee2 interfaces properties.

Elements

AccessType Elements	Description
AT_NONE	Undefined.
AT_ALLOW	Allow.
AT_DENY	Deny.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import "fsplugin.dll"

CanChangeVersionStatus Enumeration

This enumeration specifies flags that are passed to IVersion3::CanChangeStatus property in sWhatToChanges parameter.

Elements

CanChangeVersionStatus Elements	Description
CCVS_CanPublish	Is version can be published.
CCVS_CanUnPublish	Is published status can be removed.
CCVS_CanMakeReadOnly	Is version can be made read-only.
CCVS_CanMakeWritable	Is version can be made writable.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import "fsplugin.dll"

CheckInActions Enumeration

This enumeration contains flags that are passed to IProfile::UnlockProfile in the sAction parameter.

Elements

CheckInActions Elements	Description
UF_UNLOCK	Indicates whether a document should be just unlocked.
UF_REPLACE	Indicates whether a stored document should be replaced by a file.
UF_NEWVERSION	Indicates whether a new document's version should be created.
UF_NEWSUBVERSION	Indicates whether a new document's subversion should be created.
UF_NEWDOCUMENT	Indicates whether a new document should be created.
UF_REMOVE_READONLY	Indicates that read-only status should be removed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

ColumnFlags Enumeration

This enumeration contains flags that used to specify attributes for a column. These attributes are stored in Registry with the "HKEY_LOCAL_MACHINE\SOFTWARE\Hummingbird\PowerDOCS\Core\Plugins\Fusion\Settings\DetailColumnsDefinition" key.

Elements

ColumnFlags Elements	Description
CF_HIDDEN	Do not display a column.
CF_SHOWBYDEFAULT	Display a column by default.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <fsplugin.dll>

ColumnTypes Enumeration

This enumeration contains flags that are returned by the IColumn::Type property.

Elements

ColumnTypes Elements	Description
CT_VARCHAR	Indicates that an object is a string.
CT_DATE	Indicates that an object is a date.
CT_TIME	Indicates that an object is a time.
CT_INTEGER	Indicates that an object is an integer.
CT_MULTIVALUE	Indicates whether an object is a multi-valued field.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IColumn" on page 269

CreateVersionFromFileFlags Enumeration

This enumeration specifies flags that are passed to IProfile::CreateVersionFromFile method in sFlags parameter.

Elements

CreateVersionFromFileFlags Elements	Description
CVF_SERVER_FILENAME	Indicates that file on the fileserver will be created with the same filename.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import "fsplugin.dll"

DataTypes Enumeration

This enumeration contains property types that are returned by the IProperty::Type property.

Elements

DataTypes Elements	Description
DT_None	Indicates that an object type is unspecified.
DT_Short	Indicates that an object is a short integer.
DT_Long	Indicates that an object is a long integer.
DT_Single	Indicates that an object has the single-precision floating-point type.
DT_Double	Indicates that an object has the double-precision floating-point type.
DT_Date	Indicates that an object is a date.
DT_String	Indicates that an object is a string.
DT_Boolean	Indicates that an object is a Boolean value.
DT_Binary	Indicates that an object is a binary stream.
DT_Byte	Indicates that an object is an integer represented by a single byte.
DT_Object	Indicates that an object is an object.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProperty" on page 551

DisplayMessageTypes Enumeration

This enumeration specifies message types that are passed to the IApplication4::DisplayMessage method in the nType parameter.

Elements

DisplayMessageTypes Elements	Description
DMT_NONE	Simple message.
DMT_ERROR	Error message.
DMT_INFO	Info message.
DMT_QUESTION	Question message.
DMT_WARNING	Warning message.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <fsplugin.dll>

See Also

"IApplication4" on page 262

FileTypes Enumeration

This enumeration contains flags that are passed to the IVersion2::Download method in the vFlags parameter.

Elements

FileTypes Elements	Description
LFT_NATIVE	Indicates a version's native application local file (either temporary or shadowed).
LFT_RIFF	Indicates a temporary RIFF file with the search term highlighted.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

FormsResults Enumeration

Defines the form result code.

Elements

FormsResults Elements	Description
FR_CANCELLED	Indicates that operation was cancelled by a user.
FR_OK	Indicates that operation is completed successfully.
FR_DATACHANGED	Indicates that data have been changed during operation.
FR_CLOSED	Indicates that a form has been closed internally.
FR_APPLY_TO_ALL	Indicates that apply to all was checked.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

FTypes Enumeration

This enumeration contains identifiers of the launch method integration schemes that are returned by the IDOCSForm::Type property.

Elements

FTypes Elements	Description
FT_PROFILE	The current form is a profile form.
FT_HITLIST	The current form is a hitlist form.
FT_QBE	The current form is a search form.
FT_DEFAULT_QBE	The current form is a default search form.
FT_DEFAULT_PROFILE	The current form is a default profile form.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IDOCSForm" on page 306

LaunchMethodChooserFlags Enumeration

This enumeration contains flags that are passed to IAppInfo::ShowLaunchMethodChooserForm in the lFlags parameter.

Elements

LaunchMethodChooserFlags Elements	Description
LMC_UseDefault	Indicates that if there is a default launch method, it will be used and no form should be presented.
LMC_IgnoreDefault	Indicates that a form should appear not depending on whether there is a default launch method or not.
LMC_ShowForPrint	Indicates whether launch methods, which are valid for printing, are only shown.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IAppInfo" on page 220

LM_INTEGRATIONSCHEME Enumeration

This enumeration contains identifiers of the launch method integration schemes that are returned by the ILaunchMethod::IntegrationScheme property.

Elements

LM_INTEGRATIONSCHEME Elements	Description
LM_NOINTEGRATION	"No Integration" scheme.
LM_FULL	"Full Integration" scheme.
LM_DIRMON	"Uses Directory Monitoring" scheme.
LM_FULLWITHQS	"Full Integration with Quick Save" scheme.
LM_ODMA	"ODMA Compliant" scheme.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILaunchMethod" on page 331

LockProfileFlags Enumeration

This enumeration specifies flags that are passed to IPofile::LockProfile method in sFlags parameter.

Elements

LockProfileFlags Elements	Description
LPF_LOCK	Lock Profile.
LPF_MAKE_READONLY	Make Read-Only.
LPF_CHECKONLY	Check only if document can be locked.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import "fsplugin.dll"

MethodsFlags Enumeration

Defines standard flags for calling methods.

Elements

MethodsFlags Elements	Description
OF_NORMAL	Indicates that a method may display any messages and forms.
OF_NOERRORS	Indicates that no error messages should be displayed.
OF_NOWARNINGS	Indicates that no warning messages should be displayed.
OF_SILENT	Indicates whether neither error nor warning messages should be displayed.
OF_SHOWFORM	Indicates whether a method should display an associated form.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

Modes Enumeration

This enumeration specifies modes that are returned by the IApplication3::Mode property.

Elements

Modes Elements	Description
MODE_ONLINE	Indicates that the user is working online.
MODE_OFFLINE	Indicates that the user is working offline.
MODE_TOONLINE	Indicates that the DM Server is available but the user has not switched to the online mode.
MODE_TOOFFLINE	Indicates that the DM Server is unavailable but the user has not switched to the offline mode.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IApplication3" on page 261

OMActivityConsts Enumeration

This enumeration contains available document activity constants.

Elements

OMActivityConsts Elements	Description
ACTIVITY_USER_DEFINED	User defined activity.
ACTIVITY_CREATE_DOCUMENT	"Create".
ACTIVITY_EDIT_DOCUMENT	"Edit".
ACTIVITY_PRINT_DOCUMENT	"Print".
ACTIVITY_PROOFREAD_DOCUMENT	"Proofread".
ACTIVITY_ARCHIVE_DOCUMENT	"Archive".
ACTIVITY_DELETE_DOCUMENT	"Delete".
ACTIVITY_FULLTEXT_DOCUMENT	"Full Text Index".
ACTIVITY_CREATE_NEW_VERSION	"New Version".

OMActivityConsts Elements	Description
ACTIVITY_DELETE_VERSION	"Delete Version".
ACTIVITY_IMPORT_DOCUMENT	"Import".
ACTIVITY_CHECKOUT_DOCUMENT	"Check-out".
ACTIVITY_CHECKIN_DOCUMENT	"Check-in".
ACTIVITY_SEND_DOCUMENT	"Transfer To".
ACTIVITY_RECEIVE_DOCUMENT	"Transfer From".
ACTIVITY_ACCESS_DOCUMENT	"Access".
ACTIVITY_ACCESS_DENIED_DOCUMENT	"Access denied".
ACTIVITY_COPIED_TO_DOCUMENT	"Copied To".
ACTIVITY_COPIED_FROM_DOCUMENT	"Copied From".
ACTIVITY_LOOK_DOCUMENT	"Look".
ACTIVITY_EDIT_DOCUMENT_PROFILE	"Edit Profile".
ACTIVITY_MOVE_DOCUMENT	"Move To".
ACTIVITY_FAX_DOCUMENT	"Fax".
ACTIVITY_CHANGE_SECURITY	"Edit Security".
ACTIVITY_TRANSFER_DOCUMENT_FROM	"Transfer From".
ACTIVITY_TRANSFER_DOCUMENT_TO	"Transfer To".
ACTIVITY_RESTORE_DOCUMENT	"Restore".
ACTIVITY_ANNOTATE_DOCUMENT	"Annotate".
ACTIVITY_DISPLAY_DOCUMENT	"Display".
ACTIVITY_CHECKOUT_CANCELLED	"Check-out Cancelled".
ACTIVITY_MAIL_COPY	"Mail Copy".
ACTIVITY_MAIL_COPY_AND_CHECKOUT	"Mail Copy and Check-out".
ACTIVITY_MAIL_REFERENCE	"Mail Reference".
ACTIVITY_CHECKOUT_COPY_DOCUMENT	"Check-out Copy".
ACTIVITY_DOCUMENT_ROUTED	"Routed".
ACTIVITY_DOCUMENT_QUEUED_FOR_DEL	"Queued for Deletion".
ACTIVITY_RELATIONSHIP_CREATED	"Relationship created".
ACTIVITY_RELATIONSHIP_BROKEN	"Relationship removed".

OMActivityConsts Elements	Description
ACTIVITY_MAKE_READ_ONLY	"Make Read-Only".
ACTIVITY_REMOVE_READ_ONLY	"Remove Read-Only".
ACTIVITY_MAKE_VERSION_READ_ONLY	"Make Version Read-Only".
ACTIVITY_MAKE_VERSION_WRITABLE	"Make Version Writable".
ACTIVITY_PUBLISH_VERSION	"Publish Version".
ACTIVITY_RESET_PUBLISHED_VERSION	"Reset Published Version".
ACTIVITY_ADD_LINK	"Add Link".
ACTIVITY_REMOVE_LINK	"Remove Link".
ACTIVITY_UPDATE_LINK	"Update Link".

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

OMErrorCodes Enumeration

This enumeration contains available Object Model error codes that are returned by the IOMErrInfo::ErrCode property.

Elements

OMErrorCodes Elements	Description
OM_SUCCESS	Indicates that an operation succeeded.
OM_CANCELLED_BY_USER	Indicates that an operation was cancelled by the user.
OM_SYSTEM_ERROR	Indicates that a system error occurred.
OM_INVALID_ARGUMENTS	Indicates that invalid arguments were specified for a method.
OM_INSUFFICIENT_RIGHTS	Indicates that the user has insufficient rights to complete an operation.
OM_ERROR_LOAD_FORM	Indicates that an error occurred when loading a form.
OM_ERROR_IN_FORM	Indicates that an error in a form occurred.
OM_ERROR_LOADING_MODULE	Indicates that an error occurred when loading an external module.

OMErrorCodes Elements	Description
OM_ERROR_IN_EXTERNAL_MODULE	Indicates that an error in an external module occurred.
OM_INVALID_OBJECT_STATE	Indicates that an object is in invalid state.
OM_INVALID_OBJECT	Indicates that a method cannot be invoked for an object.
OM_FUSION_ERROR	Indicates that a DM Server error occurred.
OM_INTERNAL_ERROR	Indicates that an internal error occurred.
OM_UNPLUGGED_ERROR	Indicates that the unplugged mode do not support this property or method.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IOMErrInfo" on page 412

OMFlagsLookUp Enumeration

This enumeration contains flags that are passed to IColumn::DoLookup in the sMode parameter.

Elements

OMFlagsLookUp Elements	Description
DL_SHOW_LOOKUP	Indicates that a lookup form should be presented to fill in a lookup field.
DL_VALIDATE_ONLY	Indicates that the value of a lookup field should be validated.
DL_AUTO_COMPLETE	Indicates that a lookup field should be auto completed if possible.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IColumn" on page 269

OMFlagsQR Enumeration

This enumeration contains flags that are passed to IQuickRetrieve::DisplayModal in the sFlags parameter.

Elements

OMFlagsQR Elements	Description
QR_NORMAL	Indicates that a form allows multiple selections.
QR_SINGLE_SELECTION	Indicates that a form does not allow multiple selections.
QR_IN_APPLICATION	Indicates that a form is called from an application.
QR_ONLY_FOLDERS	Indicates that only folders should be displayed.
QR_ONLY_DOCUMENTS	Indicates that only documents should be displayed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IQuickRetrieve" on page 556

ProfileDeleteFlags Enumeration

This enumeration contains flags that are passed to IProfile::Delete in the sFlags parameter.

Elements

ProfileDeleteFlags Elements	Description
PD_DELETE_DOCUMENT	Specifies to delete a document completely.
PD_DELETE_CONTENT	Specifies to delete a document content.
PD_QUEUE_FOR_DELETION	Specifies to queue a document for deletion.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

ProfileRights Enumeration

This enumeration specifies available values for document rights.

Elements

ProfileRights Elements	Description
RIGHTS_PROFILE_VIEW	Allows to view the document's profile.
RIGHTS_PROFILE_EDIT	Allows to make changes to the document's profile.
RIGHTS_DOCUMENT_VIEW	Allows to view the document's contents. Not used for profiled folders.
RIGHTS_DOCUMENT_RETRIEVE	Allows to retrieve the document's contents. Not used for profiled folders.
RIGHTS_DOCUMENT_EDIT	Allows to make changes to the document or changed folder content for profiled folders.
RIGHTS_DOCUMENT_COPY	Allows to copy the document and its profile.
RIGHTS_DOCUMENT_DELETE	Allows to delete the document and its profile.
RIGHTS_DOCUMENT_CONTROL	Allows to grant privileges to and revoke privileges from other trustees.
RIGHTS_VIEW_PUBLISHED_ONLY	Access to view published versions of document only. Not used for profiled folders.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

ProfileStatus Enumeration

This enumeration contains available values for the document profile status.

Elements

ProfileStatus Elements	Description
PS_AVAILABLE	Indicates that a profile has the "Available" status.
PS_DOCUMENT_EDITED	Indicates that a profile has the "Document Being Edited" status.
PS_PROFILE_EDITED	Indicates that a profile has the "Profile Being Edited" status.
PS_CHECKOUT	Indicates that a profile has the "Checked out" status.
PS_NOT_AVAILABLE	Indicates that a profile has the "Not Available" status.
PS_INDEXED	Indicates that a profile has the "Being Indexed" status.
PS_ARCHIVED	Indicates that a profile has the "Archived" status.
PS_BEING_ARCHIVED	Indicates that a profile has the "Being Archived" status.
PS_DELETED	Indicates that a profile has the "Deleted" status.
PS_READONLY	Indicates profile has "Read-Only" status.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

ProjectChooserFlags Enumeration

This enumeration contains flags that are passed to IApplication::ShowProjectChooserForm in the sFlags parameter.

Elements

ProjectChooserFlags Elements	Description
PCF_ALLOWMULTISEL	Indicates whether a form allows multiple project selections.
PCF_PRIMARY	Indicates whether a form includes primary library projects. Obsolete.
PCF_SEARCHBYDEF	Indicates whether a form includes projects from the search by default libraries. Obsolete.
PCF_OTHERS	Indicates whether a form includes projects from libraries other than primary and searched by default. Obsolete.
PCF_ALL	Indicates whether a form includes projects from all libraries. Obsolete.
PCF_CURRENT	Indicates whether a form includes current library projects. Obsolete.
PCF_AUTOSELECT	Auto select if only one project is available.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IApplication" on page 241

ProjectRights Enumeration

Obsolete. ProfileRights values are used for folders.

This enumeration specifies available values for project rights.

Elements

ProjectRights Elements	Description
RIGHTS_USE_PROJECT	Allows to see the project.
RIGHTS_EDIT_PROJECT	Allows to make changes to the project.
RIGHTS_DELETE_PROJECT	Allows to delete the project.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

QRCReturnFlags Enumeration

This enumeration contains flags that are returned from the IQuickRetrieveCallback::OnOk method in the pResult parameter.

Elements

QRCReturnFlags Elements	Description
QRCR_CONTINUE	Closes a form and continues.
QRCR_STOP	Stops closing a form.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IQuickRetrieveCallback" on page 566

QuickSearchesID Enumeration

The enumeration contains pre-defined QuickSearches identifiers.

Elements

QuickSearchesID Elements	Description
QS_NOT_SAVED	Indicates that a search has not been saved.
QS_RECENTLY_EDITED	Indicates that a search is for recently edited documents.
QS_TEMPLATES	Indicates that a search is for templates.
QS_CHECKED_OUT	Indicates that a search is for checked-out documents.
QS_RESTORED	Indicates that a search is for restored documents.
QS_DELETED	Indicates that a search is for deleted documents.
QS_ALLFOLDERS	Indicates that a search is for all folders.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

QuickSearchRights Enumeration

This enumeration specifies available values for QuickSearch rights.

Elements

QuickSearchRights Elements	Description
RIGHTS_USE_SEARCH	Allows to see the quick search and to launch it.
RIGHTS_EDIT_SEARCH	Allows to make changes to the quick search.
RIGHTS_DELETE_SEARCH	Allows to delete the quick search.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

SAOFFlags Enumeration

This enumeration contains flags that are passed to IVersion::ShowSaveAsOptionsForm in the sFlags parameter.

Elements

SAOFFlags Elements	Description
SAOF_NO_NEW_DOC	Disables creating a new document.
SAOF_DISABLE_REPLACE	Disables replacing an original.
SAOF_ENABLE_ONLY_NEW_DOC	Enables only a new document.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

SelectForEditFlags Enumeration

This enumeration contains flags that are passed to IProfile::SelectForEdit in the sFlags parameter.

Elements

SelectForEditFlags Elements	Description
SFE_LOCK_PROFILE	Locks a profile only.
SFE_LOCK_DOCUMENT	Locks a document.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

SelectForEditReturnFlags Enumeration

This enumeration contains flags that are returned by the IProfile::SelectForEdit method in the psResult parameter.

Elements

SelectForEditReturnFlags Elements	Description
SFER_ERROR	An error occurs during selecting a document for editing.
SFER_OK	A document has been selected for editing.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

SelectVersionsFormFlags Enumeration

This enumeration contains flags that are passed to IDocProfiles::ShowSelectVersionsForm in the sFlags parameter.

Elements

SelectVersionsFormFlags Elements	Description
SVF_DisableAllVersions	Indicates that the "All versions" radio button is disabled.
SVF_EnableAllVersions	Indicates that the "All versions" radio button is enabled.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IDocProfiles" on page 292

ShowProfileFlags Enumeration

This enumeration contains flags that are passed to IProfile::ShowProfile in the sFlags parameter.

Elements

ShowProfileFlags Elements	Description
SP_READONLY	Indicates that a profile is presented in the read-only mode.
SP_EDIT	Indicates that a profile can be edited.
SP_NOCHECK_REQUIRED	Indicates whether there is no need to check required fields.
SP_SAVE	Indicates that a profile should be saved when the OK button is pressed.
SP_NODETAILS	Indicates that the "Apply All" check box should be present.
SP_APPLYALL	Indicates that additional tabs should not be displayed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

SPFFlags Enumeration

This enumeration contains flags that are passed to ILibrary::SelectProfileForm in the sFlags parameter.

Elements

SPFFlags Elements	Description
SPF_FROM_FILE	Indicates whether a file extension should be used to select a profile form.
SPF_SHOW_FORM	Always shows a form.
SPF_ALL_FORMS	Shows all available forms.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

SSFFlags Enumeration

This enumeration contains flags that are passed to ILibrary::SelectSearchForm in the sFlags parameter.

Elements

SSFFlags Elements	Description
SSF_SEARCH_RESULTS	Selects a search result form.
SSF_SHOW_ALWAYS	Always shows a form.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"ILibrary" on page 360

TransferNotifyCompletedReason Enumeration

This enumeration contains flags that are passed to IDocTransferNotify::Completed in the sReason parameter.

Elements

TransferNotifyCompletedReason Elements	Description
TN_COMPLETED	Indicates that an operation succeeded.
TN_CANCELLED	Indicates that an operation was cancelled.
TN_FATALERROR	Indicates that an operation was aborted because a fatal error is occurred.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IDocTransferNotify" on page 327

UploadFileExFlags Enumeration

This enumeration specifies flags that are passed to IVersion3::UploadFileEx method in nFlags parameter.

Elements

UploadFileExFlags Elements	Description
UFE_NORMAL	Normal upload.
UFE_FORCE	Forced upload. Quick Save flag should be ignored

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import "fsplugin.dll"

VersionChooserFormFlags Enumeration

This enumeration contains flags that are passed to IProfile::ShowVersionChooserForm in the sFlags parameter.

Elements

VersionChooserFormFlags Elements	Description
VCF_Default	Indicates that a form has two buttons: OK and Cancel.
VCF_Versions	Indicates that a form has only a Close button.
VCF_Edit	Indicates whether editing is allowed.
VCF_MultiSelection	Is accessible only in the Attache mode. Allows to choose the current version for all documents.
VCF_ShowAlways	Always shows the version chooser dialog.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IProfile" on page 419

VersionPropertiesFlags Enumeration

This enumeration contains flags that are passed to IVersion::ShowProperties in the sFlags parameter.

Elements

VersionPropertiesFlags Elements	Description
VP_VERSION	Indicates that a form displays properties of a version.
VP_ATTACHMENT	Indicates that a form displays properties of an attachment.
VP_NEW	Indicates whether properties of a new version/attachment are presented.
VP_EDIT	Indicates whether a form is opened in the edit mode.
VP_APPLYALL	Indicates that the "Apply to all" check box should be present.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <fsplugin.dll>

See Also

"IVersion" on page 625

VersionStatus Enumeration

This enumeration specifies status values that are returned/set by the IVersion3::VersionStatus property.

Elements

VersionStatus Elements	Description
VS_NORMAL	Normal status.
VS_ARCHIVED	Version is archived.
VS_MOVED	Version moved to near-line storage.
VS_DELETED	Version is deleted.
VS_READONLY	Version is read-only.
VS_PUBLISHED	Version is published.

Requirements

DM: Requires DM Extensions 5.1 or later to support VS_ARCHIVED, VS_MOVED, and VS_DELETED methods. Requires DM Extensions 5.0 or later to support VS_NORMAL, VS_READONLY, and VS_PUBLISHED methods.

Header: #import "fsplugin.dll"

Chapter

4

ActiveX Controls

In This Chapter

- "ActiveX Controls Overview" on page 710
- "Hierarchy Control" on page 710
- "Hierarchy List Control" on page 733
- "FavoritesBar Control" on page 772

ActiveX Controls Overview

To help developers create their own applications quickly and easily the Explorer Core offers several ActiveX controls. The most important ones are

- Hierarchy (tree) control
- List pane control
- FavoritesBar control

These controls operate on top of the Explorer Core and thus have all associated benefits. Developers do not have to worry about data presentation. You can just drop these controls onto your form and they will do everything! The controls support all UI features such as drag-and-drop, context menu, keyboard shortcuts without requiring the developer to explicitly write code to support these features. Besides, one can provide event procedures that will be executed in response to control events, so you can take appropriate actions in your application's code.

Hierarchy Control

This control can be used to display a hierarchy of Explorer Core objects. All the standard UI features (drag- and-drop, keyboard shortcuts, context menu) are supported. Besides the control supports standard properties, methods, and events described in Visual Basic.

Explorer Core

Explorer Core is a general integration technology that powers all the standard DM Extensions User Interface. With Explorer Core you have tremendous extensibility and integration capabilities – you can greatly extend DM Extensions functionality and provide integration with DM to any other environment you need.

You can access Explorer Core on several layers – you can write directly to its interfaces (all Explorer Core objects are COM objects) or you can use provided ActiveX controls to integrate it with your product. ActiveX controls allow you to create simple but fully functional Explorer-like application with just 10 lines of VB code! That application will fully support drag-and-drop, context menus,

customizable columns and much more without requiring you to explicitly write code to support these features. This will be shown in the examples below.

Example – Explorer Like application

```
Private Sub Form_Load()


    ' Creates a new Explorer Core session
    Dim s As New BCSession


    ' Creates a new Browser Object. Explorer Core will use this
object to
    ' provide feedback to the User Interface
    Dim b As New MiniBrowser
    Dim r As BCBrowser
    Set r = b

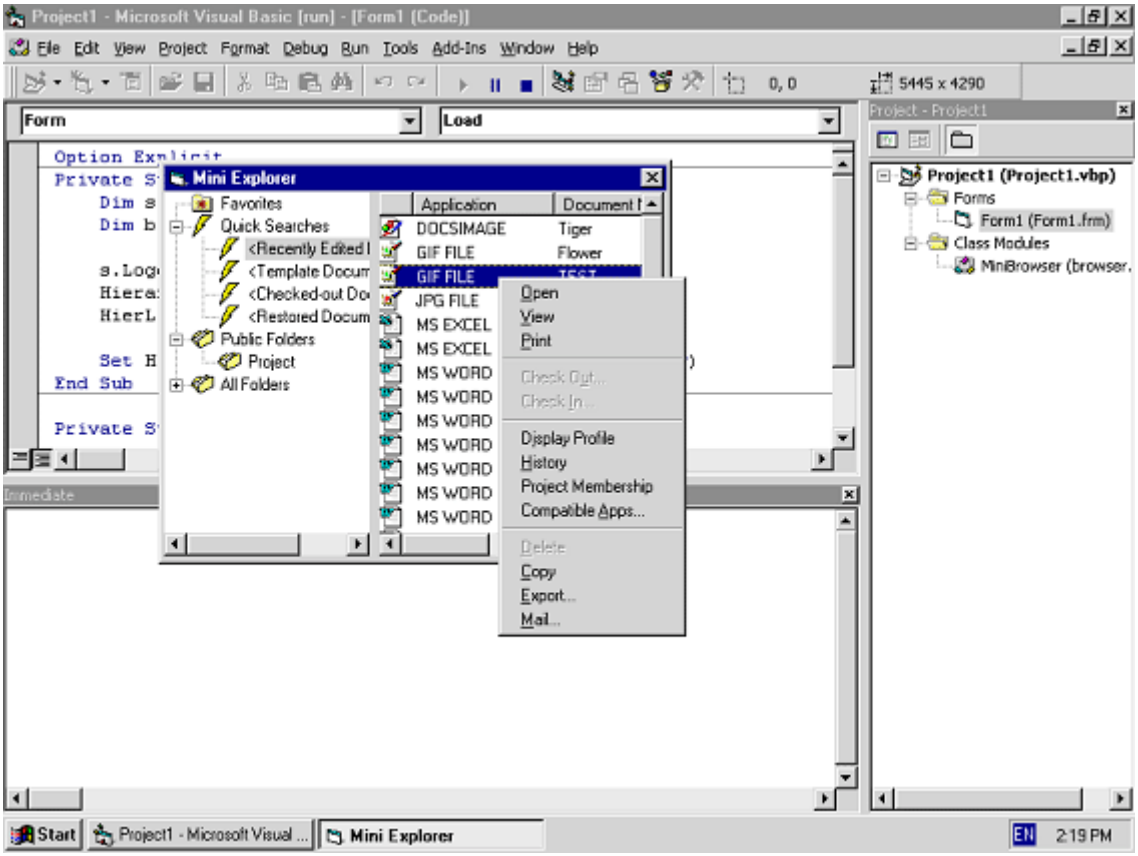

    ' Log to the Explorer Core
    s.Logon r


    ' Specify SessionIDs for the controls
    Hierarchy1.SID = s.SID
    HierList1.SID = s.SID


    ' we want to display all root objects in the hierarchy control
    Set Hierarchy1.RootEnum = s.EnumChildObjects(Nothing)
End Sub


Private Sub Hierarchy1_SelectionChanged()
    ' every time selection changes we should inform Contents
control about that
    HierList1.SetObject Hierarchy1.Selection
End Sub_
```

The following picture shows how our Mini Explorer may look:



Properties

Name	Description
Hierarchy::Appearance	The property returns or sets the value whether or not a control is painted at run time with 3-D effects.
Hierarchy::BackColor	The property returns or sets the background color of the control.
Hierarchy::BorderStyle	The property returns or sets the border style for an object.
Hierarchy::ContextMenuEnabled	The property returns or sets the value that determines whether or not a control enables or disables all context menu related operations. Also it controls keyboard shortcuts processing (enabled or disabled).
Hierarchy::Enabled	The property returns or sets a value that determines whether the control can respond to user-generated events.
Hierarchy::Filter	The property returns or sets the filter string used to retrieve tree objects. The property contains the list of object's class with separator ";". The show of objects is defined by property FilterMode
Hierarchy::FilterMode	The property returns or sets how the filter is applied.
Hierarchy::Font	The property returns or sets the font for the text of the control.
Hierarchy::ForeColor	The property returns or sets the foreground color used to display text.
Hierarchy::FullPath	The property returns or sets the fully qualified name of the current object.

Name	Description
Hierarchy::HideSelection	The property returns or sets a value that determines whether the selected item will be displayed as selected when the tree loses focus.
Hierarchy::hWnd	The property returns a handle of the object's window.
Hierarchy::Indentation	The property returns or sets the indention of the child items relatively to their parents.
Hierarchy::LineStyle	The property returns or sets the style of lines displayed between tree nodes.
Hierarchy::OLEDragMode	The property returns or sets the value whether this control can act as an OLE drag/drop source, and whether this process is started automatically or under programmatic control. This property is standard property of controls.
Hierarchy::Root	The property returns or sets the root object used to retrieve child nodes.
Hierarchy::Selection	The property is used to obtain current selection.
Hierarchy::SID	The property used to identify the owner's session to the Explorer Core.
Hierarchy::Style	The property returns or sets the value that determines how a hierarchical list of nodes is displayed.
Hierarchy::UIContext	The property returns or sets Control's UI context.

Methods

Name	Description
Hierarchy::BrowseObject	The method allows to programmatically change the currently selected item in the tree. Path parameter is a full path to the desired object (inclusive).
Hierarchy::Refresh	Forces a complete repaint of a control.

Events

Name	Description
Hierarchy::ObjectDblClicked	Occurs when object was double clicked in the control.
Hierarchy::SelectionChanged	Indicates that selection has changed. Current selection can be obtained from the Selection property.

Hierarchy::Appearance

The property returns or sets the value whether or not a control is painted at run time with 3-D effects.

Syntax

`object.Appearance [= AppearanceConstants]`

AppearanceConstants

This property can be:

Constant	Value	Description
ccFlat	0	There isn't painted 3D effect
cc3D	1	There is painted 3D effect

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
Hierarchy1. Appearance = cc3D
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSTree.ocx"

See Also

"Hierarchy Control Reference: Hierarchy" on page 713

Hierarchy::BackColor

The property returns or sets the background color of the control.

Syntax

`object.BackColor [= color]`

`Color`

An OLE_COLOR value or constant that determines the background colors of an object.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
Hierarchy1. BackColor = vbButtonText
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSTree.ocx"

See Also

"Hierarchy Control Reference: Hierarchy" on page 713

Hierarchy::BorderStyle

The property returns or sets the border style for an object.

Syntax

```
object.BorderStyle [= BorderStyleConstants]
BorderStyleConstants
```

This property can be:

Constant	Value	Description
ccNone	0	None (no border or border-related elements)
ccFixedSingle	1	Fixed Single.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
Hierarchy1.BorderStyle= ccNone
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSTree.ocx"

See Also

"Hierarchy Control Reference: Hierarchy" on page 713

Hierarchy::BrowseObject

The method allows to programmatically change the currently selected item in the tree. The path parameter is a full path to the desired object (inclusive).

Syntax

```
object.BrowseObject path [,bRelativ]
```

Properties

path

Required. The string value is a full path to the desired object.

bRelativ

Optional. A Boolean value, which defines or not a path from the control's root.

Return Values

The value of the string type consisting of Object IDs, delimited by the "/" character.

Return Values

The control should use the "ID" property of IBCObject.

If bRelative = FALSE then path defines a path from the control's root (an object on the topmost level, then one of it's children and so on), otherwise – from the currently selected item (its sub-item and so on). The control attempts to open nodes in the requested order (as specified in the path parameter). After completion, the element pointed to by the path parameter becomes selected. If (at some level) an element corresponding the path cannot be found, the control will not attempt to go further into the hierarchy. Instead, the last accessible element becomes selected. BrowseObject fires SelectionChanged event. A script can check the success of the operation by comparing the value of the control's path property with the path parameter, passed to the BrowseObject method.

Example

(Visual Basic 5.0)

```
Private Sub ...
. . .
If Flags = 1 Then
' Open in the same window with relative path
strNewPath = Hier.BrowseObject(path, True)
Else
' Open in the same window with absolute path
Hier.BrowseObject path, False
End if
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSTree.ocx"

See Also

"Hierarchy Control Reference: Hierarchy" on page 713

Hierarchy::ContextMenuEnabled

The property returns or sets the value that determines whether or not a control enables or disables all context menu related operations. Also it controls keyboard shortcuts processing (enabled or disabled).

Syntax

```
object.ContextMenuEnabled [= Boolean]
```

Boolean

This property returns True (Default) if all context menu is enabled, else False.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
Hierarchy1.ContextMenuEnabled= False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSTree.ocx"

See Also

"Hierarchy Control Reference: Hierarchy" on page 713

Hierarchy::Enabled

The property returns or sets a value that determines whether the control can respond to user-generated events.

Syntax

```
object.Enabled [= Boolean]
```

Boolean

The property determines whether an object responds to the events (True) or prevents a control from responding to the events (False).

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
Hierarchy1.Enabled= False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSTree.ocx"

See Also

"Hierarchy Control Reference: Hierarchy" on page 713

Hierarchy::Filter

The property returns or sets the filter string used to retrieve tree objects. The property contains the list of object's class with the separator ";". The show of objects is defined by property FilterMode.

Syntax

```
object.Filter [= String]
String
```

A string value contains a filter. It is formed by the control.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim strFilter as String
strFilter = Hierarchy1.FilterEnd Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSTree.ocx"

See Also

"Hierarchy Control Reference: Hierarchy" on page 713

Hierarchy::FilterMode

The property returns or sets how the filter is applied.

Syntax

```
object.FilterMode [= FilterModeConstants]
FilterModeConstants
```

This property can be:

Constant	Value	Description
ccIgnore	0	Ignore filter
ccAccept	1	Accept filter
ccReject	2	Reject filter

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
Hierarchy1.FilterMode= ccAccept
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSTree.ocx"

See Also

"Hierarchy Control Reference: Hierarchy" on page 713

Hierarchy::Font

The property returns or sets the font for the text of the control.

Syntax

```
object.Font
StdFont
```

A StdFont value determines the font for the text.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
Hierarchy1.Font.Bold = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSTree.ocx"

See Also

"Hierarchy Control Reference: Hierarchy" on page 713

Hierarchy::ForeColor

The property returns or sets the foreground color used to display the text.

Syntax

```
object.ForeColor [= color]
Color
```

An OLE_COLOR value or constant determines the foreground color of the text.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
Hierarchy1.ForeColor = vbHighLight
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSTree.ocx"

See Also

"Hierarchy Control Reference: Hierarchy" on page 713

Hierarchy::FullPath

The property returns or sets the fully qualified name of the current object.

Syntax

```
object.FullPath [= string]
```

String

A string value contains fully qualified name. The string contains the list of objects with the separator "/". It is formed by the control.

Access

Read-Write

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSTree.ocx"

See Also

"Hierarchy Control Reference: Hierarchy" on page 713

Hierarchy::HideSelection

The property returns or sets a value that determines whether the selected item will be displayed as selected when the tree loses focus.

Syntax

```
object.HideSelection [= boolean]
```

Boolean

This property is False (default).

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
Hierarchy1.HideSelection= True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSTree.ocx"

See Also

"Hierarchy Control Reference: Hierarchy" on page 713

Hierarchy::hWnd

The property returns a handle of the object's window.

Syntax

```
object.hWnd
```

A numeric (long) value contains a handle.

Access

Read-Only

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim Hndl as Long
Hndl = Hierarchy1.hWndEnd Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSTree.ocx"

See Also

"Hierarchy Control Reference: Hierarchy" on page 713

Hierarchy::Indentation

The property returns or sets the indention of the child items relatively to their parents.

Syntax

```
object.Indentation [= single]
```

single

The width of indentation.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...

Hierarchy1.Indentation= 300

End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSTree.ocx"

See Also

"Hierarchy Control Reference: Hierarchy" on page 713

Hierarchy::LineStyle

The property returns or sets the style of lines displayed between tree nodes.

Syntax

```
object.LineStyle [= TreeLineStyleConstants]
```

TreeLineStyleConstants

This property can be:

Constant	Value	Description
tvwTreeLines	0	The tree view control has lines linking child items to their corresponding parent items
tvwRootLines	1	The tree view control has lines linking child items to the root of the hierarchy

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
Hierarchy1.LineStyle= tvwRootLines
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSTree.ocx"

See Also

"Hierarchy Control Reference: Hierarchy" on page 713

Hierarchy::ObjectDblClicked

Occurs when an object is double clicked in the control.

Syntax

```
Private Sub object_ObjectDblClicked (Object)
```

Object

This parameter is an object of the IBCObject type.

Example

(Visual Basic 5.0)

```
Private Sub Hierarchy1_ObjectDblClicked(Object As IBCObject)

End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSTree.ocx"

See Also

"Hierarchy Control Reference: Hierarchy" on page 713

Hierarchy::OLEDragMode

The property returns or sets the value whether this control can act as an OLE drag/drop source, and whether this process is started automatically or under programmatic control. This property is the standard property of controls.

Syntax

`object.OLEDragMode [= OLEDragConstants]`

`OLEDragConstants`

This property can be:

Constant	Value	Description
ccOLEDragAutomatic	1	Automatic
ccOLEDragManual	0	Manual

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
Hierarchy1.OLEDragMode= ccOLEDragAutomatic
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSTree.ocx"

See Also

"Hierarchy Control Reference: Hierarchy" on page 713

Hierarchy::Refresh

Forces a complete repaint of a control.

Syntax

```
object.Refresh
```

Remarks

This is the standard method for all controls.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSTree.ocx"

See Also

"Hierarchy Control Reference: Hierarchy" on page 713

Hierarchy::Root

The property returns or sets the root object used to retrieve child nodes.

Syntax

```
object.Root [= unknown]
```

Unknown

This property is used to specify a root object for the control.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim xbcObject As DECORELib.IBCObject
Set xbcObject = _
CoreSession.ObjectByID(Desktop.VTab.GetActiveTab.Key)
If Not (xbcObject Is Nothing) Then

If Not TreeControl Is Nothing Then
```

```
Set TreeControl.Root = xbcObject
End if

ListControl.Root = xbcObject

If Not TreeControl Is Nothing Then
ListControl.FullPath = TreeControl.FullPath
Else
ListControl.FullPath = ""
End if

. . .
End if
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSTree.ocx"

See Also

"Hierarchy Control Reference: Hierarchy" on page 713

Hierarchy::Selection

The property is used to obtain the current selection.

Syntax

```
object.Selection
```

This property contains the current selection - an object of the IBCObject type.

Access

Read-Write

Remarks

A hierarchy control does not support multiple selections.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSTree.ocx"

See Also

"Hierarchy Control Reference: Hierarchy" on page 713

Hierarchy::SelectionChanged

Indicates that selection has changed. The current selection can be obtained from theSelection property.

Syntax

```
Private Sub object_SelectionChanged (Object)
```

Object

This parameter is an object of the IBCObject type.

Example

(Visual Basic 5.0)

```
Private Sub Hierarchy1_SelectionChanged(Object As IBCObject)


End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSTree.ocx"

See Also

"Hierarchy Control Reference: Hierarchy" on page 713

Hierarchy::SID

The property used to identify the owner's session to the Explorer Core.

Syntax

```
object.SID [= string]
```

String

Identifies the owner's session.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim a As New DOCSObjects.Application
Dim b As New BCSession   'Explorer Core Session
Dim DST As String

DST = a.DST     'We should use DST in code for login

Hierarchy1.SID = b.SID

Hierarchy1.Root = b.EnumChildObjects(Nothing)
'Initialize Root of the control
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSTree.ocx"

See Also

"Hierarchy Control Reference: Hierarchy" on page 713

Hierarchy::Style

The property returns or sets the value that determines how a hierarchical list of nodes is displayed.

Syntax

`object.Style [= TreeStyleConstants]`

`TreeStyleConstants`

This property can be:

Constant	Value	Description
tvwPictureText	0	Has not the TVS_HASBUTTONS and TVS_HASLINES styles
tvwPlusMinusPictureText	1	Has not the TVS_HASLINES style but has the TVS_HASBUTTONS style

Constant	Value	Description
tvwTreelinesPictureText	2	Has not the TVS_HASBUTTONS style but has the TVS_HASLINES style
tvwTreelinesPlusMinusPictureText	3	Has the TVS_HASBUTTONS and TVS_HASLINES styles

Remarks

TVS_HASBUTTONS - The tree view control adds a button to the left of each parent item.

TVS_HASLINES - The tree view control has lines linking child items to their corresponding parent items.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
Hierarchy1.Style= tvwTreelinesPictureText
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSTree.ocx"

See Also

"Hierarchy Control Reference: Hierarchy" on page 713

Hierarchy::UIContext

The property returns or sets control's UI context.

Syntax

```
object.UIContext [= string]
String
```

The control's UI context.

Access

Read-Write

Remarks

“Hierarchy” is used only.

Example

(Visual Basic 5.0)

```
Private Sub ...
Set xObject = Hierarchy1(Index).Selection

Dim objClass As String
objClass = xObject.GetProperty(CoreSession.SID,
Hierarchy1(Index).UIContext, "Class")
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSTree.ocx"

See Also

“Hierarchy Control Reference: Hierarchy” on page 713

Hierarchy List Control

It is an object that can be used to display the contents of the Explorer Core object (its child objects). The list pane control will be used by the Explorer internally. The control supports all the standard Explorer Core operations - drag-and-drop (including reorder), context menu and keyboard shortcuts. In addition the control supports standard properties, methods, and events described in Visual Basic.

Example – Explorer Core and DMO integration

DMO layer is tightly integrated with the Explorer Core. You can see that from the following example:

```
Private Sub Form_Load()

' Create new Application object
```

```
Dim app As New Application

' Create new Explorer Core Session
Dim session As New BCSession

' Creates a new Browser Object. Explorer Core will use this object
to
' provide feedback to the User Interface
Dim b As New MiniBrowser
Dim r As BCBrowser
Set r = b

' Log to the Explorer Core
session.Logon r

' Create new search object, using def_qbe form
Dim s As search
Set s = app.CurrentLibrary.CreateSearch("def_qbe")

' Display Edit Criteria Form for that object
s.ShowEditCriteriaForm True

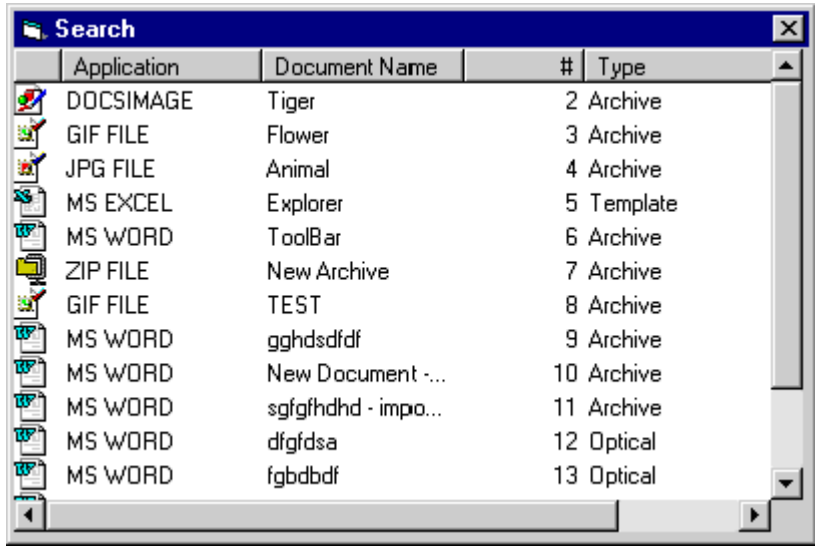
' Specify SessionIDs for the Contents control
HierList1.SID = session.SID

' Specify Root object for Contents pane
HierList1.SetObject s

' Force control to retrieve its contents (Searches have
SlowRetrieve property set to True)
HierList1.Retrieve

End Sub_
```

The program results look as follows:



Hierarchy List Control Reference:HierList

Properties

Name	Description
HierList::Appearance	The property returns or sets the value whether or not a control is painted at run time with 3-D effects.
HierList::BackColor	The property returns or sets the background color of the control.
HierList::BorderStyle	The property returns or sets the border style for an object.
HierList::CanRestoreColumnDefaults	The property returns the value whether or not a user can restore its column settings to defaults
HierList::CanSaveColumnDefaults	The property returns the value whether or not a user can save its column settings to defaults
HierList::ContextMenuEnabled	The property returns or sets the value whether all context menu related operations are enabled or disabled. Also controls keyboard shortcuts processing is enabled or disabled by this property.
HierList::Enabled	The property returns or sets a value that determines whether the control can respond to user-generated events.

Name	Description
HierList::Filter	The property returns or sets the filter string used to retrieve tree objects. The property contains the list of object's class with the separator ";". The show of objects is defined by property FilterMode.
HierList::FilterMode	The property returns or sets how the filter is applied.
HierList::Font	The property returns or sets the font for the text of the control.
HierList::ForeColor	The property returns or sets the foreground color used to display text.
HierList::FullPath	The property returns or sets the fully qualified name of the current object.
HierList::HeaderMenuEnabled	The property returns or sets a value that determines whether the column headers context menu will be displayed when a column header is right clicked.
HierList::HideColumnHeaders	The property returns or sets whether or not the control's column headers are hidden in Report/Structured view.
HierList::HideSelection	The property returns or sets a value that determines whether the selected item will be displayed as selected when the tree loses focus.
HierList::hWnd	The property returns a handle of the control's window.
HierList::MaxSelectionSize	The property returns or sets a value that determines whether a user can make multiple selections in a control.
HierList::NoChildrenText	The property returns or sets a text, which is shown when objects are absent.
HierList::ObjectCount	The property returns the number of the items in the List.
HierList::OLEDragMode	The property returns or sets the value whether this control can act as an OLE drag/drop source, and whether this process is started automatically or under programmatic control. This property is standard property of controls.
HierList::PopupMenuEnabled	The property returns or sets the value that determines whether the control's context menu will be displayed when the list is right clicked.

Name	Description
HierList::PreviewCommands	The property returns or sets the value that determines weather default command from context menu will be run. If this property is set to True and default command will be selected from context menu the PreviewCommands event will be fired and command will not be run.
HierList::PreviewDoubleClick	The property returns or sets the value that determines weather double click or Enter should be processed as default command. If this property set to True and double click or Enter is determined the default command will not be run but the PreviewCommands event will be fired.
HierList::ReorderEnabled	The property returns or sets a value that determines whether the List will be reordered after drag-and-drop operations.
HierList::RetrievingText	The property returns or sets a text, which is shown when objects are retrieved.
HierList::Root	The property returns or sets the root object used to retrieve child nodes.
HierList::Selection	The property is used to obtain the current selection.
HierList::SID	The property used to identify the owner's session to the Explorer Core.
HierList::SlowRetrieveText	The property returns or sets text, which is shown when object is retrieved.
HierList::SortingEnabled	The property returns or sets a value that indicates whether the control's items will be sorted alphabetically when a user clicks a column header.
HierList::UIContext	The property returns or sets control's UI context.
HierList::View	The property returns or sets the value of the control's current view.
HierList::ViewAvailable	The property returns available list views for the current root object.

Methods

Name	Description
HierList::Cancel	Cancels a retrieving operation.
HierList::CustomizeStructuredView	Displays the Customize Structured View form.

Name	Description
HierList::EditColumns	Displays the dialog used to customize the list columns.
HierList::PageSetup	Displays page setup dialog.
HierList::PrintOut	Print the list contents.
HierList::Refresh	Forces a complete repaint of a control.
HierList::RestoreColumnDefaults	The control restores default columns for the user by using RestoreDefaults() method on the current BCPropertyArray object.
HierList::Retrieve	Fetch the object's content even if it has the SlowRetrieveText property.
HierList::RetrieveColumns	Retrieve specified columns during object enumeration.
HierList::SaveColumnDefaults	The control should set currently displayed column headers as default headers for other users.
HierList::SelectAll	Select all items in the list.

Events

Name	Description
HierList::AfterVerb	Occurs after executing command.
HierList::BeforeVerb	Occurs before executing command.
HierList::LastObjectRetrieved	The control fires the ObjectRetrieved event after appropriate notification message from the Explorer Core.
HierList::ObjectsRetrieved	The control fires the ObjectRetrieved event after appropriate notification message from the Explorer Core.
HierList::PreviewCommands	The control fires an event before processing command.
HierList::SelectionChanged	Indicates that a selection has changed. The current selection can be obtained from the Selection property.
HierList::ShowProgressWindow	Shows the window with a progress bar to the user.
HierList::ViewChanged	Fired whenever view mode is changed in the control.

HierList::AfterVerb

Occurs after executing command.

Syntax

```
Private Sub object_AfterVerb ()
```

Example

(Visual Basic 5.0)

```
Private Sub HierList1_AfterVerb ()
. . .
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::Appearance

The property returns or sets the value whether or not the control is painted at run time with 3-D effects.

Syntax

`object.Appearance [= *AppearanceConstants*]`

`AppearanceConstants`

This property can be:

Constant	Value	Description
ccFlat	0	There is not painted 3D effect
cc3D	1	There is painted 3D effect

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
HierList1. Appearance = cc3D
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::BackColor

The property returns or sets the background color of the control.

Syntax

`object.BackColor [= color]`

Color

An OLE_COLOR value or constant that determines the background colors of an object.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
HierList1. BackColor = vbButtonText
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::BeforeVerb

Occurs before executing command.

Syntax

```
Private Sub object_BeforeVerb ()
```

Example

(Visual Basic 5.0)

```
Private Sub HierList1_BeforeVerb ()
. . .
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::BorderStyle

The property returns or sets the border style for an object.

Syntax

```
object.BorderStyle [= BorderStyleConstants]
BorderStyleConstants
```

This property can be:

Constant	Value	Description
ccNone	0	None (no border or border-related elements)
ccFixedSingle	1	Fixed Single.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
HierList1.BorderStyle= ccNone
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::Cancel

Cancels a retrieving operation.

Syntax

```
object.Cancel
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::CanRestoreColumnDefaults

The property returns a value whether or not a user can restore its column settings to defaults.

Syntax

```
object.CanRestoreColumnDefaults
```

The property (Boolean) returns TRUE if a user can restore its column settings to defaults, specified by an administrator upon setting DM.

Access

Read-Only

Example

(Visual Basic 5.0)

```
Private Sub ...
If HierList1.CanRestoreColumnDefaults Then. . .
End if
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::CanSaveColumnDefaults

The property returns a value whether or not a user can save its column settings to defaults.

Syntax

```
object.CanSaveColumnDefaults
```

The property (Boolean) returns TRUE if a user can restore its column settings to defaults, specified by an administrator upon setting DM.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
If HierList1.CanSaveColumnDefaults Then. . .
End if
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::ContextMenuEnabled

The property returns or sets the value whether all context menu related operations are enabled or disabled. Also control keyboard shortcuts processing is enabled or disabled by this property.

Syntax

```
object.ContextMenuEnabled [ = boolean]
```

Boolean

This property returns True (Default) if all context menu is enables, else False.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
HierList1.ContextMenuEnabled = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::CustomizeStructuredView

Displays the Customize Structured View form.

Syntax

```
object.CustomizeStructuredView
```

Remarks

Only possible if the current object supports Structured View.

Example

(Visual Basic 5.0)

```
Private Sub ...
HierList1.CustomizeStructuredViewEnd Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::EditColumns

Displays the dialog used to customize the list columns.

Syntax

```
object.EditColumns
```

Example

(Visual Basic 5.0)

```
Private Sub ...
HierList1(Index).EditColumnsEnd Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::Enabled

The property returns or sets a value that determines whether the control can respond to user-generated events.

Syntax

```
object.Enabled [ = boolean]
```

Boolean

The property determines whether an object responds to the events (True) or prevents the control from responding to events (False).

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
HierList1.Enabled = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::Filter

The property returns or sets the filter string used to retrieve tree objects. The property contains the list of object's class with the separator ";". The show of objects is defined by the property FilterMode.

Syntax

```
object.Filter [ = string]
```

String

A string value contained the filter is a line. It is formed by the control.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim strFilter as String
strFilter = HierList1.FilterEnd Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::FilterMode

The property returns or sets how the filter is applied.

Syntax

```
object.FilterMode [ = FilterModeConstants]
```

FilterModeConstants

This property cab be:

Constant	Value	Description
ccIgnore	0	Ignore filter
ccAccept	1	Accept filter
ccReject	2	Reject filter

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
HierList1.FilterMode = ccAccept
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::Font

The property returns or sets the font for the text of the control.

Syntax

```
object.Font [ = StdFont]
```

StdFont

A StdFont value determines the font for the text.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
HierList1.Font.Bold = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::ForeColor

The property returns or sets the foreground color used to display the text.

Syntax

```
object.ForeColor [ = color]
```

Color

An OLE_COLOR value or constant determines the foreground color of the text.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
HierList1.ForeColor = vbHighLight
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::FullPath

The property returns or sets the fully qualified name of the current object.

Syntax

```
object.FullPath [ = string]
```

String

A string value contains the fully qualified name. The string contains the list of objects with the separator "/". It is formed by the control.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim xbcObject As DECORELib.IBCObject
Set xbcObject = _
CoreSession.ObjectByID(Desktop.VTab.GetActiveTab.Key)
If Not (xbcObject Is Nothing) Then

If Not TreeControl Is Nothing Then
Set TreeControl.Root = xbcObject
End if

ListControl.Root = xbcObject
If Not TreeControl Is Nothing Then
ListControl.FullPath = TreeControl.FullPath
Else
ListControl.FullPath = ""
End if

. . .
End if
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::HeaderMenuEnabled

The property returns or sets a value that determines whether the column header context menu will be displayed when a column header is right clicked.

Syntax

```
object.HeaderMenuEnabled [ = boolean]
```

Boolean

This property is True by default.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
HierList1.HeaderMenuEnabled = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::HideColumnHeaders

The property returns or sets whether or not the control's column headers are hidden in the Report/Structured view.

Syntax

```
object.HideColumnHeaders [ = boolean]
```

Boolean

This property is False by default.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
HierList.HideColumnHeaders = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::HideSelection

The property returns or sets a value that determines whether the selected item will be displayed as selected when the tree loses focus.

Syntax

```
object.HideSelection [ = boolean]
Boolean
```

This property is False by default.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
HierList.HideSelection = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::hWnd

The property returns a handle of the control's window.

Syntax

```
object.hWnd
```

A numeric (long) value contains a handle.

Access

Read-Only

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim Hndl as Long
Hndl = HierList1.hWndEnd Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::LastObjectRetrieved

The control fires the ObjectRetrieved event after the appropriate notification message from the Explorer Core.

Syntax

```
Private Sub object_LastObjectRetrieved(Reason, Count)
```

Reason

This parameter of the integer type can be a value from the following set:

Value	Description
0	Cancelled by a user
1	Everything is OK

Count

This parameter will be fired before any objects will be retrieved by the control with the Count = 0, so the script can perform necessary initialization.

Example

(Visual Basic 5.0)

```
Private Sub HierList1_LastObjectRetrieved (Reason As Integer, _
Count As Long)


. . .
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::MaxSelectionSize

The property returns or sets a value that determines whether a user can make multiple selections in a control.

Syntax

```
object.MaxSelectionSize [ = size]
```

size

A numeric (long) expression specifies the setting multiple selections (1) or single selection (0).

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim whatControlType as Long
HierList1.MaxSelectionSize = 1
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::NoChildrenText

The property returns or sets a text, which is shown when objects are absent.

Syntax

```
object.NoChildrenText [ = string]
```

String

The shown text.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
HierList1.NoChildrenText = "Objects are absent"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::ObjectCount

The property returns the number of the items in the List.

Syntax

```
object.ObjectCount
```

Access

Read-Only

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim CountItems As Long
CountItems = HierList1.ObjectCountEnd Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::ObjectsRetrieved

The control fires the ObjectsRetrieved event after the appropriate notification message from the Explorer Core.

Syntax

```
Private Sub object_ObjectsRetrieved(Count)
```

Count

This parameter will be fired before any objects will be retrieved by the control with the Count = 0, so the script can perform necessary initialization.

Example

(Visual Basic 5.0)

```
Private Sub HierList1_ObjectsRetrieved (Count As Long)
. . .
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

“Hierarchy List Control Reference:HierList” on page 735

HierList::OLEDragMode

The property returns or sets the value whether this control can act as an OLE drag/drop source, and whether this process is started automatically or under programmatic control. This property is standard property of controls.

Syntax

```
object.OLEDragMode [ = OLEDragConstants]
OLEDragConstants
```

This property can be:

Constant	Value	Description
ccOLEDragAutomatic	1	Automatic
ccOLEDragManual	0	Manual

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
HierList1.OLEDragMode = ccOLEDragAutomatic
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

“Hierarchy List Control Reference:HierList” on page 735

HierList::PageSetup

Displays page setup dialog.

Syntax

```
object.PageSetup
```

Example

(Visual Basic 5.0)

```
Private Sub ...
HierList1.PageSetupEnd Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::PopupMenuEnabled

The property returns or sets a value that determines whether the control's context menu will be displayed when the list is right clicked.

Syntax

```
object.PopupMenuEnabled [ = boolean]
```

Boolean

This property is True by default.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
HierList1.PopupMenuEnabled = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::PreviewCommands

The property returns or sets a value that determines weather the default command from the context menu will be run. If this property sets True and the default command will be selected from the context menu the PreviewCommands event will be fired and the command will not be run.

Syntax

```
object.PreviewCommands [ = boolean]
```

Boolean

This property is False by default.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
HierList1.PreviewCommands = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::PreviewCommands

The control fires an event before processing the command.

Syntax

```
Private Sub object_PreviewCommands(Verb)
```

Verb

This parameter of the string type is the command name.

Example

(Visual Basic 5.0)

```
Private Sub HierList1_PreviewCommands (Verb As String)
```

```
. . .
If Verb = "View" Then
. . .
End If
. . .
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::PreviewDoubleClick

The property returns or sets a value that determines weather double click or Enter should be processed as the default command. If this property sets True and double click or Enter is determined the default command will not be run but the PreviewCommands event will be fired.

Syntax

```
object.PreviewDoubleClick [ = boolean]
```

Boolean

This property is False by default.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
HierList1.PreviewDoubleClick = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::PrintOut

Print the list contents.

Syntax

```
object.PrintOut
```

Example

(Visual Basic 5.0)

```
Private Sub ...
HierList1.PrintOutEnd Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::Refresh

Forces a complete repaint of the control.

Syntax

```
object.Refresh
```

Remarks

This is standard method for all controls.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::ReorderEnabled

The property returns or sets a value that determines whether the List will be reordered after Drag & Drop operations.

Syntax

```
object.ReorderEnabled [ = boolean]
```

Boolean

This property is True by default.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
HierList1.ReorderEnabled = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::RestoreColumnDefaults

The control restores default columns for the user by using the RestoreDefaults() method on the current BCPropertyArray object.

Syntax

```
object.RestoreColumnDefaults
```

Example

(Visual Basic 5.0)

```
Private Sub ...
HierList1.RestoreColumnDefaultsEnd Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::Retrieve

Fetches the object's content even if it has the SlowRetrieveText property.

Syntax

```
object.Retrieve
```

Example

(Visual Basic 5.0)

```
Private Sub ...
HierList1.RetrieveEnd Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::RetrieveColumns

Retrieves specified columns during object enumeration.

Syntax

```
object.RetrieveColumns lpszColumns
String
```

Columns to be retrieved.

Example

(Visual Basic 5.0)

```
Private Sub ...
HierList1.RetrieveColumns "DOCNAME"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::RetrievingText

The property returns or sets a text, which is shown when objects are retrieved.

Syntax

```
object.RetrievingText [ = string]
```

String

The shown text.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
HierList1.RetrievingText = " Please wait, objects " & _
"are being retrieved... "
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::Root

The property returns or sets the root object used to retrieve child nodes.

Syntax

```
object.Root [ = unknown]
```

Unknown

This property is used to specify a root object for the control.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
. . .
HierList1(Index).Root = Hierarchy1(Index).Selection
HierList1(Index).FullPath = Hierarchy1(Index).FullPath
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::SaveColumnDefaults

The control should set currently displayed column headers as default headers for other users.

Syntax

```
object.SaveColumnDefaults
```

Remarks

Available only if appropriate flag was set on BCPropertyArray for the currently displayed object.

Example

(Visual Basic 5.0)

```
Private Sub ...
HierList1.SaveColumnDefaultsEnd Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::SelectAll

Select all items in the list.

Syntax

```
object.SelectAll
```

Example

(Visual Basic 5.0)

```
Private Sub ...
HierList1.SelectAllEnd Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::Selection

The property is used to obtain the current selection.

Syntax

```
object.Selection [ = ObjectArray]
ObjectArray
```

This property contains current selection - object of the BCObjectArray type.

Access

Read-Write

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::SelectionChanged

Indicates that selection has changed. The current selection can be obtained from the Selection property.

Syntax

```
Private Sub object_SelectionChanged(Object)
```

Object

This parameter is an object of the BCObjectArray type.

Example

(Visual Basic 5.0)

```
Private Sub HierList1_SelectionChanged (Object As IBCObject)
. . .
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::ShowProgressWindow

Shows the window with a progress bar to the user.

Syntax

```
Private Sub object_ShowProgressWindow(bShow)
```

bShow

If this parameter is TRUE the script should show a window with a progress bar to the user (with the Cancel button) else the script should hide this window.

Example

(Visual Basic 5.0)

```
Private Sub HierList1_ShowProgressWindow (bShow As Boolean)
. . .
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::SID

The property used to identify the owner's session to the Explorer Core.

Syntax

```
object.SID [ = string]
String
```

Identifies the owner's session.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim a As New DOCSObjects.Application
Dim b As New BCSession     'Explorer Core Session
Dim DST As String

DST = a.DST                'We should use DST in code for login

HierList1.SID = b.SID

HierList1.Root = b.EnumChildObjects(Nothing)
'Initialize Root of the control
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

“Hierarchy List Control Reference:HierList” on page 735

HierList::SlowRetrieveText

The property returns or sets a text, which is shown when an object is retrieved.

Syntax

```
object.SlowRetrieveText [ = string]
```

String

The text, which is shown.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
HierList1.SlowRetrieveText = "Object is retrieved"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

“Hierarchy List Control Reference:HierList” on page 735

HierList::SortingEnabled

The property returns or sets a value that indicates whether the control's items will be sorted alphabetically when a user clicks a column header.

Syntax

```
object.SortingEnabled [ = boolean]
```

Boolean

This property is True by default.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
HierList1.SortingEnabled = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

“Hierarchy List Control Reference:HierList” on page 735

HierList::UIContext

The property returns or sets a value that indicates whether the control's items will be sorted alphabetically when a user clicks a column header.

Syntax

```
object.UIContext [ = string]
```

String

The control's UI context.

Access

Read-Write

Remarks

“contents” is set by default. Also this control can support "contents.SV" and "versions". These strings is not used now.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

“Hierarchy List Control Reference:HierList” on page 735

HierList::View

The property returns or sets the value of the control's current view.

Syntax

```
object.View [ = HierarchyViewConstants]
HierarchyViewConstants
```

This property can be:

Constant	Value	Description
hvwReport	1	The current view is Report
hvwList	3	The current view is List
hvwStructured	4	The current view is Structured

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
HierList1.View = hvwList
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::ViewAvailable

The property returns the list of available views for the current root object.

Syntax

```
object.ViewAvailable
```

This property (long) can be:

Constant	Value	Description
hvwReport	1	Report view
hvwList	3	List view

Constant	Value	Description
hvwStructured	4	Structured view

Access

Read-Only

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim typeView as Long
TypeView = HierList1.ViewAvailableEnd Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

HierList::ViewChanged

Fired whenever the view mode is changed in the control.

Syntax

```
Private Sub object_ViewChanged ()
```

Example

(Visual Basic 5.0)

```
Private Sub HierList1_ViewChanged ()
. . .
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DOCSList.ocx"

See Also

"Hierarchy List Control Reference:HierList" on page 735

FavoritesBar Control

This control realizes Favorites Bar. It contains some folders, on which some Items (images) settle down. Images are connected with DM Extensions objects, which can be activated if clicked. The control supports two types of items:

- BCObjects. These objects implement the IBCObject, IBCDragDropSupport interfaces, so the control may handle them automatically.
- User defined objects. - COM objects that do not implement interfaces mentioned above. This control contains 5 objects: FavoritesBar, Folder, FolderItem, FolderItems, and Folders.

All items that can be displayed in the control are grouped into folders. The control allows a developer to specify the maximum number of folders that can be created.

Folders contain items that have their own properties, defining their appearance and behavior.

Items can be of two types - Explorer Core objects and User-defined objects. In most cases, Explorer Core objects can be automatically handled by the control, but for user defined objects a user can add own functionality (via events).

The control supports the context menu. The control of an Explorer Core object uses object-provided menu (using Session property) and adds several control-specific items to it.

For user defined objects the control always notifies its owner that menu should be displayed and owner is responsible for performing an actual menu display operation.

Except described the control supports also standard properties, methods, and events described in Visual Basic.

Favorites Bar Control Reference

- "FavoritesBar" on page 773
- "Folder" on page 797
- "FolderItem" on page 805

- "Folders" on page 811
- "FolderItems" on page 816

FavoritesBar

This is the main control. It supports folders and operations with them and with items.

Properties

Name	Description
FavoritesBar::ActiveFolder	The property returns or sets a current active folder. The property contains the Folder object.
FavoritesBar::AllowHideCommand	The property returns or sets a value that determines whether it is possible to hide FavoritesBar via the context menu.
FavoritesBar::AnimationSpeed	The property returns or sets the value, which determinates a speed of the folder scroll.
FavoritesBar::Appearance	The property returns or sets the value whether or not a control is painted at run time with 3-D effects.
FavoritesBar::BackColor	The property returns or sets the background color of the FavoritesBar control.
FavoritesBar::ContextMenuMode	The property returns or sets the value that determines the menu mode.
FavoritesBar::Enabled	The property returns or sets a value that determines whether the control can respond to user-generated events.
FavoritesBar::FolderBackColor	The property returns or sets the background color of the Folder objects.
FavoritesBar::FolderFont	The property returns or sets a font for Folder objects.
FavoritesBar::FolderForeColor	The property returns or sets the foreground color used to display text for the Folder objects.
FavoritesBar::Folders	The property contains the Folders collection object.
FavoritesBar::Font	The property returns or sets a font for the FavoritesBar object.
FavoritesBar::ForeColor	The property returns or sets the foreground color used to display text for the FavoritesBar object.

Name	Description
FavoritesBar::hWnd	The property returns a handle (from Microsoft Windows) of the control's window.
FavoritesBar::MaxFolderNumber	The property returns or sets a value, which determines the maximum count of the folders in the FavoritesBar.
FavoritesBar::OLEDropMode	The property returns or sets the value whether this control can act as an OLE drag-and-drop source, and whether this process is started automatically or under programmatic control. This control property is standard one.
FavoritesBar::Session	The property returns or sets a value that contains IBCSession object.
FavoritesBar::UIContext	The property returns or sets control's UI context.
FavoritesBar::View	The property returns or sets the value of the control's current view.

Methods

Name	Description
FavoritesBar::HitTest	The method returns the FavoritesBar object by position, returned from OLEDragOver, OLEDragDrop.
FavoritesBar::Init	This method initializes the control.
FavoritesBar::SaveSettings	The method saves properties of FavoritesBar by calling the Write method of the PropertyBag interface, passed to the control in the Init method.

Events

Name	Description
FavoritesBar::FavoritesBarContextMenu	This event fired on right mouse click on a folder title. In the manual context menu mode a client is responsible for displaying and handling context menu in the event procedure.
FavoritesBar::FolderChange	Occurs when the folder was changed.

Name	Description
FavoritesBar::FolderContextMenu	This event fired on right mouse click on a folder title. In the manual context menu mode a client is responsible for displaying and handling context menu in the event procedure.
FavoritesBar::Hide	This event is fired on a user selects the "Hide Favorites Bar" command in the control context menu. A client is responsible to actually hide control in the event procedure.
FavoritesBar::ItemClick	This event is fired on mouse click on the item. A client is responsible for handling this event for user defined items.
FavoritesBar::ItemContextMenu	This event is sfired on right mouse click on the item. In the manual context menu mode a client is responsible for displaying and handling context menu for all items and in automatic mode only for user defined items.
FavoritesBar::MouseMove	This event is fired on when mouse is moved over the control.

FavoritesBar::ActiveFolder

The property returns or sets a current active folder. The property contains the Folder object.

Syntax

```
Set object.ActiveFolder [ = Folder ]
```

Folder

The Folder object.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
Folder1 As New Folder


. . .
Set FavoritesBar1.ActiveFolder = Folder1
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FavoritesBar" on page 773

FavoritesBar::AllowHideCommand

The property returns or sets a value that determines whether it is possible to hide FavoritesBar via the context menu.

Syntax

```
object.AllowHideCommand [ = boolean ]
```

Boolean

A Boolean determines whether the command context menu on a folder or empty space contains "Hide Favorites Bar" command.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
FavoritesBar1.AllowHideCommand = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FavoritesBar" on page 773

FavoritesBar::AnimationSpeed

The property returns or sets the value, which determinates a speed of the folder scroll.

Syntax

```
object.AnimationSpeed [ = FAnimationSpeedConstants ]
```

FAnimationSpeedConstants

This property can be:

Constant	Value	Description
Immediate	0	Immediate speed
Fast	1	Fast speed
Medium	2	Medium speed (Default)
Slow	3	Slow speed

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
FavoritesBar1.AnimationSpeed = Fast
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FavoritesBar" on page 773

FavoritesBar::Appearance

The property returns or sets the value, which defines whether or not a control is painted at run time with 3-D effects.

Syntax

```
object.Appearance [ = FViewConstants ]
```

FViewConstants

This property can be:

Constant	Value	Description
View2D	0	Painted without 3D effect
View3D	1	Painted with 3D effect (Default)

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
FavoritesBar1.Appearance = View2D
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FavoritesBar" on page 773

FavoritesBar::BackColor

The property returns or sets the background color of the FavoritesBar control.

Syntax

```
object.BackColor [ = color ]
```

Color

An OLE_COLOR value or constant that determines the background colors of an object.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
FavoritesBar1.BackColor = vbButtonText
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FavoritesBar" on page 773

FavoritesBar::ContextMenuMode

The property returns or sets the value that determines the menu mode.

Syntax

`object.ContextMenuMode [ = FMenuModeConstants ]`

FMenuModeConstants

This property can be:

Constant	Value	Description
MenuManual	1	The mode of the menu is manual
MenuAutomatic	2	The mode of the menu is Automatic (Default)

Access

Read-Write

Remarks

For Explorer Core objects the control uses object-provided menu (using the Session property) and adds several control-specific items to it.

For user defined items the control always notifies its owner that menu should be displayed and owner is responsible for performing the actual menu display operation.

Automatic Context Menu (ContextMenuMode=Automatic)

Folder context menu. Also shown when user clicks the folder contents (not item)	Large icons Small icons Add new group Remove group Rename group Hide Favorites Bar	Not extendable
Item context menu (for Explorer Core items)	Object defined items Remove from Favorites Bar	Object specific items will be obtained from the object itself
User defined objects	User defined menu items	See Note below

Automatic Context Menu (ContextMenuMode=Manual)

Folder context menu	Also shown when user clicks the folder contents (not item)	User defined menu items
See Note below	Item context menu (for Explorer Core items)	User defined menu items
See Note below	User defined objects	User defined menu items
See Note below		

Note

On right click the control generates the ItemContextMenu, FolderContextMenu and FavoritesBarContextMenu events. A user code in the event procedures is responsible for displaying and handling an actual menu.

Example

(Visual Basic 5.0)

```
Private Sub ...
FavoritesBar1.ContextMenuMode = MenuManual
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FavoritesBar" on page 773

FavoritesBar::Enabled

The property returns or sets a value that determines whether the control can respond to user-generated events.

Syntax

```
object.Enabled [ = boolean ]
```

Boolean

A Boolean that determines an object to respond to events (Default) else prevents the control form responding to events (False).

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
FavoritesBar1.Enabled = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FavoritesBar" on page 773

FavoritesBar::FavoritesBarContextMenu

This event is fired on right mouse button click on an empty space. In the manual context menu mode a client is responsible for displaying and handling the context menu in the event procedure.

Syntax

```
Private Sub object_FavoritesBarContextMenu (x, y)
```

x

The value is long type specifies the horizontal coordinate of the mouse pointer

y

The value is long type specifies the vertical coordinate of the mouse pointer

Example

(Visual Basic 5.0)

```
Private Sub FavoritesBar1_FavoritesBarContextMenu (x As Long, _
y As Long)


. . .
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FavoritesBar" on page 773

FavoritesBar::FolderBackColor

The property returns or sets the background color of the Folder objects.

Syntax

```
object.FolderBackColor [ = color ]
```

Color

An OLE_COLOR value or constant that determines the background colors of an object.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
FavoritesBar1.FolderBackColor = vbButtonText
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FavoritesBar" on page 773

FavoritesBar::FolderChange

Occurs when the folder was changed.

Syntax

```
Private Sub object_FolderChange (pFolder)
```

pFolder

The value determinates the changed folder.

For the latest information about this object refer to the Folder object.

Example

(Visual Basic 5.0)

```
Private Sub FavoritesBar1_FolderChange (pFolder As Folder)
. . .
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FavoritesBar" on page 773

FavoritesBar::FolderContextMenu

This event is fired on right mouse button click on a folder title. In the manual context menu mode a client is responsible for displaying and handling the context menu in the event procedure.

Syntax

```
Private Sub object_FolderContextMenu (pFolder, x, y)
```

pFolder

The value determinates the folder.

For the latest information about this object refer to the Folder object.

x

The value is long type specifies the horizontal coordinate of the mouse pointer.

y

The value is long type specifies the vertical coordinate of the mouse pointer.

Example

(Visual Basic 5.0)

```
Private Sub FavoritesBar1_FolderContextMenu (pFolder as Folder, _
x As Long, y As Long)


. . .
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FavoritesBar" on page 773

FavoritesBar::FolderFont

The property returns or sets a Font for Folder objects.

Syntax

```
object.FolderFont
```

StdFont

A StdFont value that determines a Font object.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
```

```
FavoritesBar1.FolderFont.Bold = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FavoritesBar" on page 773

FavoritesBar::FolderForeColor

The property returns or sets the foreground color used to display text for the Folder objects.

Syntax

```
object.FolderForeColor [ = color ]
```

Color

An OLE_COLOR value or constant determines the foreground color of the folder's caption.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
FavoritesBar1.FolderForeColor = vbHighLight
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FavoritesBar" on page 773

FavoritesBar::Folders

The property contains the Folders collection object.

Syntax

```
object.Folders [ = Folders ]
```

Folders

The Folders object.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim Folders1 As New Folders
. . .
Set FavoritesBar1.Folders = Folders1
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FavoritesBar" on page 773

FavoritesBar::Font

The property returns or sets a font forFavoritesBar object.

Syntax

```
object.Font
```

StdFont

A StdFont value that determines a font object.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
FavoritesBar1.Font.Bold = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FavoritesBar" on page 773

FavoritesBar::ForeColor

The property returns or sets the foreground color used to display text for the FavoritesBar object.

Syntax

```
object.ForeColor [ = color ]
```

Color

An OLE_COLOR value or constant determines the foreground color of the text.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
FavoritesBar1.ForeColor = vbHighLight
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FavoritesBar" on page 773

FavoritesBar::Hide

This event is fired when a user selects the "Hide Favorites Bar" command in the control context menu. A client is responsible to actually hide the control in the event procedure.

Syntax

```
Private Sub object_Hide
```

Example

(Visual Basic 5.0)

```
Private Sub FavoritesBar1_Hide ()
. . .
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FavoritesBar" on page 773

FavoritesBar::HitTest

The method returns the FavoritesBar object by position, returned from OLEDragOver, OLEDragDrop.

Syntax

```
object.HitTest [ = x, y, ppFolder, ppItem ]
```

x

The value of the long type specifies the horizontal coordinate.

y

The value of the long type specifies the vertical coordinate.

Return Values

The numeric (the long type) value is index for the item to be inserted at this position. (Only if drop was on empty space between items or on item. In other cases is equal to -1).

ppFolder

The value determinates the folder at this position.

For the latest information about this object refer to the Folder.

ppItem

The value determinates the item at this position (if empty space or folder title then the item is Nothing).

For the latest information about this object refer to the FolderItem.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FavoritesBar" on page 773

FavoritesBar::hWnd

The property returns a handle (from Microsoft Windows) of the control's window.

Syntax

```
object.hWnd
```

A numeric (long type) value contains a handle.

Access

Read-Only

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim HWndl as Long
HWndl = FavoritesBar1.hWndEnd Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FavoritesBar" on page 773

FavoritesBar::Init

This method initializes the control.

Syntax

```
[ Rez = ] object.Init ( lpSession, lpPropertyBag )
lpSession
```

Required. The parameter contains the BCSession object.

lpPropertyBag

Required. The parameter (the unknown data type) contains standard PropertyBag object, which is responsible for saving and retrieving control's properties. The IPropertyBag interface is implemented by a client. The Init method calls the Read method of the PropertyBag object.

Return Values

If the return value is equal to True, the method succeeded. If the return value is False, the method failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FavoritesBar" on page 773

FavoritesBar::ItemClick

This event is fired on mouse button click on the item. A client is responsible for handling this event for user defined items.

Syntax

```
Private Sub object_ItemClick (pItem)
```

pItem

The value determinates the clicked Item.

For the latest information about this object refer to the FolderItem object.

Example

(Visual Basic 5.0)

```
Private Sub FavoritesBar1_ItemClick (pItem As FolderItem)
. . .
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FavoritesBar" on page 773

FavoritesBar::ItemContextMenu

This event is fired on right mouse button click on the item. In the manual context menu mode a client is responsible for displaying and handling the context menu for all items and in the automatic mode only for user defined items.

Syntax

```
Private Sub object_ItemContextMenu (pFolderItem, x, y)
```

pFolderItem

The value is determinates the Item.

For the latest information about this object refer to the FolderItem object.

x

The value of the long type specifies the horizontal coordinate of the mouse pointer.

y

The value of the long type specifies the vertical coordinate of the mouse pointer.

Example

(Visual Basic 5.0)

```
Private Sub FavoritesBar1_ItemContextMenu (pFolderItem as _
FolderItem, x As Long, y As Long)
. . .
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FavoritesBar" on page 773

FavoritesBar::MaxFolderNumber

The property returns or sets a value, which determines the maximum count of the folders in FavoritesBar.

Syntax

```
object.MaxFolderNumber [ = long ]
```

Long

A numeric expression specifies the maximum count of the folders.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim maxCount as Long
maxCount = FavoritesBar1.MaxFolderNumberEnd Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FavoritesBar" on page 773

FavoritesBar::MouseMove

This event is fired on when mouse is moved over the control.

Syntax

```
Private Sub object_MouseMove (Button, Shift, x, y)
```

Button

Mouse button state.

Shift

Keybord state (Ctrl, Shift and Alt key state).

x

The value of the long type specifies the horizontal coordinate of the mouse pointer.

y

The value of the long type specifies the vertical coordinate of the mouse pointer.

Example

(Visual Basic 5.0)

```
Private Sub FavoritesBar1_MouseMove (Button As _
Integer, Shift As Integer, x As Long, y As Long)
. . .
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FavoritesBar" on page 773

FavoritesBar::OLEDropMode

The property returns or sets the value whether this control can act as an OLE drag-and-drop source, and whether this process is started automatically or under programmatic control. This property is the standard property of controls.

Syntax

```
object.OLEDropMode [ = FDragDropModeConstants ]
FDragDropModeConstants
```

This property can be:

Constant	Value	Description
OLEDropNone	0	None
OLEDropManual	1	Manual
OLEDropAutomatic	2	Automatic

OLEDropMode=OLEDropAutomatic:

Drag-and-drop operations with BCObjects carried out for BCObject will be processed by the control itself. In this case the key value of an item is taken from the ID property of IBCObject. ToolTip is a document name (a folder or Quick Search name). FavoriteBar

automatically adds an item into the folder. Drag-and-drop on folder's empty space is processed by the control for BCObjects only.

All drag-and-drop operations for user defined objects are processed via events. If an object is User Defined, the FavoriteBar control generates the OLEDragOver and OLEDragDrop event. A user code in these event procedures is responsible for carrying out the drag-and-drop operation and for calling the Add method for the appropriate FolderItems collection.

OLEDropMode=OLEDropManual:

All drag-and-drop operations are processed via events.

The control generates the OLEDragOver and OLEDragDrop event. A user code in these event procedures is responsible for accepting Drag and for calling Add method for appropriate FolderItems collection.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
FavoritesBar1.OLEDropMode = OLEDropAutomatic
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FavoritesBar" on page 773

FavoritesBar::SaveSettings

The method saves properties of FavoritesBar by calling the Write method of the PropertyBag interface, passed to the control in the Init method.

Syntax

```
object.SaveSettings
```

Return Values

If the return value (the Boolean type) is equal True, the method succeeded. If the return value is False, the method failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FavoritesBar" on page 773

FavoritesBar::Session

The property returns or sets a value that contains the IBCSession object.

Syntax

```
object.Session [ = IBCSession ]
```

IBCSession

The IBCSession object.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim Session1 As New BCSession


. . .
Set FavoritesBar1.Session = Session1
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FavoritesBar" on page 773

FavoritesBar::UIContext

The property returns or sets control's UI context.

Syntax

```
object.UIContext [ = string ]
```

String

The control's UI context.

Access

Read-Write

Remarks

“FavoritesBar” is used only.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

“FavoritesBar” on page 773

FavoritesBar::View

The property returns or sets the value of the control's current view.

Syntax

```
object.View [ = FStyles ]
```

FStyles

This property can be:

Constant	Value	Description
LargeIcons	0	Items is shown as the large icons (Default)
SmallIcons	1	Items is shown as the small icons

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
FavoritesBar1.View = LargeIcons
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FavoritesBar" on page 773

Folder

Folder is an object that contains an item collection.

Properties

Name	Description
Folder::Active	The property returns or sets a value that determines whether the folder is active.
Folder::CanRemove	The property returns or sets a value that determines whether the folder can be removed by a user.
Folder::CanRename	The property returns or sets a value that determines whether the folder can be renamed.
Folder::Caption	Returns or sets the description of the folder.
Folder::ChildControl	This property contains handle of the Favorites Bar child window, where folder items are displayed.
Folder::Enabled	The property returns or sets a value that determines whether the folder can respond to user-generated events.
Folder::FolderItems	The property returns or sets a value that contains the FolderItems collection object.
Folder::Position	Returns or sets the folder position in FavoritesBar.
Folder::Visible	The property returns or sets a value that determines whether the folder is visible or hidden.

Methods

Name	Description
Folder::Activate	The method activates the current folder.
Folder::Rename	The method renames the current folder.

Folder::Activate

The method activates the current folder.

Syntax

```
object.Activate
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"Folder" on page 797

Folder::Active

The property returns or sets a value that determines whether the folder is active.

Syntax

```
object.Active [= boolean]
```

Boolean

A Boolean value determined that the folder is active (True) or not (False).

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
If Folder1.Active then

'some operations with active folder
. . .
End If
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"Folder" on page 797

Folder::CanRemove

The property returns or sets a value that determines whether the folder can be removed by a user.

Syntax

```
object.CanRemove [= boolean]
```

Boolean

A Boolean value determines that a folder can be removed (Default).

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
If Folder1.CanRemove then

'Some operations before removal of a folder
. . .
End If
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"Folder" on page 797

Folder::CanRename

The property returns or sets a value that determines whether the folder can be renamed.

Syntax

```
object.CanRename [= boolean]
```

Boolean

A Boolean value determines that a folder can be renamed (Default).

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
If Folder1.CanRename then

'Some operations before renaming of a folder
. . .
End If
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"Folder" on page 797

Folder::Caption

Returns or sets the description of the folder.

Syntax

```
object.Caption [= string]
```

String

A line contains the caption.

Access

Read-Write

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"Folder" on page 797

Folder::ChildControl

This property contains a handle of the Favorites Bar child window, where folder items are displayed.

Syntax

```
object.ChildControl
```

A numeric (long type) value contents a handle.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
FavoritesBar1.ChildControl = Hierarchy1.hWnd
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"Folder" on page 797

Folder::Enabled

The property returns or sets a value that determines whether the folder can respond to user-generated events.

Syntax

```
object.Enabled [= boolean]
Boolean
```

A Boolean value determines that the control can respond to events (Default) else prevents control responding to events (False).

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
Folder1.Enabled = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"Folder" on page 797

Folder::FolderItems

The property returns or sets a value that contains the FolderItems collection object.

Syntax

```
Set object.FolderItems [= FolderItems]
```

FolderItems

The FolderItems object.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim FolderItems1 As New FolderItems

. . .
Set Folder1.FolderItems = FolderItems1
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"Folder" on page 797

Folder::Position

Returns or sets the folder position in FavoritesBar.

Syntax

```
Set object.Position [= long]
```

Long

The position of the folder in FavoritesBar.

Access

Read-Write

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"Folder" on page 797

Folder::Rename

The method activates the current folder.

Syntax

```
object.Rename
```

Return Values

If the return value (the Boolean type) is equal to True, the method succeeded. If the return value is False, the method failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"Folder" on page 797

Folder::Visible

The property returns or sets a value that determines whether the folder is visible or hidden.

Syntax

```
object.Visible [= boolean]
```

Boolean

A Boolean value determines that the folder is visible (Default) else an object is hidden (False).

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
Folder1.Visible = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"Folder" on page 797

FolderItem

This object contains the items - images, a caption and an actually object of DM, which appears when clicking the image.

Properties

Name	Description
FolderItem::BCObject	The property returns or sets a value that contains BCObject.
FolderItem::Caption	Returns or sets the description of the image.
FolderItem::Enabled	The property returns or sets a value that determines whether the item can respond to user-generated events.
FolderItem::Image	Returns or sets the item's image.
FolderItem::Position	Returns or sets the item position in its folder.
FolderItem::SmallImage	Returns or sets the item's small image.
FolderItem::ToolTipText	Returns or sets the text displayed when the mouse is paused over the item.
FolderItem::Visible	The property returns or sets a value that determines whether an object is visible or hidden.

Methods

Name	Description
FolderItem::Rename	The method renames Item.

FolderItem::BCObject

The property returns or sets a value that contains BCObject.

Syntax

```
object.BCObject [= BCObject]
```

BCObject

The property has the BCObject data type.

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim BCObject1 As New BCObject


. . .
Set FolderItem1.BCObject = BCObject1
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FolderItem" on page 805

"IBCObject" on page 80

FolderItem::Caption

Returns or sets the description of the image.

Syntax

`object.Caption [= string]`

String

A line contains the caption.

Access

Read-Write

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FolderItem" on page 805

FolderItem::Enabled

The property returns or sets a value that determines whether the item can respond to user-generated events.

Syntax

```
object.Enabled [= boolean]
```

Boolean

A Boolean value determines that an item can respond to events (Default) or prevents the control respond to events (False).

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
FolderItem1.Enabled = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FolderItem" on page 805

FolderItem::Image

Returns or sets the item's image.

Syntax

```
object.Image [= IPictureDisp]
```

IPictureDisp

An IPictureDisp value contains the picture object.

Access

Read-Write

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FolderItem" on page 805

FolderItem::Position

Returns or sets the item position in its folder.

Syntax

```
object.Position [= long]
```

Long

The position of the item in its folder.

Access

Read-Write

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FolderItem" on page 805

FolderItem::Rename

The method renames Item.

Syntax

```
object.Rename
```

Return Values

If return value (Boolean) is equal to True, the method succeeded. If return value is False, the method failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FolderItem" on page 805

FolderItem::SmallImage

Returns or sets the item's small image.

Syntax

```
object.SmallImage [= IPictureDisp]
```

IPictureDisp

An IPictureDisp value contains the picture object.

Access

Read-Write

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FolderItem" on page 805

FolderItem::ToolTipText

Returns or sets the text displayed when the mouse is paused over the item.

Syntax

```
object.ToolTipText [= string]
```

String

The any string information.

Access

Read-Write

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FolderItem" on page 805

FolderItem::Visible

The property returns or sets a value that determines whether an object is visible or hidden.

Syntax

```
object.Visible [= boolean]
```

Boolean

A Boolean value determines that the object is visible (Default) or hidden (False).

Access

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
FolderItem1.Visible = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FolderItem" on page 805

Folders

The Folders object is the collection the Folder objects.

Properties

Name	Description
Folders::Count	The property returns a number of the folder in the collection.
Folders::Item	The property returns one of the folders by the index.
Folders::_Item	The property returns one of the folders by the index (this is default property).

Methods

Name	Description
Folders::Add	Adds a folder into the end of the array or before specifies folder.
Folders::Remove	Removes specified folder from the collection.

Folders::Add

Adds a folder into the end of the array or before specifies folder.

Syntax

```
object.Add Caption As String, [Key], [Before]
```

Caption

Required. The property is a caption of the folder, the string data type.

Key

Optional. The property is a uniquestring value that identifies the folder.

Before

Optional. The property is uniquestring value that determines folder before which it is necessary to add a new folder.

Return Value

Long

Index in the folder collection.

Example

(Visual Basic 5.0)

```
Private Sub ...
```

```
. . .
Folders1.Add "My Favorites"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"Folders" on page 811

Folders::Count

The property returns a number of the folder in the collection.

Syntax

```
object.Count
```

This property (the long type) is count of the folders in the collection.

Access

Read-Only

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim CountFolders As Long
CountFolders = Folders1.CountEnd Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"Folders" on page 811

Folders::Item

The property returns one of the folders by the index.

Syntax

```
object.Item (index)
```

This property (the Folder type) is items in the folder collection. Index identifies the folder to be returned. A range is from 1 to value of the Count property.

index

Required. An expression that specifies the position of a member of the collection. If a numeric expression, an index should be a number from 1 to the value of the collection's Count property. If a string expression, an index should correspond to the key argument specified when the member was added to the collection.

Access

Read-Only

Example

(Visual Basic 5.0)

```
Private Sub ...
For i = 1 To Folders1.Count
v = Folders1.Item( i )
'Some operations with the v
. . .
Next i
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"Folders" on page 811

Folders::_Item

The property returns one of the folders by the index (this is default property).

Syntax

`object (`*index*`)`

This property (the Folder type) is items in the folder collection. Index identifies the folder to be returned. A range is from 1 to value of the Count property.

index

Required. An expression that specifies the position of a member of the collection. If a numeric expression, an index should be a number from 1 to the value of the collection's Count property. If a string expression, an index should correspond to the key argument specified when the member was added to the collection.

Access

Read-Only

Example

(Visual Basic 5.0)

```
Private Sub ...
For i = 1 To Folders1.Count
v = Folders1( i )
'Some operations with the v
. . .
Next i
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"Folders" on page 811

Folders::Remove

Removes specified folder from the collection.

Syntax

```
object.Remove index
```

index

The integer or other type identifying a folder in a collection.

Return Value

Boolean

A value is True if folder was removed successfully else a value is False.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"Folders" on page 811

FolderItems

The FolderItems object is the collection of the FolderItem objects.

Properties

Name	Description
FolderItems::Count	The property returns a number of the items in the collection.
FolderItems::Item	The property returns one of the Items by the index.
FolderItems::_Item	The property returns one of the Items by the index.(This property is default property)

Methods

Name	Description
FolderItems::Add	Adds an item into the end of the collection or before specified item.
FolderItems::AddObject	Adds the object into the end of the array.
FolderItems::Remove	Removes specified item from the collection.

FolderItems::Add

Adds an item into the end of the collection or before specified item.

Syntax

```
object.Add Caption, Image, [SmallImage], [Key], [Before]
```

Caption

Required. The property is a caption of the item, the string data type.

Image

Required. The property is an image for the item, the IPictureDisp data type.

SmallImage

Optional. The property is an image of the small size of the item,IPictureDisp data type.

Key

Optional. The property is a uniquestring value that identifies an item.

Before

Optional. The property is a uniquestring value that identifies an item before which it is necessary to add a new item.

Return Value

Long

Index of the item in the collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FolderItems" on page 816

FolderItems::AddObject

Adds the object into the end of the array.

Syntax

`object.AddObject` *`BCObject`*`, [`*`Key`*`], [`*`Before`*`]`

BCObject

Required. The property contains an object of the IBCObject type.

Key

Optional. The property is a unique string value that identifies an item.

Before

Optional. The property is a unique string value that identifies an item before which it is necessary to add a new item.

Return Value

Long

Index of the item in the collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FolderItems" on page 816

FolderItems::Count

The property returns a number of the items in the collection.

Syntax

```
object.Count
```

This property (the long type) is a number of items in the collection.

Access

Read-Only

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim CountItems As Long
CountItems = FolderItems1.CountEnd Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FolderItems" on page 816

FolderItems::Item

The property returns one of the Items by the index.

Syntax

```
object.Item (index)
```

This property (the FolderItem type) returns an item from the folder collection. Index identifies an item to be returned. The range is from 1 to the value of the Count property.

index

Required. An expression that specifies the position of FolderItem in the collection. If a numeric expression, an index should be a number from 1 to the value of the collection's Count property. If a string expression, an index should correspond to the key argument specified when the item was added to the collection.

Access

Read-Only

Example

(Visual Basic 5.0)

```
Private Sub ...
For i = 1 To FolderItems1.Count
v = FolderItems1.Item( i )
'Some operations with the v
. . .
Next i
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FolderItems" on page 816

FolderItems::_Item

The property returns one of the Items by the index.(This property is default property)

Syntax

`object (`*index*`)`

This property (the FolderItem type) returns an item from the folder collection. Index identifies an item to be returned. The range is from 1 to the value of the Count property.

index

Required. An expression that specifies the position of FolderItem in the collection. If a numeric expression, an index should be a number from 1 to the value of the collection's Count property. If a string expression, an index should correspond to the key argument specified when the item was added to the collection.

Access

Read-Only

Example

(Visual Basic 5.0)

```
Private Sub ...
For i = 1 To FolderItems1.Count
v = FolderItems1( i )
'Some operations with the v
. . .
Next i
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FolderItems" on page 816

FolderItems::Remove

Removes specified item from the collection.

Syntax

`object.Remove` *index*

index

The integer or other identifies the item in the collection.

Return Value

Boolean

A value equals True if an item was removed successfully else a value is False.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FavoritesBar.ocx"

See Also

"FolderItems" on page 816

Chapter

5

Event Handlers

In This Chapter

- "About Events API" on page 824
- "Using Events API" on page 824
- "Events API Interfaces" on page 829
- "Events API Enumerations" on page 862

About Events API

The Events API allows the developer to modify the internal functions of the DM Extensions at runtime. Functions can be added to the menus, other functions can be restricted by programmable criteria, and still other functions can be completely removed from the Desktop.

You can also register your event handler code, so that it will be executed every time when a specific Object Model method is called. Thus, you can extend or even completely replace standard method processing.

Event handlers can be chained, i.e. several event handlers can be installed at the same time. The handlers will be called in turn.

Notifications are supported via the standard mechanism of COM connection points. The developer can create event handlers using any language that supports COM, e.g., Visual Basic.

Implementing the Events API requires the developer to write an ActiveX DLL to handle or process an event. To install an event handlerIBCRegistrarExt::RegisterExtObject method should be used.

Using Events API

- "Notifications" on page 824
- "How to create a notifications handler" on page 825
- "How to install a notifications handler" on page 825
- "Using Events API Example #1" on page 825
- "Using Events API Example #2" on page 827

Notifications

The mechanism of notifications gives you access to several important features of the DM Extensions – menus and notifications.

Using notifications you can extend DM Extensions menus by adding your own items or customize standard menu items (or items, added by different extensions) on the fly. For example, you can disable menu items (like Delete) based on the user's primary group.

The notification's mechanism also allows you to create and install a notifications handler code, which will be called every time when a particular activity (such as method calls) takes place in DMO.

Notification handlers can also add custom processing before or/and after the DMO method will be executed. Notification handlers can be chained, meaning that you can have several notification handlers installed at the same time (they will be called in sequence).

Notifications are supported via standard COM Connection Points mechanism – that is why notification handlers should be implemented as COM modules. You can create notification handlers using any language that supports COM, including VB.

How to create a notifications handler

This section includes two notification handler examples – they show you how easily you can extend pull-down and context menu and how you can add your own custom processing to standard document events.

How to install a notifications handler

To install a notifications handler you should create a new key under HKEY_LOCAL_MACHINE\SOFTWARE\PC DOCS Inc.\PowerDOCS\OM\EventHandlers and specify class ID (or progID) of your notification's handler as a default value here.

Using Events API Example #1

Extending context menu

This example shows how you can add your own command to the context menu and disable one of the standard menu items (Delete).

```
' Class module, which implements notifications interfaces
Implements IOMEventHandler
Implements IOMDocumentMenuSink

Public Sub IOMEventHandler_Init(ByVal pApp As Application)

    ' VBConnector component is used to connect our event handler
code
    '  to the Object Model
    Dim connector As New VBConnector
```

```
    Dim interfacesConnected As Integer

    ' connect method returns a number of successful connections
    interfacesConnected = connector.Connect(pApp, Me)

End Sub

Public Sub IOMEventHandler_Terminate(ByVal pApp As IApplication)

    ' Nothing to do here

End Sub

' Document menu events handler
Public Sub IOMDocumentMenuSink_GetMenuItems(ByVal pDocProfs As
DocProfiles, _
                                            ByVal pCM As
IControlMenu)
    Dim lpVerbs As IBCEnumVerbs
    Dim cmd As IBCVerb

    Set lpVerbs = pCM.GetMenu()

    ' Disable Delete command
    For Each cmd In lpVerbs
        ' command ID
        Dim cmdID As String
        cmdID = cmd.ID

        If cmdID = "DOCSOPEN.Delete" Then
            pCM.ChangeState cmdID, VF_GRAYED
        End If
    Next

    ' Add our test menu item
    pCM.AddVerb "TEST.Test", "Test command", "DOCSOPEN.DOCUMENT",
_
        "DOCSOPEN.Delete", VF_ENABLED

End Sub
```

```
Public Sub IOMDocumentMenuSink_SelectionChanged(ByVal pDocProfs As
DocProfiles, _
                                                ByVal pCM As
IControlMenu)

    ' if more than one document is selected then ...
    If pDocProfs.Count > 1 Then
        ' gray out our menu item
        pCM.ChangeState "TEST.Test", vbGrayed
    End If

    ' you can also modify standard menu items here (and other
extensions' items)
End Sub

Public Function IOMDocumentMenuSink_ExecuteMenuItem(ByVal
pDocProfs As DocProfiles, _
                ByVal cmdID As String) As Integer

    ' if this is our menu item...
    If cmdID = "TEST.Test" Then
        ' ... do something useful
        MsgBox "Test !!"
    End If

    ' you can also process standard menu items here

End Function
```

Using Events API Example #2

Receiving notifications about different DMO events

This example shows how you can receive notifications when OM methods are invoked.

```
' Class module, which implements notifications interfaces
Implements IOMDocumentEventSink
Implements IOMEventHandler

' IOMEventHandler
```

```
Public Sub IOMEventHandler_Init(ByVal pApp As Application)

    ' VBConnector component is used to connect our event handler
code
    '  to the Object Model
    Dim connector As New VBConnector

    Dim interfacesConnected As Integer

    ' connect method returns a number of successful connections
    interfacesConnected = connector.Connect(pApp, Me)

End Sub

Public Sub IOMEventHandler_Terminate(ByVal pApp As IApplication)

    ' Nothing to do here

End Sub

' IOMDocumentEventSink interface – used to receive notifications
about profile and version events
Public Function IOMDocumentEventSink_ProfileEvent(ByVal EventID As
Integer, _
                ByVal EventType As Integer, ByVal p As Profile, _
                ByVal pParams As IBCCoreCtx) As Integer

    ' if user wants to delete a document
    If EventID = DE_DELETE And EventType = ET_PRE Then
        ' warn him
        MsgBox "you are about to delete " & p.Name
    End If

End Function

Public Function IOMDocumentEventSink_VersionEvent(ByVal EventID As
Integer, _
             ByVal EventType As Integer, ByVal p As Version, _
             ByVal pParams As IBCCoreCtx) As Integer
```

```
    ' nothing to do here

End Function_
```

Events API Interfaces

- "IControlMenu" on page 829
- "IOMDocumentEventSink" on page 838
- "IOMDocumentMenuSink" on page 848
- "IOMEventHandler" on page 852
- "IOMEnumSink" on page 854
- "IOMMenuSink" on page 856

IControlMenu

The ControlMenu object represents a menu. The object has methods that help to manipulate menu commands (verbs).

Methods

IControlMenu Methods	Description
IControlMenu::AddVerb	The method adds a command to a menu.
IControlMenu::ChangeState	The method changes the state of a command.
IControlMenu::DeleteVerb	The method removes a command from a menu.
IControlMenu::GetMenu	The method returns an enumeration of all available commands for a menu.
IControlMenu::SetVerbIcons	The method assigns icons to a menu command.
IControlMenu::SetVerbProperties	The method sets attributes of a menu command.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "FsPlugin.dll"

IControlMenu::AddVerb

The method adds a command to a menu.

C++ Syntax

```
HRESULT Add(
BSTR bstrCmdID,
BSTR bstrName,
BSTR  bstrParent,
BSTR  bstrInsBefore,
long  lFlags);
```

Visual Basic Syntax

```
ControlMenu.Add (bstrCmdID, bstrName,  bstrParent,  bstrInsBefore,
lFlags)_
```

Parameters

bstrCmdID	[in] Required. A string that uniquely identifies the command.
bstrName	[in] Required. A string that specifies the command name.
bstrParent	[in] Required. A string that specifies the identifier of the parent menu item (to create a pop-up menu).
bstrInsBefore	[in] Required. A string that specifies the identifier of the command before which this verb will be placed. If it is an empty string, the command will be added at the end of the menu.
lFlags	[in] Required. A long that specifies the command state. The parameter can be one or more of the VerbFlags.

Return values

S_OK	Successful operation description.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FsPlugin.dll"

See Also

"IControlMenu" on page 829

"VerbFlags Enumeration" on page 182

IControlMenu::ChangeState

The method changes the state of a command.

C++ Syntax

```
HRESULT ChangeState(
BSTR bstrVerbID,
long  lState);
```

Visual Basic Syntax

```
ControlMenu.ChangeState (bstrVerbID,  lState)_
```

Parameters

bstrVerbID	[in] Required. A string that uniquely identifies the command.
long	[in] Required. A long that specifies the command state. The parameter can be one or more of the VerbFlags.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FsPlugin.dll"

See Also

"IControlMenu" on page 829

"VerbFlags Enumeration" on page 182

IControlMenu::DeleteVerb

The method removes a command from a menu.

C++ Syntax

```
HRESULT DeleteVerb(

BSTR bstrVerbID);
```

Visual Basic Syntax

```
ControlMenu.DeleteVerb (bstrVerbID)_
```

Parameters

bstrVerbID	[in] Required. A string that uniquely identifies the command.

Return values

S_OK	Successful operation description.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FsPlugin.dll"

See Also

"IControlMenu" on page 829

IControlMenu::GetMenu

The method returns an enumeration of all available commands for a menu.

C++ Syntax

```
HRESULT GetMenu(IBCEnumVerbs** ppEnumVerbs);
```

Visual Basic Syntax

```
[ Set ppEnumVerbs  = ] ControlMenu.GetMenu_
```

Parameters

ppEnumVerbs	[out, retval] Returns a BCEnumVerbs object that specifies the requested commands.

Return values

S_OK	Successful operation description.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FsPlugin.dll"

See Also

"IControlMenu" on page 829

"IBCEnumVerbs" on page 74

IControlMenu::SetVerbIcons

The method assigns icons to a menu command.

C++ Syntax

```
HRESULT SetVerbIcons(
BSTR bstrVerbID,
VARIANT_BOOL bSmall,
long hIcon,
long hActiveIcon,
long hIconDown);
```

Visual Basic Syntax

```
ControlMenu.SetVerbIcons (bstrVerbID, bSmall, hIcon, hActiveIcon,
hIconDown)_
```

Parameters

bstrVerbID	[in] Required. A string that uniquely identifies the command.
bSmall	[in] Required. A Boolean value that indicates whether the icons being assigned to the command are small or large as follows: True: Icons are small. False: Icons are large.
hIcon	[in] Required. A long that specifies the HICON value of the default icon for command's normal or "up" state.
hActiveIcon	[in] Required. A long that specifies the HICON value of the icon for command's focused state (when the mouse is over the icon on the toolbar).
hIconDown	[in] Required. A long that specifies the HICON value of the icon for command's selected or "down" state.

Return values

S_OK	Successful operation description.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FsPlugin.dll"

See Also

"IControlMenu" on page 829

IControlMenu::SetVerbProperties

The method sets attributes of a menu command.

C++ Syntax

```
HRESULT SetVerbProperties(
BSTR bstrVerbID,
```

```
BSTR bstrShortName,
BSTR  bstrDescription,
BSTR  bstrToolTip,
long  lAccelKey);
```

Visual Basic Syntax

```
ControlMenu.SetVerbProperties (bstrVerbID, bstrShortName,
bstrDescription,  bstrToolTip,  lAccelKey)_
```

Parameters

bstrVerbID	[in] Required. A string that uniquely identifies the command.
bstrShortName	[in] Required. A string that specifies the command short name, which will appear on the toolbar.
bstrDescription	[in] Required. A string that specifies the full descriptive name of the command to appear in the status bar.
bstrToolTip	[in] Required. A string that specifies the command tooltip.
lAccelKey	[in] Required. A long that specifies the command shortcut. Its low-order word should be either a virtual-key code or an ASCII character code. The high-order word should specify the accelerator flags. This word can be a combination of the following values: FSHIFT (&H4): The Shift key should be held down when the accelerator key is pressed. FCONTROL (&H8): The Ctrl key should be held down when the accelerator key is pressed. FALT (&H10): The Alt key should be held down when the accelerator key is pressed.

Return values

S_OK	Successful operation description.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "DECore.idl"

See Also

"IControlMenu" on page 829

IOMDocumentEventSink

The IOMDocumentEventSink interface receives notifications about various profile and version events.

When to Implement

The IOMDocumentEventSink interface is implemented by an event handler.

Methods

IOMDocumentEventSink Methods	Description
IOMDocumentEventSink::ProfileEvent	The method receives notifications about profile events.
IOMDocumentEventSink::VersionEvent	The method receives notifications about version events.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "FsPlugin.dll"

IOMDocumentEventSink::ProfileEvent

The method receives notifications about profile events.

C++ Syntax

```
HRESULT ProfileEvent(
short sEventID,
short sEventType,IProfile*  pProfile,IBCCoreCtx*  pParams,
short*  psRetCode);
```

Visual Basic Syntax

```
[  psRetCode  = ] OMDocumentEventSink.ProfileEvent (sEventID,
sEventType,  pProfile,  pParams)_
```

Parameters

sEventID	[in] An integer that uniquely identifies the event. The parameter can be one of the DEvents values.
sEventType	[in] An integer that specifies the event type. The parameter can be one of the DEventTypes values.
pProfile	[in] A Profile object that specifies the document's profile the event is related to.
psRetCode	[out,retval] Returns an integer that specifies whether or not the event has been processed. For possible values see HResultCodes Enumeration and HResultCodesEx Enumeration.
pParams	[in] A BCCoreCtx object that specifies event parameters as follows:

Event ID	Context Type	Key	Value
All	All	ParentWindow	A window handle that should be used to display any messages and dialogs during event processing.
All	ET_POST	Success	A Boolean value that indicates whether the operation succeeded.
DE_OPEN	ET_PRE	Mode	A long that corresponds to the *mode* parameter of the Open method.
DE_OPEN	ET_POST	-	-

Event ID	Context Type	Key	Value
DE_OPEN_COMPATIBLE	ET_PRE	Flags	An integer that corresponds to the *flags* parameter of the OpenInCompatibleApp method.
Mode	A long that corresponds to the *mode* parameter of the OpenInCompatibleApp method.	Application	A variant that corresponds to the *application* parameter of the OpenInCompatibleApp method.
DE_OPEN_COMPATIBLE	ET_POST	-	-
DE_LOCK	ET_PRE	Flags	An integer that corresponds to the *flags* parameter of the LockProfile method.
ReturnDate	A date that corresponds to the *date* parameter of the LockProfile method.	Comments	A string that corresponds to the *comments* parameter of the LockProfile method.
DE_LOCK	ET_POST	-	-
DE_UNLOCK	ET_PRE	Flags	An integer that corresponds to the *flags* parameter of the UnlockProfile method.
EditTime	A long that corresponds to the *time* parameter of the UnlockProfile method.	CopyFrom	A string that corresponds to the *path* parameter of the UnlockProfile method.

Event ID	Context Type	Key	Value
Action	An integer that corresponds to the *action* parameter of the UnlockProfile method.	DE_UNLOCK	ET_POST
-	-	DE_DELETE	ET_PRE
Flags	An integer that corresponds to the *flags* parameter of the Delete method.	DE_DELETE	ET_POST
-	-	DE_PRINT	ET_PRE
-	-	DE_PRINT	ET_POST
-	-	DE_PREVIEW	ET_PRE
-	-	DE_PREVIEW	ET_POST
-	-	DE_SELECT_FOR_EDIT	ET_PRE
Flags	An integer that corresponds to the *flags* parameter of the SelectForEdit method.	DE_SELECT_FOR_EDIT	ET_POST
-	-	DE_RELEASE_DOC	ET_PRE
-	-	DE_RELEASE_DOC	ET_POST
-	-	DE_SAVE	ET_PRE
Flags	An integer that corresponds to the *flags* parameter of the Save method.	DE_SAVE	ET_POST
-	-	DE_NEW_VERSION	ET_PRE
-	-	DE_NEW_VERSION	ET_POST
-	-	DE_NEW_DOC	ET_PRE

Event ID	Context Type	Key	Value
-	-	DE_NEW_DOC	ET_POST
-	-	DE_ADD_LINK	ET_PRE
Link	An object (can be obtained via GetObjectParam) that indicate document/folder is being added to folder.	LinkVersionType	A string that represent version type of component.
LinkDisplayName	A string that represent display name of component.	DE_ADD_LINK	ET_POST
-	-	DE_REMOVE_LINK	ET_PRE
Link	An object (can be obtained via GetObjectParam) that indicate document/folder is being removed from folder.	DE_REMOVE_LINK	ET_POST
-	-	DE_UPDATE_LINK	ET_PRE
Link	An object (can be obtained via GetObjectParam) that indicate document/folder is being updated.	LinkVersionType	A string that represent new version type of component.
LinkDisplayName	A string that represent new display name of component.	DE_UPDATE_LINK	ET_POST
-	-		

On output an event handler can specify the following values in this parameter:

Event ID	Context Key	Value
All	ErrCode	An integer that will be used to set an error code in the IOMErrInfo object.
ErrText	A string that will be used to set an error text in the IOMErrInfo object.	

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

The method is called before and after the corresponding method on a Profile object is executed. Your are able to add your custom code here.

Example

The following example displays a message box each time a document is about to be deleted.

```
Public Function IOMDocumentEventSink_ProfileEvent( _
ByVal EventID As Integer, ByVal EventType As Integer, _
ByVal Prof As Profile, ByVal Params As BCCoreCtx) As Integer
If EventID = DE_DELETE And EventType = DE_PRE Then
MsgBox "You are about to delete document #" + _
Str(Prof.DocNumber) + " – " + Prof.Name + "."
End if
IOMDocumentEventSink_ProfileEvent = HS_NEXT_HANDLER
End Function_
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FsPlugin.dll"

See Also

"IProfile" on page 419

"IBCCoreCtx" on page 46

"DEvents Enumeration" on page 862

"DEventTypes Enumeration" on page 865

"IOMErrInfo" on page 412

"HResultCodes Enumeration" on page 865

"HResultCodesEx Enumeration" on page 866

IOMDocumentEventSink::VersionEvent

The method receives notifications about version events.

C++ Syntax

```
HRESULT VersionEvent(
short sEventID,
short sEventType,IVersion*  pVersion,IBCCoreCtx*  pParams,
short*  psRetCode);
```

Visual Basic Syntax

```
[  psRetCode  = ] OMDocumentEventSink.VersionEvent (sEventID,
sEventType,  pVersion,  pParams)_
```

Parameters

sEventID	[in] An integer that uniquely identifies the event. The parameter can be one of the DEvents values.
sEventType	[in] An integer that specifies the event type. The parameter can be one of the DEventTypes values.
pVersion	[in] A Version object that specifies the document version the event is related to.
psRetCode	[out,retval] Returns an integer that specifies whether or not the event has been processed. For possible values see HResultCodes Enumeration and HResultCodesEx Enumeration.
pParams	[in] A BCCoreCtx object that specifies event parameters as follows:

Event ID	Context Type	Key	Value
All	All	ParentWindow	A window handle that should be used to display any messages and dialogs during event processing.
All	ET_POST	Success	A **Boolean** value that indicates whether the operation succeeded.
DE_OPEN	ET_PRE	Mode	A long that corresponds to the *mode* parameter of the Open method.
DE_OPEN	ET_POST	-	-

Event ID	Context Type	Key	Value
DE_OPEN_COMPATIBLE	ET_PRE	Flags	An integer that corresponds to the *flags* parameter of the OpenInCompatibleApp method.
Mode	A long that corresponds to the *mode* parameter of the OpenInCompatibleApp method.	Application	A variant that corresponds to the *application* parameter of the OpenInCompatibleApp method.
DE_OPEN_COMPATIBLE	ET_POST	-	-
DE_DELETE	ET_PRE	-	-
DE_DELETE	ET_POST	-	-
DE_PRINT	ET_PRE	-	-
DE_PRINT	ET_POST	-	-
DE_PREVIEW	ET_PRE	-	-
DE_PREVIEW	ET_POST	-	-
DE_CHECK_OUT	ET_PRE	ReturnDate	A date that corresponds to the *date* parameter of the CheckOut method.
Comments	A string that corresponds to the *comments* parameter of the CheckOut method.	Path	A string that corresponds to the *path* parameter of the CheckOut method.
DE_CHECK_OUT	ET_POST	-	-
DE_UPLOAD	ET_PRE	-	-
DE_UPLOAD	ET_POST	-	-
DE_DOWNLOAD	ET_PRE	-	-
DE_DOWNLOAD	ET_POST	-	-

Event ID	Context Type	Key	Value
DE_LOCK_FOR_EDIT	ET_PRE	Flags	An integer that corresponds to the *flags* parameter of the LockVersionForEdit method.
DE_LOCK_FOR_EDIT	ET_POST	-	-
DE_UNLOCK_VERSION	ET_PRE	Flags	An integer that corresponds to the *flags* parameter of the UnlockVersion method.
DE_UNLOCK_VERSION	ET_POST	-	-

On output an event handler can specify the following values in this parameter:

Event ID	Context Key	Value
All	ErrCode	An integer that will be used to set an error code in the IOMErrInfo object.
ErrText	A string that will be used to set an error text in the IOMErrInfo object.	

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

The method is called before and after the corresponding method on a Version object is executed. You are able to add your custom code here.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FsPlugin.dll"

See Also

"IVersion" on page 625

"IBCCoreCtx" on page 46

"DEvents Enumeration" on page 862

"DEventTypes Enumeration" on page 865

"HResultCodes Enumeration" on page 865

"HResultCodesEx Enumeration" on page 866

IOMDocumentMenuSink

The IOMDocumentMenuSink interface makes it possible to modify the DM Extensions menu at runtime.

When to Implement

The developer should implement this interface if he/she wants to receive notifications about events related to the DM Extensions menu.

Methods

IOMDocumentMenuSink Methods	Description
IOMDocumentMenuSink::ExecuteMenuItem	The method executes a menu command.
IOMDocumentMenuSink::GetMenuItems	The method returns all menu items for a collection of documents available to the user.
IOMDocumentMenuSink::SelectionChanged	The method notifies a sink that the user has selected other documents and the menu items should be modified in accordance with the user's selection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "FsPlugin.dll"

IOMDocumentMenuSink::ExecuteMenuItem

The method executes a menu command.

C++ Syntax

```
HRESULT ExecuteMenuItem(IDocProfiles*  pProfiles,
BSTR  bstrCmdID,
```

```
short*  psRetCode);
```

Visual Basic Syntax

```
[  psRetCode  = ] OMDocumentMenuSink.ExecuteMenuItem ( pProfiles,
bstrCmdID)_
```

Parameters

pProfiles	[in] A DocProfiles object that specifies the collection of document's profiles to execute the command.
bstrCmdID	[in] A string that uniquely identifies the command to execute.
psRetCode	[out,retval] Returns an integer that specifies whether or not the command has been processed. For possible values see HResultCodes Enumeration.

Return values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The method is called when necessary to execute a menu command. You should handle all your custom menu commands here.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FsPlugin.dll"

See Also

"IDocProfiles" on page 292

"IControlMenu" on page 829

"HResultCodes Enumeration" on page 865

IOMDocumentMenuSink::GetMenuItems

The method returns all menu items for a collection of documents available to the user.

C++ Syntax

```
HRESULT GetMenuItems(IDocProfiles*  pProfiles,IControlMenu*
pMenu,
);
```

Visual Basic Syntax

```
OMDocumentMenuSink.GetMenuItems ( pProfiles,  pMenu)_
```

Parameters

pProfiles	[in] A DocProfiles object that specifies the collection of document's profiles to determine their available menu items. If pProfiles is NULL, then the requested menu items will be embedded in the main DM Extensions Document menu.
pMenu	[in] A ControlMenu object that specifies all menu items available to the collection of documents.

Return values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The method is called when necessary to construct the complete context or pull-down menu for the selected documents.

The developer can affect the available menu by modifying the ControlMenu object the sink receives as an initial menu.

Example

The following example disables the Delete command and adds a new command to the menu.

```
Public Sub IOMDocumentMenuSink_GetMenuItems( _
ByVal Profs As DocProfiles, ByVal CM As ControlMenu)
Dim cmd As BCVerb


' Disable the Delete command
For Each cmd In CM.GetMenu
```

```
If cmd.ID = "DOCSOPEN.Delete" Then
CM.ChangeState cmd.ID, VF_DISABLED
End if
Next

' Add the test menu item
CM.AddVerb "TEST.Test", "Test", "DOCSOPEN.DOCUMENT", _
"DOCSOPEN.Delete", VF_ENABLED
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FsPlugin.dll"

See Also

"IDocProfiles" on page 292

"IControlMenu" on page 829

IOMDocumentMenuSink::SelectionChanged

The method notifies a sink that the user has selected other documents and the menu items should be modified in accordance with the user's selection.

C++ Syntax

```
HRESULT SelectionChanged(IDocProfiles*  pProfiles,IControlMenu*
pMenu);
```

Visual Basic Syntax

```
OMDocumentMenuSink.SelectionChanged ( pProfiles,  pMenu)_
```

Parameters

pProfiles	[in] A DocProfiles object that specifies the collection of document's profiles the user has selected.
pMenu	[in] A ControlMenu object that specifies all menu items available to the collection of documents.

Return values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

The method is called when necessary to change the state of menu items in accordance to the documents the user selected. The developer can affect the available menu by modifying the ControlMenu object the sink receives as an initial menu.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FsPlugin.dll"

See Also

"IDocProfiles" on page 292

"IControlMenu" on page 829

IOMEventHandler

The IOMEventHandler interface is used to initialize and terminate an event handler.

When to Implement

IOMEventHandler is the main interface, which should implement each event handler.

Methods

IOMEventHandler Methods	Description
IOMEventHandler::Init	The method initializes an event handler.
IOMEventHandler::Terminate	The method terminates an event handler.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "FsPlugin.dll"

IOMEventHandler::Init

The method initializes an event handler.

C++ Syntax

```
HRESULT Init(IApplication*   pApp);
```

Visual Basic Syntax

```
OMEventHandler.Init ( pApp)_
```

Parameters

pApp	[in] An Application object that identifies the Document Management plug-in.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

The method is called up at once when the Document Management plug-in starts. The developer should provide all necessary initializations here.

The Application object supports the standard COM connection points mechanism. You should establish a connection between the Application connection point object and your notification sink.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FsPlugin.dll"

See Also

"IApplication" on page 241

IOMEventHandler::Terminate

The method terminates an event handler.

C++ Syntax

```
HRESULT Terminate(IApplication*   pApp);
```

Visual Basic Syntax

```
OMEventHandler.Terminate ( pApp)_
```

Parameters

pApp	[in] An Application object that identifies the Document Management plug-in.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Remarks

The method is called up at once when the Document Management plug-in unloads. The developer should provide all necessary termination codes here.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FsPlugin.dll"

See Also

"IApplication" on page 241

IOMEnumSink

This object allows to replace standard enumeration.

Methods

IOMMenuSink Methods	Description
IOMEnumSink::OnEnumChildObjects	Replace enumerator for an object.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "FsPlugin.dll"

IOMEnumSink::OnEnumChildObjects

Replace enumerator for an object.

C++ Syntax

```
HRESULT OnEnumChildObjects(IBCObject*  pParentObject,IBCCoreCtx*
pCtx,IBCEnumObjects**  ppEnum,short*  psResult);
```

Visual Basic Syntax

```
[ psResult = ] OMEnumSink.OnEnumChildObjects (pParentObject, pCtx,
ppEnum) _
```

Parameters

pParentObject	[in] An object contents of that is being enumerated.
pCtx	[in] Context that are used for enumeration. For more information see IBCObject::EnumChildObjects.
ppEnum	[in, out] On input – standard enumerator object. On output – replaced enumerator object.
psResult	[out, retval] Returns an integer that specifies whether or not the event has been processed. For possible values see HResultCodes Enumeration.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "FsPlugin.dll"

See Also

"IOMEnumSink" on page 854

IOMMenuSink

This interface makes it possible to modify the DM Extensions menu at runtime.

When to Implement

The developer should implement the IOMMenuSink interface if he/she wants to receive notifications about events related to the DM Extensions menu. This interface superceded IOMDocumentMenuSink.

This is a sink interface, which should be passed to the Document Management plug-in connectable object (implemented by the main

IApplication object). Then the sink receives notifications through connection point mechanism.

Methods

IOMMenuSink Methods	Description
IOMMenuSink::GetMenuItems	Modifies the menu the Document Management plug-in provides for its objects.
IOMMenuSink::SelectionChanged	Notifies the sink that the user has selected other objects and the menu items should be modified in accordance with the user's selection.
IOMMenuSink::ExecuteMenuItem	Executes a specified menu command.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "FsPlugin.dll"

See Also

"IOMDocumentMenuSink" on page 848

IOMMenuSink::ExecuteMenuItem

The method executes a specified menu command.

C++ Syntax

```
HRESULT ExecuteMenuItem(IOMObjects*  pObjects,
BSTR   bstrCmdID,
short*   psRetCode,
);
```

Visual Basic Syntax

```
[ psRetCode = ] OMMenuSink.ExecuteMenuItem (pObjects, bstrCmdID) _
```

Parameters

pObjects	[in] A pointer to the IOMObjects interface that specifies the currently selected objects.
bstrCmdID	[in] An address of a Basic wide character string (two bytes per character) to contain the unique identifier of a command.
psRetCode	[out, retval] A pointer to a short variable to receive one of the HResultCodes result codes upon successful return.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.
E_OUTOFMEMORY	There is not enough memory to complete the operation.
E_UNEXPECTED	An unexpected error occurred.

Remarks

The method is called every time a menu or toolbar command is about to be executed. You should handle all your custom menu commands here.

It is the connectable object's (source's) responsibility to track the result code in such a way to guarantee the concurrent work of different event handlers.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FsPlugin.dll"

See Also

"IOMObjects" on page 414

"HResultCodes Enumeration" on page 865

IOMMenuSink::GetMenuItems

The method modifies the menu the Document Management plug-in provides for its objects.

C++ Syntax

```
HRESULT GetMenuItems(IOMObjects*  pObjects,IControlMenu*  pMenu,
);
```

Visual Basic Syntax

```
OMMenuSink.GetMenuItems ( pObjects,  pMenu )_
```

Parameters

pObjects	[in] A pointer to the IOMObjects interface that specifies the currently selected objects. The pointer is NULL if the sink should provide main menu commands available for all objects.
pMenu	[in] A pointer to the IControlMenu interface that specifies all menu items available for objects from the pObjects collection. The sink is allowed to modify the IControlMenu's commands to customize the menu.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.
E_OUTOFMEMORY	There is not enough memory to complete the operation.
E_UNEXPECTED	An unexpected error occurred.

Remarks

The method is called when necessary to construct the main DM Extensions menu or the context menu for the selected objects.

The developer can affect the available menu by modifying the ControlMenu object the sink receives as an initial menu. The connectable object (source) is responsible for changing the actual DM Extensions menu in accordance with IControlMenu the sink returns

but it is the sink's responsibility to provide the unique names and identifiers for sink's custom commands.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FsPlugin.dll"

See Also

"IOMObjects" on page 414

"IControlMenu" on page 829

IOMMenuSink::SelectionChanged

The method notifies the sink that the user has selected other objects and the menu items should be modified in accordance with the user's selection.

C++ Syntax

```
HRESULT SelectionChanged(IOMObjects*  pObjects,IControlMenu*
pMenu,
);
```

Visual Basic Syntax

```
OMMenuSink.SelectionChanged (pObjects, pMenu) _
```

Parameters

pObjects	[in] A pointer to the IOMObjects interface that specifies the currently selected objects.
pMenu	[in] A pointer to the IControlMenu interface that specifies all menu items available for objects from the pObjects collection. The sink is allowed to modify the IControlMenu's commands to customize the menu.

Return values

S_OK	Successful operation description.
S_FALSE	The operation failed.
E_OUTOFMEMORY	There is not enough memory to complete the operation.
E_UNEXPECTED	An unexpected error occurred.

Remarks

The method is called every time the user changes the currently selected objects. The developer can affect the available menu by modifying the IControlMenu object the sink receives as an initial menu. For instance, the developer can change the menu item state. The connectable object (source) is responsible for changing the actual DM Extensions menu in accordance with the IControlMenu the sink returns.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "FsPlugin.dll"

See Also

"IOMObjects" on page 414

"IControlMenu" on page 829

Events API Enumerations

- "DEvents Enumeration" on page 862
- "DEventTypes Enumeration" on page 865
- "HResultCodes Enumeration" on page 865
- "HResultCodesEx Enumeration" on page 866

DEvents Enumeration

The enumeration contains available document events that are passed to the IOMDocumentEventSink::ProfileEvent and IOMDocumentEventSink::VersionEvent methods in the sEventID parameter.

Elements

DEvents Elements	Description
DE_OPEN	Indicates that a document is being opened in its native application with the Open method or version is being opened in its native application with the Open method.
DE_OPEN_COMPATIBLE	Indicates that a document is being opened in an application compatible with its native application with the OpenInCompatibleApp method or version is being opened in an application compatible with its native application with the OpenInCompatibleApp method.
DE_LOCK	Indicates that a document profile is being locked with the LockProfile method.
DE_UNLOCK	Indicates that a document profile is being unlocked with the UnlockProfile method.
DE_DELETE	Indicates that a document is being deleted with the Delete method or version is being deleted with the Delete method.
DE_PRINT	Indicates that a document is being printed with the PrintDoc method or a document version is being printed with the PrintVersion method.
DE_PREVIEW	Indicates that the documents are being viewed with the Preview method or version's contents are being viewed with the Preview method.
DE_SELECT_FOR_EDIT	Indicates that a document is being selected for editing with the SelectForEdit method.

DEvents Elements	Description
DE_RELEASE_DOC	Indicates that a document is being unlocked with the ReleaseDoc method.
DE_SAVE	Indicates that a document profile is being saved with the Save method.
DE_NEW_VERSION	Indicates that a new version of a document is being created with the CreateVersionFromFile method.
DE_NEW_DOC	Indicates that a new profile is being created with the CreateProfile method on a library.
DE_CHECK_OUT	Indicates that a document version is being checked out with the CheckOut method.
DE_UPLOAD	Indicates that a document version is being downloaded from the server with the DownloadFile or Download method.
DE_DOWNLOAD	Indicates that version is being downloaded.
DE_LOCK_FOR_EDIT	Indicates that a document version is being locked for editing with the LockVersionForEdit method.
DE_UNLOCK_VERSION	Indicates that a document version is being unlocked with the UnlockVersion method.
DE_UNLOCK_VERSION	Indicates that a document version is being unlocked with the UnlockVersion method.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "FsPlugin.dll"

See Also

"IProfile" on page 419

"IVersion" on page 625

"IOMDocumentEventSink" on page 838

DEventTypes Enumeration

This enumeration contains event types that are passed to IOMDocumentEventSink::ProfileEvent and IOMDocumentEventSink::VersionEvent in the sEventType parameter.

Elements

DEventTypes Elements	Description
ET_PRE	Indicates that the method is called before the event occurs and the corresponding Profile or Version method is processed.
ET_POST	Indicates that the method is called when the event has occurred and the corresponding Profile or Version method has been processed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "FsPlugin.dll"

See Also

"IProfile" on page 419

"IVersion" on page 625

"IOMDocumentEventSink" on page 838

HResultCodes Enumeration

This enumeration specifies available result codes that an event handler may return as the notification result.

Elements

HResultCodes Elements	Description
HS_EVENT_PROCESSED	Indicates that an event has been processed and other event handlers should not be notified. An event should not be processed. E.g. the Document Management Objects module should not download version. If this event has the ET_POST event type, other event handlers will not be notified.
HS_NEXT_HANDLER	Indicates that an event should be passed to next event handler and processed in the Document Management Objects module.
HS_CANCEL_NOTIFY	Indicates that an event should not be passed to next event handlers and processed in the Document Management Objects module.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "FsPlugin.dll"

HResultCodesEx Enumeration

This enumeration specifies available extended result codes that an event handler may return as the notification result. These codes can be used in combination with values from HResultCodes. If event handlers support these extended result codes it may significantly increase DM Extensions performance. A result code from this enumeration relates to combination of Event and EventType. E.g. if an event handler returns HS_SKIP_NOTIFY for (DE_DOWNLOAD, ET_PRE) an event it will be still notified about the (DE_DOWNLOAD, ET_POST) event.

Elements

HResultCodesEx Elements	Description
HSEX_SKIP_NOTIFY = 0x10	Indicates that notification for this event and event type may not be passed for an event handler. An event handler may still receive this kind of notification. Notification will not be sent only if all event handlers will skip this notification.
HSEX_ASYNC_NOTIFY= 0x20	Indicates that notification for this event can be processed by notification handlers asynchronously. This code means that an event handler want to receive this notification for information purpose only and the HS_EVENT_PROCESSED result code will not be used. If an event handler return the HS_EVENT_PROCESSED result code in asynchronous notification this code will be ignored.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "FsPlugin.dll"

See Also

"HResultCodes Enumeration" on page 865

Chapter

6

Custom Forms

In This Chapter

- "About Custom Forms" on page 870
- "Custom Forms API Reference" on page 873
- "Custom Forms Enumerations" on page 954

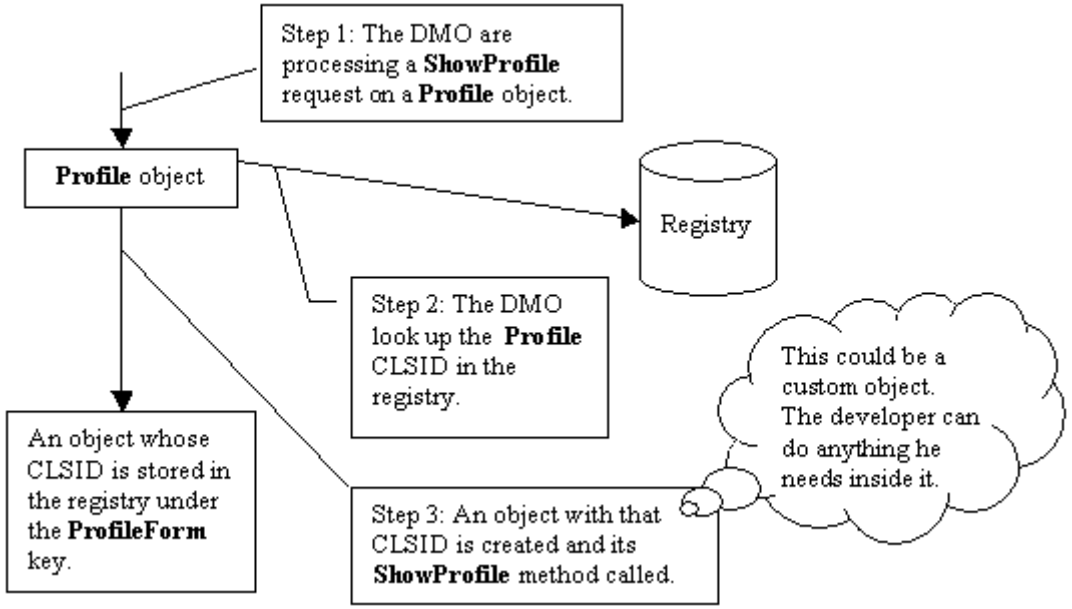
About Custom Forms

All forms in the product are treated as external COM modules. This "form" is a COM module, which implements an interface associated with the form. For example, an IProfileForm interface (with the only method called ShowProfileForm) may be implemented by a COM object, which is intended to replace the standard profile form.

This has several important benefits:

- Forms can be written independently of the DMO core.
- Every form is treated as a black box, so there are no restrictions on what it can and cannot do.
- Forms can use any APIs and interfaces, for example nothing prevents a form to be implemented as a DHTML dialog.
- Forms can be replaced without making any changes to the DMO. The developer just should change a single line in the system registry.

The DMO use forms as shown below:



These forms can have different look (for example, developers can use a background bitmap with their company logos in all product forms, even in forms than cannot be customized with the DM Designer), provide additional functionality or both.

Forms are generally used to get additional information from the user. When a DMO method is called, an appropriate form may be displayed, for instance, to allow the user to select a launch method to open a document.

Displaying a form is not necessary, but it is usually done. After getting all the necessary data from the user (and probably validating them) an appropriate DMO method is called up. This time, a flag should be specified to disable forms (otherwise, the form code will be called again, causing a loop). This procedure applies to the so-called "method-related" forms (check-out, etc.).

In other words, DMO rely on method-related forms to get missing method parameters from the user or to confirm those parameters' values.

There are also forms (such as profile forms) that are not related to any operations over a DMO object. They change internal state of the object instead (e.g., profile column values).

The DMO make almost no assumptions on the internal structure or behavior of custom forms. Developers are not required to show any forms at all or can display different forms, depending on the method parameters and so on. For example, you can override the lookup form and display different lookup forms for different columns (one for applications, another for peoples, etc.). You can also use whatever controls you like on your profile form (any ActiveX control), for example, you can create multi-page forms using a tab control and so on.

Implementing Custom Forms

Every form in the DM Extensions is treated as an external COM object. That is a form is a COM object, which exposes an interface associated with the form.

Consider, for example, a ProjectForm form. To replace the standard project form you should follow the step sequence:

1. Create in VB IDE a new ActiveX DLL project.

2. Implement at least the IProjectForm interface in your module. The only method you are to write will be ShowProjectForm.

3. Register your custom form as described in the Registering Custom Froms section.

You may use any APIs and interfaces, for example nothing prevents a form to be implemented as a DHTML dialog.

Registering Custom Forms

To register your custom form for DM Extensions versions up to 5.0 you should perform following actions:

1. Create a registration file using the following pattern:

```
REGEDIT4


[HKEY_LOCAL_MACHINE\SOFTWARE\Hummingbird\PowerDOCS\Forms]
;Insert custom form name and its ProgID, e.g.
;"ProjectForm"="ProjectFormPrj.ProjectFormClass"


[HKEY_LOCAL_MACHINE\SOFTWARE\Hummingbird\PowerDOCS\Core\Components
]
;Insert path to PwDRes.dll file, e.g.
;"PwDRes"="C:\\Program Files\\Hummingbird\\Hummingbird DM" _
```

2. Specify a path to the DM installation folder under the key [HKEY_LOCAL_MACHINE\SOFTWARE\Hummingbird\PowerDOCS\Core\Components].

3. Run this file from the Explorer Shell to incorporate settings to the system registry.

4. Copy PwDRes.dll and VBRes.dll file to the DM Extensions installation folder.

5. Register following in-proc servers: VBForm.dll, VHelper.dll and all components implementing customs forms.

For DM Extensions 5.0 or later registration of custom objects is performed through a special object BCRegistrarExt of the Explorer Core.

Custom Forms API Reference

- "IChangeDisplayNameForm" on page 874
- "ICompatibleAppsForm" on page 875
- "IExportForm" on page 877
- "IGroupCheckinForm" on page 880
- "IGroupCheckoutForm" on page 881
- "IHistoryForm" on page 883
- "ILaunchMethodChooserForm" on page 885
- "ILibrariesForm" on page 886
- "ILibraryChooserForm" on page 887
- "ILocationForm" on page 889
- "ILookupForm" on page 890
- "IMailForm" on page 893
- "IProfileAccessForm" on page 894
- "IProfileAccessForm2" on page 896
- "IProfileForm" on page 904\
- "IProfileFormChooser" on page 906
- "IProjectAccessForm" on page 908
- "IProjectAccessForm2" on page 909
- "IProjectChooserForm" on page 917
- "IProjectChooserForm2" on page 919
- "IProjectForm" on page 921
- "IProjectMembershipForm" on page 922
- "IProjectMembershipForm2" on page 924
- "IPropertiesForm" on page 926
- "IQuickRetrieveForm" on page 928

- "IRelatedItemsForm" on page 929
- "ISaveAsOptionsForm" on page 931
- "ISearchAccessForm" on page 933
- "ISearchAccessForm2" on page 934
- "ISearchCriteriaForm" on page 942
- "ISearchFormChooser" on page 944
- "ISearchOptionsForm" on page 945
- "ISelectVersionsForm" on page 947
- "ISelectVersionTypeForm" on page 949
- "ITemplateChooserForm" on page 951
- "IVersionChooserForm" on page 952

IChangeDisplayNameForm

A custom form that shows display name for folder component.

Methods

IChangeDisplayNameForm Methods	Description
IChangeDisplayNameForm::ShowChangeDisplayNameForm	The method displays form that allow to change display name for component.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <FsPForms.dll>

IChangeDisplayNameForm::ShowChangeDisplayNameForm

The method displays form that allow to change display name for component.

C++ Syntax

```
HRESULT ShowChangeDisplayNameForm(  IDOComponent* pDOComponent,
IBCObject* pParentObject,  short* psResult,);
```

Visual Basic Syntax

```
[ psResult  = ] ChangeDisplayNameForm.ShowChangeDisplayNameForm (
pDOComponent, pParentObject)
```

Parameters

pDOComponent	[in] DOComponent object that contain folder components.
pParentObject	[in] A BCObject object that point parent object (Folder or “Public Folders” node).
psResult	[out, retval] Returns an integer that specifies user’s action on the form. For possible values see FormsResults Enumeration.

Return Values:

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <FsPForms.dll>

See Also

“IChangeDisplayNameForm” on page 874

ICompatibleAppsForm

The form that appears when the user is prompted to select a compatible application while opening a document should implement this interface. Such methods asOpenInCompatibleApp onProfile and Version use this interface to display the form for selecting a compatible application.

Methods

ICompatibleAppsForm Methods	Description
ICompatibleAppsForm::ShowCompatibleAppsForm	Displays the form for selecting a compatible application for a specified one

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IProfile" on page 419

"IVersion" on page 625

"ICloseNotify" on page 976

"IIntegrationScheme" on page 978

ICompatibleAppsForm::ShowCompatibleAppsForm

The method displays the form for selecting a compatible application for a specified one and returns the application selected by the user.

C++ Syntax

```
HRESULT ShowCompatibleAppsForm(  BSTR profileApp,  ILibrary*
pLibrary,  BSTR* pCompatApp);
```

Visual Basic Syntax

```
[ pCompatApp  = ] CompatibleAppsForm.ShowCompatibleAppsForm
(profileApp, pLibrary)
```

Parameters

profileApp	[in] A string that specifies the given application identifier, e.g., "WORD", "AMIPRO", or "L123".
pLibrary	[in] A Library object that identifies the library from which applications is to be taken.
pCompatApp	[out, retval] Returns a string with the identifier of the application the user selected or an empty string if the user selected no application.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"ICompatibleAppsForm" on page 875

IExportForm

A form for exporting documents.

Methods

IExportForm Methods	Description
IExportForm::ShowExportForm	Displays a form for export documents
IExportForm::ShowExportFormAndAction	Displays a form for export documents and perform actions

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import <FsPForms.dll>

IExportForm::ShowExportForm

The method displays a form for export documents.

C++ Syntax

```
HRESULT ShowExportForm(  BSTR bstrTitle,  long* pnFlags,
IDocProfiles* pProfiles,  DATE* pdtReturnDate,  BSTR*
pbstrComments,  IVersions** ppVersions);
```

Visual Basic Syntax

```
[ Set ppVersions  = ] ExportForm.ShowExportForm (bstrTitle,
pnFlags, pProfiles, pdtReturnDate, pbstrComments)
```

Parameters

bstrTitle	[in] A string that specifies the form caption.
pnFlags	[in, out] Input form flags. For possible values see ExportInFlags Enumeration. On output contains values from ExportOutFlags.
pProfiles	[in] Profile objects collection that should be exported.
pdtReturnDate	[out] Check-out return date if check-out option was selected.
pbstrComments	[out] Comments if check-out option was selected.
ppVersions	[out, retval] Document's versions collection that was selected for export.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IExportForm" on page 877

IExportForm::ShowExportFormAndAction

The method displays a form for export documents and perform actions.

C++ Syntax

```
HRESULT ShowExportFormAndAction(  BSTR bstrTitle,  long* pnFlags,
IDocProfiles* pProfiles,  VARIANT* pvContents);
```

Visual Basic Syntax

```
[ pvContents  = ] ExportForm.ShowExportFormAndAction (bstrTitle,
pnFlags, pProfiles)
```

Parameters

bstrTitle	[in] A string that specifies the form caption.
pnFlags	[in, out] Input form flags. For possible values see ExportInFlags Enumeration. On output contains values from ExportOutFlags.
pProfiles	[in] Profile objects collection that should be exported.
pvContents	[out, retval] Array that contains IStream objects. Each IStream object contains file name and file contents for exported file.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IExportForm" on page 877

IGroupCheckinForm

A custom form that prompts the user to set check-in options when checking in documents should implement this interface. TheExecuteMenuItems method onDocProfiles when executing the check-in command and theUnlockProfile method on a document profile use this interface to display such a form.

Methods

IGroupCheckinForm Methods	Description
IGroupCheckinForm::ShowGroupCheckinForm	Displays the custom form that prompts the user to set check-in options and unlocks specified documents

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IDocProfiles::ExecuteMenuItems" on page 296

"IProfile::UnlockProfile" on page 466

IGroupCheckinForm::ShowGroupCheckinForm

The method displays the custom form that prompts the user to set check-in options and unlocks specified documents.

C++ Syntax

```
HRESULT ShowGroupCheckinForm( IDocProfiles* pDocProfiles, long
EditTime, BSTR location, short action, IVersion** pVersion);
```

Visual Basic Syntax

```
[ Set pVersion = ] GroupCheckinForm.ShowGroupCheckinForm
(pDocProfiles, EditTime, location, action)
```

Parameters

pDocProfiles	[in] A DocProfiles object that specifies the collection of documents being checked in.
EditTime	[in] A long that specifies how long the profiles have been edited in minutes.
location	[in] A string that specifies the full path (without the file name) to the document versions.
action	[in] An integer value that specifies which action should be performed. The parameter can be one of the CheckInActions values.
pVersion	[out, retval] Returns a Version object that specifies the current version of the last unlocked document. If the operation was canceled, an error occurred, or the UF_UNLOCK flag is specified the return value is NULL.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IGroupCheckinForm" on page 880

IGroupCheckoutForm

A custom form that prompts the user to set check-out options when checking out documents should implement this interface.

TheExecuteMenuItems (when executing the check-out command) andCheckOut methods onDocProfiles and theLockProfile method on a document profile use this interface to display such a form.

Methods

IGroupCheckoutForm Methods	Description
IGroupCheckoutForm::ShowGroupCheckoutForm	Displays the custom form that prompts the user to set check-out options and locks specified documents

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IDocProfiles::CheckOut" on page 294

"IDocProfiles::ExecuteMenuItems" on page 296

"IProfile::LockProfile" on page 439

IGroupCheckoutForm::ShowGroupCheckoutForm

The method displays the custom form that prompts the user to set check-out options and locks specified documents.

C++ Syntax

```
HRESULT ShowGroupCheckoutForm( IDocProfiles* pDocProfiles, DATE
returnDate, BSTR comments, short* pResult);
```

Visual Basic Syntax

```
[ pResult = ] GroupCheckoutForm.ShowGroupCheckoutForm
(pDocProfiles, returnDate, comments)
```

Parameters

pDocProfiles	[in] A DocProfiles object that specifies the collection of documents being checked out.
returnDate	[in] Required. A date that specifies when the profiles is to be checked in.
comments	[in] Required. A string of comments.
pResult	[out, retval] Returns an integer that specifies user's action on the form. For possible values see FormsResults Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IGroupCheckoutForm" on page 881

IHistoryForm

A custom form that presents data from the activity log of a document should implement this interface. TheShowHistoryForm method on a document profile uses the interface to display such a form.

Methods

IHistoryForm Methods	Description
IHistoryForm::ShowHistoryForm	Displays the history form that presents data from the activity log of a specified document

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IProfile::ShowHistoryForm" on page 458

IHistoryForm::ShowHistoryForm

The method displays the history form that presents data from the activity log of a specified document.

C++ Syntax

```
HRESULT ShowHistoryForm(  IProfile* pProfile);
```

Visual Basic Syntax

```
HistoryForm.ShowHistoryForm (pProfile)
```

Parameters

pProfile	[in] A Profile object that specifies the document whose activity log is to be displayed.

Return Values

S_OK	The operation succeeded.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IHistoryForm" on page 883

ILaunchMethodChooserForm

A custom form that prompts the user to select a launch method of an application should implement the interface. TheShowLaunchMethodChooserForm method on anAppInfo object uses the interface to display such a form.

Methods

ILaunchMethodChooserForm Methods	Description
ILaunchMethodChooserForm::ShowLaunchMethodChooserForm	Displays the custom form that prompts the user to select a launch method of a specified application and returns the launch method selected by the user

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IAppInfo::ShowLaunchMethodChooserForm" on page 232

ILaunchMethodChooserForm::ShowLaunchMethodChooserForm

The method displays the custom form that prompts the user to select a launch method of a specified application and returns the launch method selected by the user.

C++ Syntax

```
HRESULT ShowLaunchMethodChooserForm(  long Flags,  IAppInfo*
pAppInfo,  ILaunchMethod** pSelected);
```

VBScript Syntax

```
[ Set pSelected  = ]
LaunchMethodChooserForm.ShowLaunchMethodChooserForm (Flags,
pAppInfo)
```

Parameters

Flags	[in] A long that specifies how the method should be executed. The parameter can be a combination of the LaunchMethodChooserFlags values.
pAppInfo	[in] A AppInfo object that specifies the application whose launch method the user should select.
pSelected	[out, retval] Returns a LaunchMethod object that specifies the launch method selected by the user. If the user selected no launch method, the return value is NULL.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"ILaunchMethodChooserForm" on page 885

ILibrariesForm

A custom form that appears when the user is prompted to select libraries searchable by default and to set the current library should implement this interface. TheShowLibrariesForm method on theApplication object uses the interface to display such a form.

Methods

ILibrariesForm Methods	Description
ILibrariesForm::ShowLibrariesForm	Displays the custom form for selecting search libraries

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IApplication::ShowLibrariesForm" on page 250

ILibrariesForm::ShowLibrariesForm

The method displays the custom form for selecting search libraries.

C++ Syntax

```
HRESULT ShowLibrariesForm();
```

Visual Basic Syntax

```
LibrariesForm.ShowLibrariesForm
```

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"ILibrariesForm" on page 886

ILibraryChooserForm

The form that appears when the user is prompted to select a library should implement this interface. TheShowLibraryChooserForm method on theApplication object uses this interface to display the form.

Methods

ILibraryChooserForm Methods	Description
ILibraryChooserForm::LibraryChooserForm	Displays the form for selecting a library and returns the library selected by the user

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IApplication::ShowLibraryChooserForm" on page 252

ILibraryChooserForm::LibraryChooserForm

The method displays the form for selecting a library and returns the library selected by the user.

C++ Syntax

```
HRESULT LibraryChooserForm(  ILibrary** ppLibrary);
```

Visual Basic Syntax

```
[ Set ppLibrary  = ] LibraryChooserForm.LibraryChooserForm
```

Parameters

ppLibrary	[out, retval] Returns a Library object that specifies the library selected by the user. If the user selected no library by pressing a Cancel button, the return value is NULL.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"ILibraryChooserForm" on page 887

ILocationForm

A form for selection library and projects.

Methods

ILocationForm Methods	Description
ILocationForm::ShowLocationForm	Displays a form for selecting a location in library

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

ILocationForm::ShowLocationForm

The method displays a form for selecting a location in library.

C++ Syntax

```
HRESULT ShowLocationForm(  BSTR bstrTitle,  ILibrary** ppLirary,
short* pReturn);
```

Visual Basic Syntax

```
[ pReturn  = ] LocationForm.ShowLocationForm (bstrTitle, ppLirary)
```

Parameters

bstrTitle	[in] A string that specifies the form caption.
ppLirary	[in, out] On input - library for initial selection. On output - selected library.
pReturn	[out, retval] Returns an integer that specifies user's action on the form. For possible values see FormsResults Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"ILocationForm" on page 889

ILookupForm

A custom lookup form should implement this interface. The DoLookup method on aColumn object uses the interface to display such a form.

Methods

ILookupFormesForm Methods	Description
ILookupForm::DoLookup	Displays a lookup form associated with a specified column, validates the column values, or auto-completes them

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IColumn::DoLookup" on page 271

ILookupForm::DoLookup

The method displays a lookup form associated with a specified column, validates the column values, or auto-completes them.

C++ Syntax

```
HRESULT DoLookup(  VARIANT_BOOL bAllowMultipleSelection,
IColumns* pColumns,  BSTR libName,  BSTR formName,  BSTR lookupID,
BSTR targetProp,  short lookupMode,  IBCCoreCtx** pCtx);
```

Visual Basic Syntax

```
[ Set pCtx  = ] LookupForm.DoLookup (bAllowMultipleSelection,
pColumns, libName, formName, lookupID, targetProp, lookupMode)
```

Parameters

bAllowMultipleSelection	[in] A Boolean value that indicates whether the form allows multiple selection or only one item can be highlighted. True: The form allows multiple selection. False: Only one item can be highlighted in the form.
pColumns	[in] A Columns object that specifies all the columns on the form.
libName	[in] A string that specifies the library name.
formName	[in] A string that indicates the name of the form where the lookup is launched.
lookupID	[in] A string that indicates the lookup name.
targetProp	[in] A string that specifies the name of the column for which the lookup is launched.
lookupMode	[in] An integer value that specifies how the method should be executed. The parameter can be one of the OMFlagsLookUp values.
pCtx	[out, retval] Returns a BCCoreCtx object that specifies which columns have changed. The context object contains the key ChangedColumns that lists the names of all columns that have to be changed as a result of calling DoLookup. For each name in ChangedColumns there exists a registered string with a key being the column name and the value equal to its new value (assigned by the lookup).

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

The method doesn't change column values. The developer has to change them manually using the Value property.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"ILookupForm" on page 890

"IColumn::DoLookup" on page 271

IMailForm

A custom form that appears when sending documents by e-mail should implement this interface. TheExecuteMenuItems method onDocProfiles uses this interface to display the form when executing the mail command.

Methods

IMailForm Methods	Description
IMailForm::ShowMailForm	Displays the form for e-mailing a collection of documents

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IDocProfiles::ExecuteMenuItems" on page 296

IMailForm::ShowMailForm

The method displays the form for e-mailing a collection of documents.

C++ Syntax

```
HRESULT ShowMailForm(  IDocProfiles* pDocProf,  short* pResult);
```

Visual Basic Syntax

```
[ pResult  = ] MailForm.ShowMailForm (pDocProf)
```

Parameters

pDocProf	[in] A DocProfiles object that specifies the collection of documents being sending by e-mail.
pResult	[out, retval] Returns an integer that specifies user's action on the form. For possible values see FormsResults Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IMailForm" on page 893

IProfileAccessForm

The interface should be implemented by a custom document's access control form. TheShowAccessForm method on aProfile object uses the interface to display such a form.

Methods

IProfileAccessForm Methods	Description
IProfileAccessForm::ShowProfileAccessForm	Displays the document's access control form

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IProfile::ShowAccessForm" on page 456

IProfileAccessForm::ShowProfileAccessForm

The method displays the document's access control form.

C++ Syntax

```
HRESULT ShowProfileAccessForm(  IProfile* prof,  VARIANT_BOOL
bAllowEdits,  short* pResult);
```

Visual Basic Syntax

```
[ pResult  = ] ProfileAccessForm.ShowProfileAccessForm (prof,
bAllowEdits)
```

Parameters

prof	[in] A Profile object that specifies the document's profile whose access control form is to be displayed.
bAllowEdits	[in] A Boolean value that specifies whether or not the document's trustee list can be changed. True: Trustee list can be changed. False: Trustee list can't be changed.
pResult	[out, retval] Returns an integer that specifies user's action on the form. For possible values see FormsResults Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IProfileAccessForm" on page 894

IProfileAccessForm2

The IProfileAccessForm2 interface extends functionality of ProfileAccessForm object

Properties

IProfileAccessForm2 Properties	Description
IProfileAccessForm2::CanAdd	Disable/Enable Add button.
IProfileAccessForm2::CanRemove	Disable/Enable Remove button
IProfileAccessForm2::EnabledRights	Disable/Enable change specific rights.
IProfileAccessForm2::HiddenTrustees	Hidden trustees collection.

IProfileAccessForm2 Properties	Description
IProfileAccessForm2::RestrictedTrustees	Trustees collection with restricted access for changing.
IProfileAccessForm2::RightsTemplates	Rights templates collection.
IProfileAccessForm2::VisibleRights	Show/hide specific right check boxes.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <FsPForms.dll>

See Also

"IProfileAccessForm" on page 894

IProfileAccessForm2::CanAdd

Disable/Enable Add button.

C++ Syntax

```
HRESULT get_CanAdd(  VARIANT_BOOL* pVal);
HRESULT put_CanAdd(  VARIANT_BOOL Val);
```

Visual Basic Syntax

```
[ pVal = ] ProfileAccessForm2.CanAdd
ProfileAccessForm2.CanAdd [ =  Val ]
```

Parameters

pVal	[out, retval] Returns a boolean value that indicates whether or not user can add users or groups to trustee's list.
Val	[in] Enable or disable user ability add users or groups to trustee's list.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <FsPForms.dll>

IProfileAccessForm2::CanRemove

Disable/Enable Remove button.

C++ Syntax

```
HRESULT get_CanRemove(  VARIANT_BOOL* pVal);
HRESULT put_CanRemove(  VARIANT_BOOL Val);
```

Visual Basic Syntax

```
[ pVal = ] ProfileAccessForm2.CanRemove
ProfileAccessForm2.CanRemove [ = Val ]
```

Parameters

pVal	[out, retval] Returns a boolean value that indicates whether or not user can remove users or groups from trustee's list.
Val	[in] Enable or disable user ability remove users or groups to trustee's list.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <FsPForms.dll>

IProfileAccessForm2::EnabledRights

Disable/Enable change specific rights.

C++ Syntax

```
HRESULT get_EnabledRights(  long* pVal);
HRESULT put_EnabledRights(  long Val);
```

Visual Basic Syntax

```
[ pVal = ] ProfileAccessForm2.EnabledRights
ProfileAccessForm2.EnabledRights [ = Val ]
```

Parameters

pVal	[out, retval] Returns a rights mask that can be modified by user. For possible values see ProfileRights Enumeration.
Val	[in] Enable or disable user ability to change specific rights. For possible values see ProfileRights Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <FsPForms.dll>

See Also

"ProfileRights Enumeration" on page 695

IProfileAccessForm2::HiddenTrustees

Hidden trustees collection.

C++ Syntax

```
HRESULT get_HiddenTrustees(  ITrusteeCollection* ppVal);
```

Visual Basic Syntax

```
[ Set ppVal = ] ProfileAccessForm2.HiddenTrustees
```

Parameters

ppVal	[out, retval] Returns collection of IProfileTrustee objects that should not be displayed. This collection may be modified to hide specific users or group.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <FsPForms.dll>

See Also

"ITrusteeCollection" on page 609

"IProfileTrustee" on page 474

IProfileAccessForm2::RestrictedTrustees

Trustees collection with restricted access for changing.

C++ Syntax

```
HRESULT get_RestrictedTrustees(  ITrusteeCollection* ppVal);
```

Visual Basic Syntax

```
[ Set ppVal = ] ProfileAccessForm2.RestrictedTrustees
```

Parameters

ppVal	[out, retval] Returns collection of IProfileTrustee objects that should not be modified by user. This collection may be modified to limit access for specific users or group.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <FsPForms.dll>

See Also

"ITrusteeCollection" on page 609

"IProfileTrustee" on page 474

IProfileAccessForm2::RightsTemplates

Rights templates collection.

C++ Syntax

```
HRESULT get_RightsTemplates (  ITrusteeCollection* ppVal);
```

Visual Basic Syntax

```
[ Set ppVal = ] ProfileAccessForm2.RightsTemplates
```

Parameters

ppVal	[out, retval] Returns collection of IProfileTrustee objects that contains templates for set rights. By default the collection contains the following items: “View Profile”, “Read Only”, “Normal Access”, “Full Access”.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <FsPForms.dll>

See Also

“ITrusteeCollection” on page 609

“IProfileTrustee” on page 474

IProfileAccessForm2::VisibleRights

Show/hide specific right check boxes.

C++ Syntax

```
HRESULT get_VisibleRights(  long* pVal);

HRESULT put_VisibleRights(  long Val);
```

Visual Basic Syntax

```
[ pVal = ] ProfileAccessForm2.VisibleRights

ProfileAccessForm2.VisibleRights [ = Val ]
```

Parameters

pVal	[out, retval] Returns a rights mask that visible to user. For possible values see ProfileRights Enumeration.
Val	[in] Hide or display specific rights. For possible values see ProfileRights Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <FsPForms.dll>

See Also

"ProfileRights Enumeration" on page 695

IProfileForm

A custom profile form should implement this interface. The ShowProfile method on a document profile uses the interface to display such a form.

Methods

IProfileForm Methods	Description
IProfileForm::ShowProfileForm	Displays the profile form for a specified document

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IProfile::ShowProfile" on page 458

IProfileForm::ShowProfileForm

The method displays the profile form for a specified document.

C++ Syntax

```
HRESULT ShowProfileForm(  IProfile* p,  short flags,  short*
pResult);
```

Visual Basic Syntax

```
[ pResult  = ] ProfileForm.ShowProfileForm (p, flags)
```

Parameters

p	[in] A Profile object that specifies the document whose data should appear on the form.
flags	[in] Required. An integer value that specifies how the method should be executed. The parameter can be a combination of the ShowProfileFlags values.
pResult	[out, retval] Returns an integer that specifies user's action on the form. For possible values see FormsResults Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IProfileForm" on page 904

IProfileFormChooser

A custom form that prompts the user to select a profile form should implement the interface. TheSelectProfileForm method on a Library object uses the interface to display such a form and to create a new profile.

Methods

IProfileFormChooser Methods	Description
IProfileFormChooser::SelectProfileForm	Creates an empty document profile on a library by the profile form selected by the user and by a specific application

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"ILibrary::SelectProfileForm" on page 392

IProfileFormChooser::SelectProfileForm

The method creates an empty document profile on a library by the profile form selected by the user and by a specific application.

C++ Syntax

```
HRESULT SelectProfileForm(  ILibrary* pLibrary,  short sFlags,
VARIANT info,  IProfile** pProfile);
```

Visual Basic Syntax

```
[ Set pProfile  = ] ProfileFormChooser.SelectProfileForm
(pLibrary, sFlags, info)
```

Parameters

pLibrary	[in] A Library object that identifies the library on which a new profile is to be created.
sFlags	[in] For possible values see SPFFlags Enumeration.
info	[in] An expression that specifies the application the Profile object will be associated with. If a string expression, info must specify either the application extension or the application identifier. An empty string identifies profile paper documents if the bFromFile flag is False. If True, the user will be able to specify the application when filling out the profile form. If an object, info must specify a validAppInfo object that identifies an application.
pProfile	[out, retval] Returns a Profile object that specifies the created profile. If the method failed to create a new profile (for instance, if the user pressed a Cancel button), the return value is NULL.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

The method creates a blank Profile object. Before creating the profile, a custom form should pop up where the user can select a profile form associated with the specified application the profile will be based on.

The created Profile object will be associated with the specified application. It should be pre-filled out with the default profile values

for the current user. Besides, the typist field should be also set to the current user. Values must be assigned for all required fields in the profile before it can be saved.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IProfileFormChooser" on page 906

IProjectAccessForm

Obsolete. IProfileAccessForm is used to display a folder access form.

The interface should be implemented by a custom project's access control form. TheShowAccessForm method on aProject object uses the interface to display such a form.

Methods

IProjectAccessForm Methods	Description
IProjectAccessForm::ShowProjectAccessForm	Displays the project's access control form

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IProject::ShowAccessForm" on page 517

IProjectAccessForm::ShowProjectAccessForm

The method displays the project's access control form.

C++ Syntax

```
HRESULT ShowProjectAccessForm(  IProject* proj,  VARIANT_BOOL
bAllowEdits,  short* pResult);
```

Visual Basic Syntax

```
[ pResult  = ] ProjectAccessForm.ShowProjectAccessForm (proj,
bAllowEdits)
```

Parameters

proj	[in] A Project object that specifies the project whose access control form is to be displayed.
bAllowEdits	[in] A Boolean value that specifies whether or not the project's trustee list can be changed. True: Trustee list can be changed. False: Trustee list can't be changed.
pResult	[out, retval] Returns an integer that specifies user's action on the form. For possible values see FormsResults Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IProjectAccessForm" on page 908

IProjectAccessForm2

Obsolete. IProfileAccessForm2 is used to display a profiled folder access form.

The IProjectAccessForm2 interface extends functionality of ProjectAccessForm object

Properties

IProjectAccessForm2 Properties	Description
IProjectAccessForm2::CanAdd	Disable/Enable Add button.
IProjectAccessForm2::CanRemove	Disable/Enable Remove button

IProjectAccessForm2 Properties	Description
IProjectAccessForm2::EnabledRights	Disable/Enable change specific rights.
IProjectAccessForm2::HiddenTrustees	Hidden trustees collection.
IProjectAccessForm2::RestrictedTrustees	Trustees collection with restricted access for changing.
IProjectAccessForm2::RightsTemplates	Rights templates collection.
IProjectAccessForm2::VisibleRights	Show/hide specific right check boxes.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <FsPForms.dll>

See Also

"IProjectAccessForm" on page 908

IProjectAccessForm2::CanAdd

Disable/Enable Add button.

C++ Syntax

```
HRESULT get_CanAdd(  VARIANT_BOOL* pVal);
HRESULT put_CanAdd(  VARIANT_BOOL Val);
```

Visual Basic Syntax

```
[ pVal = ] ProjectAccessForm2.CanAdd
ProjectAccessForm2.CanAdd[ =  Val]
```

Parameters

pVal	[out, retval] Returns a boolean value that indicates whether or not user can add users or groups to trustee's list.
Val	[in] Enable or disable user ability add users or groups to trustee's list.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <FsPForms.dll>

IProjectAccessForm2::CanRemove

Disable/Enable Remove button.

C++ Syntax

```
HRESULT get_CanRemove(  VARIANT_BOOL* pVal);
HRESULT put_CanRemove(  VARIANT_BOOL Val);
```

Visual Basic Syntax

```
[ pVal = ] ProjectAccessForm2.CanRemove
ProjectAccessForm2.CanRemove [ = Val ]
```

Parameters

pVal	[out, retval] Returns a boolean value that indicates whether or not user can remove users or groups from trustee's list.
Val	[in] Enable or disable user ability remove users or groups to trustee's list.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <FsPForms.dll>

IProjectAccessForm2::EnabledRights

Disable/Enable change specific rights.

C++ Syntax

```
HRESULT get_EnabledRights(  long* pVal);
HRESULT put_EnabledRights(  long Val);
```

Visual Basic Syntax

```
[ pVal = ] ProjectAccessForm2.EnabledRights
ProjectAccessForm2.EnabledRights [ = Val ]
```

Parameters

pVal	[out, retval] Returns a rights mask that can be modified by user. For possible values see ProjectRights Enumeration.
Val	[in] Enable or disable user ability to change specific rights. For possible values see ProjectRights Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <FsPForms.dll>

See Also

"ProjectRights Enumeration" on page 697

IProjectAccessForm2::HiddenTrustees

Hidden trustees collection.

C++ Syntax

```
HRESULT get_HiddenTrustees(  ITrusteeCollection* ppVal);
```

Visual Basic Syntax

```
[ Set ppVal = ] ProjectAccessForm2.HiddenTrustees
```

Parameters

ppVal	[out, retval] Returns collection of IProjectTrustee objects that should not be displayed. This collection may be modified to hide specific users or group.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <FsPForms.dll>

See Also

"ITrusteeCollection" on page 609

"IProjectTrustee" on page 540

IProjectAccessForm2::RestrictedTrustees

Trustees collection with restricted access for changing.

C++ Syntax

```
HRESULT get_RestrictedTrustees(  ITrusteeCollection* ppVal);
```

Visual Basic Syntax

```
[ Set ppVal = ] ProjectAccessForm2.RestrictedTrustees
```

Parameters

ppVal	[out, retval] Returns collection of IProjectTrustee objects that should not be modified by user. This collection may be modified to limit access for specific users or group.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <FsPForms.dll>

See Also

"ITrusteeCollection" on page 609

"IProjectTrustee" on page 540

IProjectAccessForm2::RightsTemplates

Rights templates collection.

C++ Syntax

```
HRESULT get_RightsTemplates(  ITrusteeCollection* ppVal);
```

Visual Basic Syntax

```
[ Set ppVal = ] ProjectAccessForm2.RightsTemplates
```

Parameters

ppVal	[out, retval] Returns collection of IProjectTrustee objects that contains templates for set rights. By default the collection contains the following items: “View Project”, “Read Only”, “Normal Access”, “Full Access”.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <FsPForms.dll>

See Also

“ITrusteeCollection” on page 609

“IProjectTrustee” on page 540

IProjectAccessForm2::VisibleRights

Show/hide specific right check boxes.

C++ Syntax

```
HRESULT get_VisibleRights(  long* pVal);

HRESULT put_VisibleRights(  long Val);
```

Visual Basic Syntax

```
[ pVal = ] ProjectAccessForm2.VisibleRights

ProjectAccessForm2.get_VisibleRights [ = Val ]
```

Parameters

pVal	[out, retval] Returns a rights mask that visible to user. For possible values see ProjectRights Enumeration.
Val	[in] Hide or display specific rights. For possible values see ProjectRights Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <FsPForms.dll>

See Also

"ProjectRights Enumeration" on page 697

IProjectChooserForm

Obsolete. QuickRetrieve form is used to choose profiled folders.

A custom form that prompts the user to select a project (or projects if multiple selection is allowed) should implement the interface. TheShowProjectChooserForm method on theApplication object uses the interface to display such a form.

Methods

IProjectChooserForm Methods	Description
IProjectChooserForm::ShowProjectChooserForm	Displays the form for selecting a project (or projects if multiple selection is allowed) and returns a collection of the projects selected by the user

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IApplication::ShowProjectChooserForm" on page 253

IProjectChooserForm::ShowProjectChooserForm

The method displays the form for selecting a project (or projects if multiple selection is allowed) and returns a collection of the projects selected by the user.

C++ Syntax

```
HRESULT ShowProjectChooserForm(  short flags,  IProjects**
pSelectedProjects);
```

Visual Basic Syntax

```
[ Set pSelectedProjects  = ]
ProjectChooserForm.ShowProjectChooserForm (flags)
```

Parameters

flags	[in] An integer value that specifies the form appearance. The parameter can be a combination of the ProjectChooserFlags values.
pSelectedProjects	[out, retval] Returns a Projects object that specifies the collection of all the projects selected by the user. If the user selected no projects, the return value is NULL.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IProjectChooserForm" on page 917

IProjectChooserForm2

Obsolete.

A custom form that prompts the user to select a project (or projects if multiple selection is allowed) should implement the interface.

The IProjectChooserForm2 interface extends functionality of ProjectChooserForm object

Methods

IProjectChooserForm2 Methods	Description
IProjectChooserForm2::SelectProjectForCriteria	The method displays the form for selecting a project (or projects if multiple selection is allowed) and returns a collection of the projects selected by the user. Projects will be displayed according to criteria specified in pSearchCtx parameter.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IProjectChooserForm" on page 917

IProjectChooserForm2::SelectProjectForCriteria

The method displays the form for selecting a project (or projects if multiple selection is allowed) and returns a collection of the projects selected by the user. Projects will be displayed according to criteria specified in pSearchCtx parameter.

C++ Syntax

```
HRESULT SelectProjectForCriteria(  short flags,  IBCCoreCtx*
pSearchCtx,  IProjects** pSelectedProjects);
```

Visual Basic Syntax

```
[Set pSelectedProjects =
```

```
]ProjectChooserForm2.SelectProjectForCriteria(flags, pSearchCtx)
```

Parameters

flags	[in] Flags parameter as specified in ProjectChooserFlags.
pSearchCtx	[in] Context for search. The following values are passed to this context: SearchLibraries: A String that represents list of libraries for search delimited by semicolon. Criteria: A String that represents list of criteria for search delimited by semicolon. <CriteriaX>: A String that contains criteria for specific column. OrderBy: A String that contains list of columns that should be sorted for.
pSelectedProjects	[out, retval] Returns a Projects object that specifies the collection of all the projects selected by the user. If the user selected no projects, the return value is NULL.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import <FsPForms.dll>

See Also

"ICompatibleAppsForm" on page 875

IProjectForm

A custom form that presents properties of a folder and its contents should implement the interface. TheDisplayModal method on a folder uses the interface to display such a form.

Methods

IProjectForm Methods	Description
IProjectForm::ShowProjectForm	Displays the folder form to the user

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IProject::DisplayModal" on page 504

IProjectForm::ShowProjectForm

The method displays the folder form to the user.

C++ Syntax

```
HRESULT ShowProjectForm(  VARIANT_BOOL bAllowEdits,  IProject*
pProject,  short* pResult);
```

Visual Basic Syntax

```
[ pResult  = ] ProjectForm.ShowProjectForm (bAllowEdits, pProject)
```

Parameters

bAllowEdits	[in] A Boolean value that specifies whether or not folder properties can be changed. True: Folder properties can be changed. False: Folder properties can't be changed.
pProject	[in] A Folder object whose form is to pop up.
pResult	[out, retval] Returns an integer that specifies user's action on the form. For possible values see FormsResults Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IProjectForm" on page 921

IProjectMembershipForm

A custom form that presents a list of folders and complex documents a profile belongs to should implement the interface. TheShowProjectsMembershipForm method on the profile uses the interface to display such a form.

Methods

IProjectMembershipForm Methods	Description
IProjectMembershipForm::ShowProjectMembershipForm	Displays a list of folders and complex documents that contain a specified profile item.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IProfile::ShowProjectsMembershipForm" on page 460

IProjectMembershipForm::ShowProjectMembershipForm

The method displays a list of folders and complex documents that contain a specified profile item.

C++ Syntax

```
HRESULT ShowProjectMembershipForm(  IProfile* p,  short*
pResults);
```

VBScript Syntax

```
[ pResults  = ] ProjectMembershipForm.ShowProjectMembershipForm (
p )
```

Parameters

p	[in] A Profile object that specifies the document's profile that belongs to folders and/or complex documents.
pResults	[out, retval] Returns an integer that specifies user's action on the form. For possible values see FormsResults Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IProjectMembershipForm" on page 922

IProjectMembershipForm2

IProjectMembershipForm2 extends IProjectMembershipForm. It allows security flags to be passed to a custom form. Unlike prior versions, starting with version 5.1 DM Extensions allows the project membership list to be modified. This interface controls security so that only authorized users can modify the project membership list.

Methods

IProjectMembershipForm2 Methods	Description
IProjectMembershipForm2::ShowProjectMembershipForm	Displays a list of folders and complex documents that contain a specified profile item.

Requirements

DM: Requires DM Extensions 5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IProfile::ShowProjectsMembershipForm" on page 460

IProjectMembershipForm2::ShowProjectMembershipForm

This method displays a list of folders and complex documents that contain a specified profile item.

C++ Syntax

```
HRESULT ShowProjectMembershipForm(  IProfile* p,  short* pResults,
short sFlags);
```

VBScript Syntax

```
[ pResults  = ] ProjectMembershipForm2.ShowProjectMembershipForm (
p, sFlags )
```

Parameters

p	[in] A Profile object that specifies the document's profile that belongs to folders and/or complex documents.
sFlags	[in] Security flags. For possible values see ShowProfileFlags Enumeration.
pResults	[out, retval] Returns an integer that specifies user's action on the form. For possible values see FormsResults Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IProjectMembershipForm2" on page 924

IPropertiesForm

The form that appears as the version/attachment properties dialog should implement this interface. SuchVersion methods as ShowProperties, SaveAsNewVersion, etc. use the interface to display the form.

Methods

IPropertiesForm Methods	Description
IPropertiesForm::Show	Displays the version/attachment properties dialog form

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IVersion::ShowProperties" on page 652

"IVersion::SaveAsNewVersion" on page 650

IPropertiesForm::Show

The method displays the version/attachment properties dialog form.

C++ Syntax

```
HRESULT Show( short Flags, IProfile* pProfile, BSTR*
VersionComments, BSTR* Author, BSTR* Typist, BSTR*
VersionLabel, short* pResult);
```

Visual Basic Syntax

```
[pResult = ]PropertiesForm.Show(Flags, pProfile, VersionComments,
Author, Typist, VersionLabel)
```

Parameters

Flags	[in] An integer value that specifies the form appearance. The parameter can be a combination of the VersionPropertiesFlags values.
pProfile	[in] A Profile object that specifies the document the version belongs to.
VersionComments	[in, out] A string that indicates the version comments.
Author	[in, out] A string of the version author's name.
Typist	[in, out] A string of the version typist's name.
VersionLabel	[in, out] A string that specifies the version label. The parameter is used when creating a new subversion or attachment. In that case, for subversions it must contain the number of its parent version (e.g., "1"). For attachments VersionLabel must specify the attachment label. The attachment label is a three characters long string. The first character must be in the A-Z (uppercase) range. The others must be any uppercase Latin letter or digit.
pResult	[out, retval] Returns an integer that specifies user's action on the form. For possible values see FormsResults Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IPropertiesForm" on page 926

IQuickRetrieveForm

A custom Quick Retrieve form should implement this interface. The DisplayModal method on aQuickRetrieve object uses the interface to display such a form.

Methods

IQuickRetrieveForm Methods	Description
IQuickRetrieveForm::ShowQuickRetrieveForm	Displays the Quick Retrieve window

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IQuickRetrieve::DisplayModal" on page 559

IQuickRetrieveForm::ShowQuickRetrieveForm

The method displays the Quick Retrieve window.

C++ Syntax

```
HRESULT ShowQuickRetrieveForm(  IQuickRetrieve* pQR,  short flags,
short* pResult);
```

Visual Basic Syntax

```
[pResult = ]QuickRetrieveForm.ShowQuickRetrieveForm(pQR, flags)
```

Parameters

pQR	[in] A QuickRetrieve object that is to be displayed.
flags	[in] An integer value that specifies the form appearance. The parameter can be one of the OMFlagsQR values.
pResult	[out, retval] Returns an integer that specifies user's action on the form. For possible values see FormsResults Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_FAIL	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IQuickRetrieveForm" on page 928

IRelatedItemsForm

A custom form that displays related profiled items (documents, paper profile documents, folders). The ShowRelatedItemsForm method on a IDocProfiles2 object uses the interface to display such a form.

Methods

IRelatedItemsForm Methods	Description
IRelatedItemsForm::ShowRelatedItemsForm	The method displays the related items form.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <FsPForms.dll>

IRelatedItemsForm::ShowRelatedItemsForm

The method displays the related items form.

C++ Syntax

```
HRESULT ShowRelatedItemsForm(  short flags,  IDocProfiles*
pDocProfiles,  short* pResult);
```

Visual Basic Syntax

```
[pResult = ] RelatedItemsForm.ShowRelatedItemsForm(flags,
pDocProfiles)
```

Parameters

flags	[in] Reserved. Should be 0..
pDocProfiles	[in] A DocProfiles object that contain profiled items.
pResult	[out, retval] Returns an integer that specifies user's action on the form. For possible values see FormsResults Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <FsPForms.dll>

See Also

"IRelatedItemsForm" on page 929

ISaveAsOptionsForm

The custom Save As options form should implement the interface. The ShowSaveAsOptionsForm methods on a document profile and version use the interface to display such a form.

Methods

ISaveAsOptionsForm Methods	Description
ISaveAsOptionsForm::ShowSaveAsOptionsForm	Displays the custom Save As options form and allows the user to select an option

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IProfile::ShowSaveAsOptionsForm" on page 461

ISaveAsOptionsForm::ShowSaveAsOptionsForm

The method displays the custom Save As options form and allows the user to select an option.

C++ Syntax

```
HRESULT ShowSaveAsOptionsForm(  short* Flags,  IVersion* pVersion,
IVersion** ppNewVer);
```

Visual Basic Syntax

```
[ ppNewVer  = ] SaveAsOptionsForm.ShowSaveAsOptionsForm (Flags,
pVersion)
```

Parameters

Flags	[in,out] An integer value that specifies how the method should be executed. The parameter can be zero or a combination of theSAOFFlags values. On return this parameter specifies user's action on the form. For possible values seeFormsResults.
pVersion	[in] A Version object that specifies the version, which is to be saved as.
ppNewVer	[out, retval] Returns a Version object that specifies the saved version. It may be the same version as pVersion, a new version/subversion of the same document or a new document. If the operation was canceled or an error occurred, the return value is NULL.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Remarks

This method allows the user to select Save As options. The options given to the user are saved as a new document, as a new version, as a new subversion, and replace original one.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"ISaveAsOptionsForm" on page 931

ISearchAccessForm

The interface should be implemented by the custom search's access control form. The ShowAccessForm method on aSearch object uses the interface to display such a form.

Methods

ISearchAccessForm Methods	Description
ISearchAccessForm::ShowSearchAccessForm	Displays the search's access control form

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"ISearch::ShowAccessForm" on page 586

ISearchAccessForm::ShowSearchAccessForm

The method displays the search's access control form.

C++ Syntax

```
HRESULT ShowSearchAccessForm(  ISearch* pS,  VARIANT_BOOL
bAllowEdits,  short* pResult);
```

Visual Basic Syntax

```
[ pResult  = ] SearchAccessForm.ShowSearchAccessForm (pS,
bAllowEdits)
```

Parameters

pS	[in] A Search object that specifies the quick search whose access control form is to be displayed.
bAllowEdits	[in] A Boolean value that specifies whether or not the search's trustee list can be changed. True: Trustee list can be changed. False:Trustee list cannot be changed.
pResult	[out, retval] Returns an integer that specifies user's action on the form. For possible values see FormsResults Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"ISearchAccessForm" on page 933

ISearchAccessForm2

The ISearchAccessForm2 interface extends functionality of the SearchAccessForm object.

Properties

ISearchAccessForm2 Properties	Description
ISearchAccessForm2::CanAdd	Disable/Enable the Add button.
ISearchAccessForm2::CanRemove	Disable/Enable the Remove button

ISearchAccessForm2 Properties	Description
ISearchAccessForm2::EnabledRights	Disable/Enable changing specific rights.
ISearchAccessForm2::HiddenTrustees	Hidden trustees collection.
ISearchAccessForm2::RestrictedTrustees	Trustees collection with restricted access for changing.
ISearchAccessForm2::RightsTemplates	Rights templates collection.
ISearchAccessForm2::VisibleRights	Show/hide specific right check boxes.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <FsPForms.dll>

See Also

"ISearchAccessForm" on page 933

ISearchAccessForm2::CanAdd

Disable/Enable the Add button.

C++ Syntax

```
HRESULT get_CanAdd(  VARIANT_BOOL* pVal);
HRESULT put_CanAdd(  VARIANT_BOOL Val);
```

Visual Basic Syntax

```
[ pVal = ] SearchAccessForm2.CanAdd
SearchAccessForm2.CanAdd [ = Val ]
```

Parameters

pVal	[out, retval] Returns a Boolean value that indicates whether or not a user can add users or groups to trustee's list.
Val	[in] Enable or disable user's ability to add users or groups to trustee's list.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <FsPForms.dll>

ISearchAccessForm2::CanRemove

Disable/Enable the Remove button.

C++ Syntax

```
HRESULT get_CanRemove(  VARIANT_BOOL* pVal);
HRESULT put_CanRemove(  VARIANT_BOOL Val);
```

Visual Basic Syntax

```
[ pVal = ] SearchAccessForm2.CanRemove
SearchAccessForm2.CanRemove [ = Val ]
```

Parameters

pVal	[out, retval] Returns a Boolean value that indicates whether or not a user can remove users or groups from trustee's list.
Val	[in] Enable or disable user's ability to remove users or groups from trustee's list.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <FsPForms.dll>

ISearchAccessForm2::EnabledRights

Disable/Enable changing specific rights.

C++ Syntax

```
HRESULT get_EnabledRights(  long* pVal);
HRESULT put_EnabledRights(  long Val);
```

Visual Basic Syntax

```
[ pVal = ] SearchAccessForm2.EnabledRights
SearchAccessForm2.get_EnabledRights [ = Val ]
```

Parameters

pVal	[out, retval] Returns a rights mask that can be modified by a user. For possible values see QuickSearchRights Enumeration.
Val	[in] Enable or disable user's ability to change specific rights. For possible values see QuickSearchRights Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <FsPForms.dll>

See Also

"QuickSearchRights Enumeration" on page 699

ISearchAccessForm2::HiddenTrustees

Hidden trustee's collection.

C++ Syntax

```
HRESULT get_HiddenTrustees(  ITrusteeCollection* ppVal);
```

Visual Basic Syntax

```
[ Set ppVal = ] SearchAccessForm2.HiddenTrustees
```

Parameters

ppVal	[out, retval] Returns collection of ISearchTrustee objects that should not be displayed. This collection may be modified to hide specific users or group.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <FsPForms.dll>

See Also

"ITrusteeCollection" on page 609

"ISearchTrustee" on page 595

ISearchAccessForm2::RestrictedTrustees

Trustee's collection with restricted access for changing.

C++ Syntax

```
HRESULT get_RestrictedTrustees(  ITrusteeCollection* ppVal);
```

Visual Basic Syntax

```
[ Set ppVal = ] SearchAccessForm2.RestrictedTrustees
```

Parameters

ppVal	[out, retval] Returns a collection of ISearchTrustee objects that should not be modified by user. This collection may be modified to limit access for specific users or group.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <FsPForms.dll>

See Also

"ITrusteeCollection" on page 609

"ISearchTrustee" on page 595

ISearchAccessForm2::RightsTemplates

Rights template collection.

C++ Syntax

```
HRESULT get_RightsTemplates(  ITrusteeCollection* ppVal);
```

Visual Basic Syntax

```
[ Set ppVal = ] SearchAccessForm2.RightsTemplates
```

Parameters

ppVal	[out, retval] Returns a collection of ISearchTrustee objects that contains templates to set rights. By default the collection contains the following items: “View Search”, “Read Only”, “Normal Access”, “Full Access”.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <FsPForms.dll>

See Also

“ITrusteeCollection” on page 609

“ISearchTrustee” on page 595

ISearchAccessForm2::VisibleRights

Show/hide specific right check boxes.

C++ Syntax

```
HRESULT get_VisibleRights(  long* pVal);

HRESULT put_VisibleRights(  long Val);
```

Visual Basic Syntax

```
[ pVal = ] SearchAccessForm2.get_VisibleRights

SearchAccessForm2.get_VisibleRights [ = Val ]
```

Parameters

pVal	[out, retval] Returns a rights mask that visible to a user. For possible values see QuickSearchRights Enumeration.
Val	[in] Hide or display specific rights. For possible values see QuickSearchRights Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import <FsPForms.dll>

See Also

"QuickSearchRights Enumeration" on page 699

ISearchCriteriaForm

A custom search criteria form that is used to edit criteria of a quick search should implement the interface. TheShowEditCriteriaForm method on a quick search uses the interface to display such a form.

Methods

ISearchCriteriaForm Methods	Description
ISearchCriteriaForm::ShowSearchCriteriaForm	Displays the search criteria form

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"ISearch" on page 567

ISearchCriteriaForm::ShowSearchCriteriaForm

The method displays the search criteria form.

C++ Syntax

```
HRESULT ShowSearchCriteriaForm(  ISearch* pSearch,  VARIANT_BOOL
bAllowEdits,  short* pResult);
```

Visual Basic Syntax

```
[ pResult  = ] SearchCriteriaForm.ShowSearchCriteriaForm (pSearch,
bAllowEdits)
```

Parameters

pSearch	[in] A Search object whose criteria form is to pop up.
bAllowEdits	[in] A Boolean value that specifies whether or not the search criteria can be changed. True: Search criteria can be changed. False: Search criteria cannot be changed.
pResult	[out, retval] Returns an integer that specifies user's action on the form. For possible values see FormsResults Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"ISearchCriteriaForm" on page 942

ISearchFormChooser

A custom form that prompts the user to select a search form should implement the interface. TheSelectSearchForm method on a library uses the interface to display such a form.

Methods

ISearchFormChooser Methods	Description
ISearchFormChooser::SelectSearchForm	Displays the custom form that prompts the user to select a search form and returns the selected form's name

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"ILibrary::SelectSearchForm" on page 394

ISearchFormChooser::SelectSearchForm

The method displays the custom form that prompts the user to select a search form and returns the selected form's name.

C++ Syntax

```
HRESULT SelectSearchForm(  short Flags,  ILibrary* pLibrary,
BSTR* pFormName);
```

Visual Basic Syntax

```
[ pFormName  = ] SearchFormChooser.SelectSearchForm (Flags,
pLibrary)
```

Parameters

Flags	[in] An integer value that specifies how the method should be executed. The parameter can be a combination of the SSFFlags values.
pLibrary	[in] A Library object whose search forms is to be presented.
pFormName	[out, retval] The method returns a string that contains the name of the form the user selected. If the user selected no search form by pressing a Cancel button, an empty string is returned.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_FAIL	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"ISearchFormChooser" on page 944

ISearchOptionsForm

A custom form that presents properties of a quick search should implement the interface. TheShowSearchOptionsForm method on a quick search uses the interface to display such a form.

Methods

ISearchOptionsForm Methods	Description
ISearchOptionsForm::ShowSearchOptionsForm	Displays the search form

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"ISearch" on page 567

ISearchOptionsForm::ShowSearchOptionsForm

The method displays the search form.

C++ Syntax

```
HRESULT ShowSearchOptionsForm(  ISearch* pSearch,  VARIANT_BOOL
bAllowEdits,  short* pResult,);
```

Visual Basic Syntax

```
[ pResult  = ] SearchOptionsForm.ShowSearchOptionsForm (pSearch,
bAllowEdits)
```

Parameters

pSearch	[in] A Search object whose form is to pop up.
bAllowEdits	[in] A Boolean value that specifies whether or not search properties can be changed. True: Search properties can be changed. False: Search properties cannot be changed.
pResult	[out, retval] Returns an integer that specifies user's action on the form. For possible values see FormsResults Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"ISearchOptionsForm" on page 945

ISelectVersionsForm

The form that appears as the multiple version selection dialog when the user needs to choose more than one version for opening, viewing, etc. should implement the interface. TheShowSelectVersionsForm method onDocProfiles uses the interface to display such a form.

Methods

ISelectVersionsForm Methods	Description
ISelectVersionsForm::ShowSelectVersionsForm	Displays the form for selecting document versions and/or attachments

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IDocProfiles::ShowSelectVersionsForm" on page 301

ISelectVersionsForm::ShowSelectVersionsForm

The method displays the form for selecting document versions and/or attachments.

C++ Syntax

```
HRESULT ShowSelectVersionsForm(  short Flags,  IDocProfiles* pDP,
BSTR caption,  IVersions** ppVersions);
```

Visual Basic Syntax

```
[ Set ppVersions  = ] SelectVersionsForm.ShowSelectVersionsForm
(Flags, pDP, caption)
```

Parameters

Flags	[in] An integer value that specifies the form appearance. The parameter can be one of the SelectVersionsFormFlags values.
pDP	[in] A DocProfiles object that specifies the documents whose versions the user will select.
caption	[in] A string that specifies the form caption.
ppVersions	[out, retval] Returns a Versions object that specifies the collection of versions selected by the user. If the user selected no versions, the return value is NULL.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"ISelectVersionsForm" on page 947

ISelectVersionTypeForm

A custom form that displays version type for folder components.

Methods

ISelectVersionTypeForm Methods	Description
ISelectVersionTypeForm::ShowSelectVersionTypeForm	The method displays form that allow to select version type of folder components.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <FsPForms.dll>

ISelectVersionTypeForm::ShowSelectVersionTypeForm

The method displays form that allows to select version type of folder components.

C++ Syntax

```
HRESULT ShowSelectVersionTypeForm(  IDOComponents* pDOComponents,
IProject* pParentProject,  short* psResult,);
```

Visual Basic Syntax

```
[ psResult  = ] SelectVersionTypeForm.ShowSelectVersionTypeForm (
pDOComponents, pParentProject )
```

Parameters

pDOComponents	[in] DOComponents object that contain folder components.
pParentProject	[in] A Project object that point to parent folder.
psResult	[out, retval] Returns an integer that specifies user's action on the form. For possible values see FormsResults Enumeration.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or later.

Header: #import <FsPForms.dll>

See Also

"ISelectVersionTypeForm" on page 949

ITemplateChooserForm

The form that appears when the user is prompted to select a template should implement this interface. TheShowTemplateChooserForm method on theApplication object uses this interface to display the form for selecting a template.

Methods

ITemplateChooserForm Methods	Description
ITemplateChooserForm::SelectTemplate	Displays the form for selecting a template and returns the template profile selected by the user

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IApplication::ShowTemplateChooserForm" on page 255

ITemplateChooserForm::SelectTemplate

The method displays the form for selecting a template and returns the template profile selected by the user.

C++ Syntax

```
HRESULT SelectTemplate(  long flags,  IProfile** pVal);
```

Visual Basic Syntax

```
[ Set pVal  = ] TemplateChooserForm.SelectTemplate (flags)
```

Parameters

flags	[in] An integer reserved for future use. Callers should specify 0 for this parameter; those called should ignore it.
pVal	[out, retval] Returns a Profile object that specifies the template selected by the user. If the user selected no template by pressing a Cancel button, the return value is NULL.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"ITemplateChooserForm" on page 951

IVersionChooserForm

A custom form that prompts the user to select a version or attachment on a document should implement the interface. TheShowVersionChooserForm method on a document profile uses the interface to display such a form.

Methods

IVersionChooserForm Methods	Description
IVersionChooserForm::ShowVersionChooserForm	Displays a form for selecting a document version/attachment

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IProfile::ShowVersionChooserForm" on page 463

IVersionChooserForm::ShowVersionChooserForm

The method displays the custom form that prompts the user to select a version or attachment on a specified document.

C++ Syntax

```
HRESULT ShowVersionChooserForm(  short Flags,  BSTR caption,
IProfile* pProfile,  IVersion** pSelected);
```

Visual Basic Syntax

```
[ Set pSelected  = ] VersionChooserForm.ShowVersionChooserForm
(Flags, caption, pProfile)
```

Parameters

Flags	[in] An integer value that specifies the form appearance. The parameter can be a combination of the VersionChooserFormFlags values.
caption	[in] A string that specifies the form caption.
pProfile	[in] A Profile object that specifies the document whose version the user should select.
pSelected	[out, retval] The method returns a Version object that specifies the version selected by the user. If the user selected no version, the return value is NULL.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.
E_POINTER	The operation failed.
E_UNEXPECTED	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <FsPForms.dll>

See Also

"IVersionChooserForm" on page 952

Custom Forms Enumerations

- "ExportInFlags Enumeration" on page 954
- "ExportOutFlags Enumeration" on page 955

ExportInFlags Enumeration

Export form input flags.

Elements

ExportInFlags Elements	Description
EIF_EnableAllVersions	Enable multi version selection.
EIF_EXPORT	Use As Export form.
EIF_DISABLE_COPY	Disable copy option.
EIF_DISABLE_COPYANDCHECKOUT	Disable copy and checkout option.
EIF_DISABLE_REFERENCE	Disable reference option.
EIF_DEFAULT_COPY	Send Copy option should be defaulted.
EIF_DEFAULT_COPYANDCHECKOUT	Send Copy and Check-out option should be defaulted.
EIF_DEFAULT_REFERENCE	Send Reference option should be defaulted.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "FsPForms.dll"

ExportOutFlags Enumeration

Export form output flags.

Elements

ExportOutFlags Elements	Description
EOF_COPY	Copy option selected.
EOF_COPYANDCHECKOUT	Copy and checkout option selected.
EOF_REFERENCE	Reference option selected.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: import "FsPForms.dll"

Chapter

7

Application Integration API

In This Chapter

- "About Application Integration API" on page 958
- "Active (ODMA) Integration" on page 959
- "Application Integration Interfaces" on page 960

About Application Integration API

Application Integration API allows to integrate Document Management Objects module to third-party application integration. This can be made by replacing Application Integration Module that DM Extensions is shipped. Also additional integration scheme can be implemented and built-in DM Extensions Application Integration.

Creation of new Integration Scheme

1. Create a COM server which exposes desired functionality and implements a concreatable object that will support the IIntegrationScheme interface.

2. Register your Integration Scheme as described below.

Registration of new Integration Scheme

Before DM 5.0 version

1 Under HKLM\Software\Hummingbird \PowerDOCS\AppIntegration create a key which will hold all the information about your integration scheme.

2 Under that new key you can specify the following information:

Value	Description
@(Default)	(required) ProgID or CLSID of your IntegrationScheme object
Description	(optional) Text to describe your integration scheme
InsertAt	(optional) Letter from A to Z which indicates the position of the integration scheme, A – first, B – second and so on. Application Integration module calls integration schemes according their position.
Disable	(optional) 1 for temporary disabling this integration scheme.

DM 5.0 or later version

IBCRegistrarExt::RegisterExtObject method should be used to register new integration scheme. See Remarks section in this method for possible values of additional parameters.

Example

This example registers "SimpleIntegrationScheme".

```
Dim pCtx As New BCCoreCtx

Dim pRegistrar As New BCRegistrar

pCtx.RegisterVariantParam "InsertAt", "M"

pRegistrar.RegisterExtObject rcRegister, "AppIntegrationScheme",
"SimpleIntegrationScheme",
"SimpleIntegrationScheme.SimpleIntegrationSchemeClass", pCtx
```

Active (ODMA) Integration

Open Document Management API (ODMA) is an API that allows applications and document management systems to inter-operate through a single high level API. ODMA is the most preferable way to communicate with document management systems.

The Active integration scheme provides the integration functionality for ODMA-compliant applications. It supports the ODMA 1.0, ODMA 1.5, and ODMA 2.0 standards.

The ODMA connection manager, ODMA32.DLL, serves as a common entry point for applications that wish to call ODMA functions. Its implementation of ODMRegisterApp figures out which DMS is registered for default use by the calling application. It will then load the appropriate DMS's DLL and pass through the function call. All other ODMA function calls that do not include a document ID will be passed to this default DMS.

An ODMA call includes a document ID, the ODMA connection manager will determine which DMS created the ID. If that DMS has not been initialized yet by the calling application, the connection manager will call the provider's ODMGetODMInterface function and will then pass through the current ODMA function call.

The following scheme shows the interaction of the Document Management plug-in, AIM, ODMA integration scheme, and an ODMA-compliant application.

See Also

"ODMGetODMInterface" on page 1001

Application Integration Interfaces

- "IAppIntegration" on page 960
- "ICloseNotify" on page 976
- "IIntegrationScheme" on page 978

IAppIntegration

The IAppIntegration object is implemented in Application Integration Module and interacts with the Document Management Objects module and integration schemes.

Methods

IAppIntegration Methods	Description
IAppIntegration::Open	Opens a document's version in the specific mode using an application
IAppIntegration::OpenDocument	Opens a document's version in the specific mode using the specific launch method
IAppIntegration::Preview	Previews a document's version
IAppIntegration::PrintDoc	Prints a specified version
IAppIntegration::LaunchApplication	Launches a specified application
IAppIntegration::MonitorClose	Monitors when a document being edited is closed
IAppIntegration::CancelMonitor	Cancels monitoring when a document being edited is closed
IAppIntegration::HasOpenDocuments	Indicates whether or not there are opened documents
IAppIntegration::CloseOpenDocuments	Closes all documents opened via the AIM

IAppIntegration Methods	Description
IAppIntegration::BuildCmdLineString	Builds a command line string by a specific version and launch method
IAppIntegration::RemoveTempFiles	Removes temporarily files.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <AIM.exe>

See Also

"ICloseNotify" on page 976

"IIntegrationScheme" on page 978

IAppIntegration::BuildCmdLineString

This method builds a command line string by the specific version and launch method.

C++ Syntax

```
HRESULT BuildCmdLineString(  IVersion* pVersion,  ILaunchMethod*
pLaunchMethod,  IUnknown* pCtx);
```

Visual Basic Syntax

```
AppIntegration.BuildCmdLineString (pVersion, pLaunchMethod, pCtx)
```

Parameters

pVersion	[in] Pointer to the IVersion interface specifying the version being edited.
pLaunchMethod	[in] Pointer to the ILaunchMethod interface specifying the launch method that should be used to open the version.
pCtx	[in] Pointer to an object implementing the IBCCoreCtx interface that specifies additional parameters and, upon successful return, contains the requested command line as follows:

Key	Value
%CommandLine	Address of a Basic wide character string (two bytes per character) to contain the requested command line.

Return Values

This method supports the following return values:

S_OK	The operation succeeded.
E_OUTOFMEMORY	There is not enough memory to complete the operation.
E_FAIL	An unspecified error occurred.
E_POINTER	pVersion, pLaunchMethod, or pCtx point to NULL.

Remarks

This method builds the command line using the settings of the specified launch method.

The caller may redefine any of the standard command line system variables by using the IBCCoreCtx interface passed as the pCtx parameter. For instance, the ODMA ISM may use it to substitute the document ID for %FULLPATH. The implementation then finds the

respective value in the IBCCoreCtx map. If that value is found, it will be used. Otherwise, the AIM assigns the values to such system variables by itself.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <AIM.exe>

See Also

"IAppIntegration" on page 960

"IVersion" on page 625

"ILaunchMethod" on page 331

"IBCCoreCtx" on page 46

IAppIntegration::CancelMonitor

This method cancels monitoring when a document being edited is closed.

C++ Syntax

```
HRESULT CancelMonitor(  IVersion* pVersion);
```

Visual Basic Syntax

```
AppIntegration.CancelMonitor (pVersion)
```

Parameters

pVersion	[in] Pointer to the IVersion interface specifying the version being edited.

Return Values

This method supports the following return values:

S_OK	The operation succeeded.
E_OUTOFMEMORY	There is not enough memory to complete the operation.
E_FAIL	An unspecified error occurred.

Remarks

This method may be used to cancel monitoring an open document. If the document is being monitored, the implementation calls the ICloseNotify::DocumentClosed method.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <AIM.exe>

See Also

"IAppIntegration" on page 960

"IVersion" on page 625

"ICloseNotify::DocumentClosed" on page 976

IAppIntegration::CloseOpenDocuments

This method closes all documents opened via the AIM.

C++ Syntax

```
HRESULT CloseOpenDocuments();
```

Visual Basic Syntax

```
AppIntegration.CloseOpenDocuments
```

Return Values

This method supports the following return values:

S_OK	The operation succeeded.
E_FAIL	An unspecified error occurred.

Remarks

This method first asks all the ISMs to close their open documents. Then the implementation cancels monitoring all the documents opened by the AIM itself and deletes temporary files.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <AIM.exe>

See Also

"IAppIntegration" on page 960

IAppIntegration::HasOpenDocuments

This method indicates whether or not there are opened documents.

C++ Syntax

```
HRESULT HasOpenDocuments(  VARIANT_BOOL* pbRes);
```

Visual Basic Syntax

```
[ pbRes  = ] AppIntegration.HasOpenDocuments
```

Parameters

pbRes	[out, retval] Address of the VARIANT_BOOL variable to receive a request result as follows:

Value	Description
VARIANT_TRUE	The AIM has opened documents.
VARIANT_FALSE	The AIM has no opened documents.

Return Values

This method supports the following return values:

S_OK	The operation succeeded.
E_POINTER	pbRes points to NULL.

Remarks

This method first asks all the ISMs if they have opened documents. If the ISMs return false, the implementation checks whether or not the AIM opened documents by itself.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <AIM.exe>

See Also

"IAppIntegration" on page 960

IAppIntegration::LaunchApplication

The method launches a specified application.

C++ Syntax

```
HRESULT LaunchApplication(  VARIANT vApp);
```

Visual Basic Syntax

```
AppIntegration.LaunchApplication (vApp)
```

Parameters

vApp	[in] Specifies an application to launch. If of type VT_BSTR, it specifies the address of a Basic wide character string to contain an application identifier (e.g., "MS WORD"). If of type VT_DISPATCH or VT_UNKNOWN, it points to an IAppInfo interface.

Return Values

This method supports the following return values:

S_OK	The operation succeeded.
S_FALSE	The user canceled the operation.
E_OUTOFMEMORY	There is not enough memory to complete the operation.
E_FAIL	An unspecified error occurred.
E_INVALIDARG	vApp specifies an invalid application.
DISP_E_TYPEMISMATCH	vApp is not of the valid type.

Remarks

The method launches a specified application as follows:

- Determines which launch method should be used to launch the application. If there is only one launch method associated with the application specified by vApp, then it is used. Otherwise, a form is presented to the user to select a launch method by calling up IAppInfo::ShowLaunchMethodsChooserForm.

- Calls up IIntegrationScheme::LaunchApplication for each available integration scheme to find out which of them will proceed with launching the application. If there is no such integration scheme, then the method launches it by itself using launch method parameters.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <AIM.exe>

See Also

"IAppIntegration" on page 960

"IAppInfo::ShowLaunchMethodChooserForm" on page 232

"IIntegrationScheme::LaunchApplication" on page 981

IAppIntegration::MonitorClose

This method monitors when a document being edited is closed.

C++ Syntax

```
HRESULT MonitorClose(  ILaunchMethod* pLaunchMethod,  IVersion*
pVersion,  ICloseNotify* pNotify);
```

Visual Basic Syntax

```
AppIntegration.MonitorClose (pLaunchMethod, pVersion, pNotify)
```

Parameters

pLaunchMethod	[in] Pointer to the ILaunchMethod interface specifying hre launch method that was used to open the version.
pVersion	[in] Pointer to the IVersion interface specifying the version being edited.
pNotify	[in] Pointer to the ICloseNotify interface that will receive notification about closing the document. If NULL, an internal implementation is used.

Return Values

This method supports the following return values:

S_OK	The operation succeeded.
E_OUTOFMEMORY	There is not enough memory to complete the operation.
E_FAIL	An unspecified error occurred.
E_POINTER	pLaunchMethod or pVersion points to NULL.

Remarks

The implementation first tries to establish DDE conversation with the application associated with the specified launch method to monitor when the conversation is terminated. This will signal that the document is closed. If the launch method does not support DDE communication, the method will start a background thread to monitor when the application exits.

When the document is closed, the AIM calls the ICloseNotify::DocumentClosed method to notify the caller about the event.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <AIM.exe>

See Also

"IAppIntegration" on page 960

"ILaunchMethod" on page 331

"IVersion" on page 625

"ICloseNotify" on page 976

IAppIntegration::Open

The method opens a document's version in a specific mode using an application.

C++ Syntax

```
HRESULT Open(  long lMode,  IVersion* pVersion,  VARIANT vApp);
```

Visual Basic Syntax

```
AppIntegration.Open (lMode, pVersion, vApp)
```

Parameters

lMode	[in] Specifies the mode in which the version is opened. The parameter can be a combination of the AccessMode enumeration values.
pVersion	[in] Pointer to the IVersion interface specifying a version that should be opened.
vApp	[in] Specifies an application used to open the version. If of type VT_BSTR, it specifies the address of a Basic wide character string to contain an application identifier (e.g., "MS WORD"). If of type VT_DISPATCH or VT_UNKNOWN, it points to an IAppInfo interface.

Return Values

This method supports the following return values:

S_OK	The operation succeeded.
S_FALSE	The user canceled the operation.

E_OUTOFMEMORY	There is not enough memory to complete the operation.
E_FAIL	An unspecified error occurred.
E_POINTER	pVersion points to NULL.
E_INVALIDARG	vApp specifies an invalid application.
DISP_E_TYPEMISMATCH	vApp is not of valid type.

Remarks

The method opens a version in the mode specified by the dwMode parameter using an application. To do this, the method performs the following steps:

- Determines which launch method should be used to open the document. If there is only one launch method associated with the application specified by vApp, then it is used. Otherwise, a form is presented to the user to select a launch method by calling up IAppInfo::ShowLaunchMethodsChooserForm.
- Calls up IIntegrationScheme::Open for each available integration scheme to find out which of them will proceed with opening the document. If there is no such integration scheme, the method calls OpenDocument to open the document by itself.
- If the document was opened by the AIM itself, the AIM has to receive information on document closing within the application. For that, it calls its own MonitorClose method and passes to it the pointer to its own service object that implements ICloseNotify.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <AIM.exe>

See Also

"IAppIntegration" on page 960

"IVersion" on page 625

"IAppInfo" on page 220

IAppIntegration::OpenDocument

The method opens a document's version in a specific mode using the specific launch method.

C++ Syntax

```
HRESULT OpenDocument(  long lMode,  ILaunchMethod* pLaunch,
IVersion* pVersion,  IUnknown* pCtx);
```

Visual Basic Syntax

```
AppIntegration.OpenDocument (lMode, pLaunch, pVersion, pCtx)
```

Parameters

lMode	[in] Specifies the mode in which the version is opened. The parameter can be a combination of the AccessMode enumeration values.
pLaunch	[in] Pointer to the ILaunchMethod interface specifying the launch method that will be used to open the version.
pVersion	[in] Pointer to the IVersion interface specifying a version that should be opened.
pCtx	[in] Pointer to an object implementing the IBCCoreCtx interface that specifies additional parameters. The parameter can be NULL.

Return Values

This method supports the following return values:

S_OK	The operation succeeded.
E_OUTOFMEMORY	There is not enough memory to complete the operation.
E_FAIL	An unspecified error occurred.
E_POINTER	pLaunchMethod or pVersion points to NULL.

Remarks

The method opens a version in the mode specified by the dwMode parameter using the launch method specified by pLaunchMethod.

ISMs can use this method to actually open a document in an application using a particular launch method. The AIM should launch the application using the command line stored in the launch method and load the document into it. This can involve DDE conversations or whatever is required by the passed launch method.

The AIM uses this operation internally to open documents that were rejected by all the registered ISMs.

The caller may redefine any of the standard command line system variables by using the IBCCoreCtx interface passed as the pCtx parameter. For instance, the ODMA ISM may use it to substitute the document ID for %FULLPATH. The implementation then finds the respective value in the IBCCoreCtx map. If that value is found, it will be used. Otherwise, the AIM assigns the values to such system variables by itself.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <AIM.exe>

See Also

"IAppIntegration" on page 960

"IVersion" on page 625

"ILaunchMethod" on page 331

"IBCCoreCtx" on page 46

IAppIntegration::Preview

The method previews a document's version.

C++ Syntax

```
HRESULT Preview(  IVersion* pVersion);
```

Visual Basic Syntax

```
AppIntegration.Preview (pVersion)
```

Parameters

pVersion	[in] Pointer to the IVersion interface specifying a version that should be previewed.

Return Values

This method supports the following return values:

S_OK	The operation succeeded.
S_FALSE	The user canceled the operation.
E_OUTOFMEMORY	There is not enough memory to complete the operation.
E_FAIL	An unspecified error occurred.
E_POINTER	pVersion points to NULL.

Remarks

The method previews a version. To do this, the method performs the following steps:

- Finds out which application is a "view" one for this document.
- Determines which launch method should be used to preview the document. If there is only one launch method associated with the view application, then it is used. Otherwise, a form is presented to the user to select a launch method by calling up IAppInfo::ShowLaunchMethodsChooserForm.
- Calls up IIntegrationScheme::Preview for each available integration scheme to find out which of them will proceed with viewing the document. If there is no such integration scheme, the method calls OpenDocument to open the document in read-only mode.
- If an error occurred while opening the document via OpenDocument, the document is previewed using the standard DM Extensions viewer.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <AIM.exe>

See Also

"IAppIntegration" on page 960

IAppIntegration::PrintDoc

This method prints a specified version.

C++ Syntax

```
HRESULT PrintDoc(  IVersion* pVersion);
```

Visual Basic Syntax

```
AppIntegration.PrintDoc (pVersion)
```

Parameters

pVersion	[in] Pointer to the IVersion interface specifying a version that should be printed.

Return Values

This method supports the following return values:

S_OK	The operation succeeded.
S_FALSE	The user canceled the operation.
E_OUTOFMEMORY	There is not enough memory to complete the operation.
E_FAIL	An unspecified error occurred.
E_POINTER	pVersion points to NULL.

Remarks

The method prints a version. To do this, the method performs the following steps:

- Reads which application is a "print" one for this document.
- Determines which launch method should be used to print the document. If there is only one launch method associated with the print application, then it is used. Otherwise, a form is presented to the user to select a launch method by calling up IAppInfo::ShowLaunchMethodsChooserForm.

- Calls up IIntegrationScheme::Print for each available integration scheme to find out which of them will proceed with printing the document.
- If there is no such ISM, the method reads the application's DDE print settings and sends the DDE command to print the document.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <AIM.exe>

See Also

"IAppIntegration" on page 960

"IVersion" on page 625

IAppIntegration::RemoveTempFiles

This method removes temporarily files.

C++ Syntax

```
HRESULT RemoveTempFiles(  IUnknown* pCoreCtx);
```

Visual Basic Syntax

```
AppIntegration.RemoveTempFiles (pCoreCtx)
```

Parameters

pCoreCtx	[in] Pointer to an IBCCoreCtx interface that contains information about files to be deleted.

Key	Value
ATFILE*N* where *N* – 1…Number of files to be deleted	Full path to file that should be deleted.

Return Values

This method supports the following return values:

S_OK	The operation succeeded.

E_UNEXPECTED	An unspecified error occurred.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <AIM.exe>

See Also

"IAppIntegration" on page 960

"IBCCoreCtx" on page 46

ICloseNotify

This interface is used to receive notification about closing a document. This interface is passed to the IAppIntegration::MonitorClose method.

When to Implement

The interface inherits its definition from IUnknown, so its implementation must also include three standard methods of IUnknown: AddRef, Release, and QueryInterface.

Methods

ICloseNotify Methods	Description
ICloseNotify::DocumentClosed	Receives notification that a specified version has been closed

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <AIM.exe>

See Also

"IAppIntegration" on page 960

"IIntegrationScheme" on page 978

ICloseNotify::DocumentClosed

This method receives notification that a specified version has been closed.

C++ Syntax

```
HRESULT DocumentClosed(  IVersion* pVersion);
```

Visual Basic Syntax

```
CloseNotify.DocumentClosed (pVersion)
```

Parameters

pVersion	[in] Pointer to the IVersion interface specifying the version being edited.

Return Values

This method supports the following return values:

S_OK	The operation succeeded.
E_OUTOFMEMORY	There is not enough memory to complete the operation.
E_FAIL	An unspecified error occurred.
E_POINTER	pVersion points to NULL.

Remarks

If the specified document is being monitored when it is closed, the AIM calls this method to notify the object about the event.

The implementation should perform all needed steps upon this event. For instance, it may unlock the document's version and upload it to the server to save the changes the user made while editing the document.

The AIM implements this method when opening a document internally by the IAppIntegration::OpenDocument method.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <AIM.exe>

See Also

"ICloseNotify" on page 976

"IVersion" on page 625

IIntegrationScheme

This interface must be implemented by each ISM. The AIM uses this interface to interact with ISMs.

When to Implement

The interface inherits its definition from IUnknown, so its implementation must also include three standard methods of IUnknown: AddRef, Release, and QueryInterface.

Methods

IIntegrationScheme Methods	Description
IIntegrationScheme::Init	Initializes the ISM
IIntegrationScheme::Open	Opens a document's version in the specific mode using the specific launch method
IIntegrationScheme::Preview	Previews a document's version
IIntegrationScheme::Print	Prints a specified version
IIntegrationScheme::LaunchApplication	Launches an application by the specific launch method
IIntegrationScheme::HasOpenDocuments	Indicates whether or not the ISM has open documents
IIntegrationScheme::CloseOpenDocuments	Closes all documents opened via the ISM

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <AIM.exe>

See Also

"IAppIntegration" on page 960

"ICloseNotify" on page 976

IIntegrationScheme::CloseOpenDocuments

This method closes all documents opened via the ISM.

C++ Syntax

```
HRESULT CloseOpenDocuments();
```

Visual Basic Syntax

```
IntegrationScheme.CloseOpenDocuments
```

Return Values

This method supports the following return values:

S_OK	The operation succeeded.
E_FAIL	An unspecified error occurred.

Remarks

The AIM calls this method while processing IAppIntegration::CloseOpenDocuments to allow ISMs to close documents opened by the ISMs themselves.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <AIM.exe>

See Also

"IIntegrationScheme" on page 978

IIntegrationScheme::HasOpenDocuments

This method indicates whether or not the ISM has open documents.

C++ Syntax

```
HRESULT HasOpenDocuments(  VARIANT_BOOL* pbRes);
```

Visual Basic Syntax

```
[ pbRes  = ] IntegrationScheme.HasOpenDocuments
```

Parameters

pbRes	[out, retval] Address of the VARIANT_BOOL variable to receive a request result as follows:

Value	Description
VARIANT_TRUE	ISM has open documents.
VARIANT_FALSE	ISM has no open documents.

Return Values

This method supports the following return values:

S_OK	The operation succeeded.
E_POINTER	pbRes points to NULL.

Remarks

The AIM calls this method while processing IAppIntegration::HasOpenDocuments to check whether ISMs have opened documents by themselves.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <AIM.exe>

See Also

"IIntegrationScheme" on page 978

IIntegrationScheme::Init

The method initializes the ISM.

C++ Syntax

```
HRESULT Init(  IAppIntegration* pAppIntegration);
```

Visual Basic Syntax

```
IntegrationScheme.Init (pAppIntegration)
```

Parameters

pAppIntegration	[in] Pointer to the IAppIntegration interface specifying the AIM.

Return Values

This method supports the following return values:

S_OK	The operation succeeded.
E_OUTOFMEMORY	There is not enough memory to complete the operation.
E_FAIL	An unspecified error occurred.
E_POINTER	pAppIntegration points to NULL.

Remarks

A call on this method should precede any other calls on IIntegrationScheme. The AIM calls this method when loading all the registered ISMs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <AIM.exe>

See Also

"IIntegrationScheme" on page 978

IIntegrationScheme::LaunchApplication

The method launches an application by the specific launch method.

C++ Syntax

```
HRESULT LaunchApplication(  IAppIntegration* pAppIntegration,
ILaunchMethod* pLaunchMethod);
```

Visual Basic Syntax

```
IntegrationScheme.LaunchApplication (pAppIntegration,
pLaunchMethod)
```

Parameters

pAppIntegration	[in] Pointer to the IAppIntegration interface specifying the AIM.
pLaunchMethod	[in] Pointer to the ILaunchMethod interface specifying the launch method that will be used to launch the application.

Return Values

This method supports the following return values:

S_OK	The operation succeeded.
S_FALSE	The ISM rejects processing the call.
E_OUTOFMEMORY	There is not enough memory to complete the operation.
E_FAIL	An unspecified error occurred.
E_POINTER	pAppIntegration or pLaunchMethod points to NULL.
E_NOTIMPL	The method is not implemented.

Remarks

The AIM calls this method while processing IAppIntegration::LaunchApplication to allow ISMs to launch an application using its launch method by themselves. If an ISM returns S_OK, the AIM does not proceed with its internal method implementation, assuming that the ISM has executed the command.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <AIM.exe>

See Also

"IIntegrationScheme" on page 978

"IAppIntegration" on page 960

"ILaunchMethod" on page 331

IIntegrationScheme::Open

The method opens a document's version in the specific mode using the specific launch method.

C++ Syntax

```
HRESULT Open(  long dwMode,  IAppIntegration* pAppIntegration,
IVersion* pVersion,  ILaunchMethod* pLaunchMethod);
```

Visual Basic Syntax

```
IntegrationScheme.Open (dwMode, pAppIntegration, pVersion,
pLaunchMethod)
```

Parameters

dwMode	[in] Specifies the mode in which the version is opened. The parameter can be a combination of the AccessMode values.
pAppIntegration	[in] Pointer to the IAppIntegration interface specifying the AIM.
pVersion	[in] Pointer to the IVersion interface specifying a version that should be opened.
pLaunchMethod	[in] Pointer to the ILaunchMethod interface specifying a launch method that will be used to open the version.

Return Values

This method supports the following return values:

S_OK	The operation succeeded.
S_FALSE	The ISM rejects processing the call.
E_OUTOFMEMORY	There is not enough memory to complete the operation.
E_FAIL	An unspecified error occurred.
E_POINTER	pAppIntegration, pVersion, or pLaunchMethod points to NULL.
E_NOTIMPL	The method is not implemented.

Remarks

The method opens a version in the mode specified by the dwMode parameter using the launch method specified by pLaunchMethod.

The AIM calls this method while processing IAppIntegration::Open allowing ISMs to open a document by themselves before using IAppIntegration::OpenDocument. If an ISM returns S_OK, the AIM "forgets" about monitoring the document from now on, assuming that the ISM has executed the command.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <AIM.exe>

See Also

"IIntegrationScheme" on page 978

"IAppIntegration" on page 960

"IVersion" on page 625

"ILaunchMethod" on page 331

IIntegrationScheme::Preview

The method previews a document's version.

C++ Syntax

```
HRESULT Preview(  IAppIntegration* pAppIntegration,  IVersion*
pVersion,  ILaunchMethod* pLaunchMethod);
```

Visual Basic Syntax

```
IntegrationScheme.Preview (pAppIntegration, pVersion,
pLaunchMethod)
```

Parameters

pAppIntegration	[in] Pointer to the IAppIntegration interface specifying the AIM.
pVersion	[in] Pointer to the IVersion interface specifying a version that should be previewed.
pLaunchMethod	[in] Pointer to the ILaunchMethod interface specifying a launch method that will be used to view the version.

Return Values

This method supports the following return values:

S_OK	The operation succeeded.
S_FALSE	The ISM rejects processing the call.
E_OUTOFMEMORY	There is not enough memory to complete the operation.
E_FAIL	An unspecified error occurred.
E_POINTER	pAppIntegration, pVersion, or pLaunchMethod points to NULL.
E_NOTIMPL	The method is not implemented.

Remarks

The AIM calls this method while processing IAppIntegration::Preview allowing ISMs to preview a document by themselves. If an ISM returns S_OK, the AIM does not proceed with its internal preview implementation, assuming that the ISM has executed the command.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <AIM.exe>

See Also

"IIntegrationScheme" on page 978

"IAppIntegration" on page 960

"IVersion" on page 625

"ILaunchMethod" on page 331

IIntegrationScheme::Print

This method prints a specified version.

C++ Syntax

```
HRESULT Print(  IAppIntegration* pAppIntegration,  IVersion*
pVersion,  ILaunchMethod* pLaunchMethod);
```

Visual Basic Syntax

```
IntegrationScheme.Print (pAppIntegration, pVersion, pLaunchMethod)
```

Parameters

pAppIntegration	[in] Pointer to the IAppIntegration interface specifying the AIM.
pVersion	[in] Pointer to the IVersion interface specifying a version that should be printed.
pLaunchMethod	[in] Pointer to hte ILaunchMethod interface specifying a launch method that will be used to print the version.

Return Values

This method supports the following return values:

S_OK	The operation succeeded.
S_FALSE	The ISM rejects processing the call.
E_OUTOFMEMORY	There is not enough memory to complete the operation.
E_FAIL	An unspecified error occurred.
E_POINTER	pAppIntegration, pVersion, or pLaunchMethod points to NULL.
E_NOTIMPL	The method is not implemented.

Remarks

The AIM calls this method while processing IAppIntegration::PrintDoc to allow ISMs to print a document by themselves. If an ISM returns S_OK, the AIM does not proceed with its internal print implementation by sending the DDE command to print the document, assuming that the ISM has executed the command.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <AIM.exe>

See Also

"IIntegrationScheme" on page 978

"IAppIntegration" on page 960

"IVersion" on page 625

"ILaunchMethod" on page 331

Chapter

8

Interceptor

In This Chapter

- "About Interceptor" on page 990
- "Architecture" on page 991
- "Interceptor Hook Module" on page 991
- "Interceptor Core Module" on page 992
- "Document-oriented Events" on page 993
- "Processing Interceptor Core Events" on page 994
- "Wait/Release Functionality" on page 996

About Interceptor

The Interceptor is used for full integration of applications with DM Extensions via the settings of launch methods but not on the basis of the respective macro modules. The only requirement for the applications is that they have a dialog window to open/save files. Naturally, almost all the applications meet this requirement.

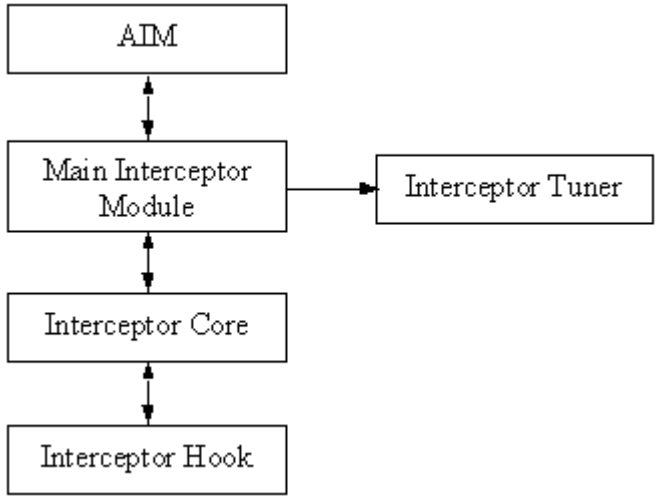
The Interceptor intercepts such operations as opening or closing a document in an application as follows:

- When the application Open dialog pops up, the Interceptor hides it and shows the Quick Retrieve form. It locks and opens the selected documents.
- When the application Save As dialog pops up, the Interceptor hides it and shows a Save As options form or a profile form to import the document to DM. Then it saves the document.
- When the document window (for MDI-applications) or the application window (for SDI- applications) closes, the Interceptor closes and unlocks the document.

The Interceptor is able to work with launch methods that support full integration.

Architecture

The Interceptor consists of four modules:



- Hook module monitors window messages and sends them to the core module for further processing.
- Core module processes and interprets the events from the hook module and generates the appropriate document-oriented events.
- Main Interceptor module processes the document-oriented events the core module generates and communicates with the AIM. The Interceptor also implements the IIntegrationScheme interface.
- Tuner module allows the user to configure the Interceptor.

See Also

"IIntegrationScheme" on page 978

Interceptor Hook Module

This module is implemented in InterceptorHooks.DLL. Its primary goal is monitoring window messages and sending the corresponding events to the core module.

This module installs a hook procedure that monitors window messages before the system sends them to the destination window procedure by calling the system SetWindowsHookEx function. It filters out unneeded messages and notifies the core module about other ones.

Interceptor Core Module

The Interceptor core module is implemented in InterceptorCore.DLL and performs the following tasks:

Initializes and terminates the hook module.

Accumulates and processes hook events.

Generates document-oriented events and notifies the appropriate sink object.

The core module implements application monitoring. It identifies all monitored applications by a unique cookie. A new cookie is assigned to an application when the Interceptor starts monitoring it.

The Interceptor works with documents by using window captions. It monitors a window, which caption matches to the document and assumes that the document is open while the corresponding window exists or has the matched caption. The technology of parsing window captions is based on the document identifier. The document ID identifies the document uniquely and has the following format:

Item1 delimiter Item2 delimiter …Itemn,

where Item1, …, Itemn are application-dependent variables and delimiter is the symbol with ASCII code 0x19 (DOCID_DELIMITER constant)

DM Extensions defines a template that the Interceptor uses for parsing document identifiers. This template is stored in the HKLM\Software\Hummingbird\PowerDOCS\AppIntegration\IntegrationSchemes\Interceptor\Settings\DocFilters registry value. The template has filter variables %1,%2,… that are replaced by Item1, …, Itemn when the Interceptor analyses a window caption. If the window caption contains the resultant string the Interceptor assumes that this window contains the document.

Before the Interceptor uses a document ID, it should be registered. After the registration, the module tries to find the window whose caption matches the registered document ID. When the core module finds such a window, it sends the OPENED event to the main Interceptor module.

Document-oriented Events

The core module can send the following predefined events to the main Interceptor module:

Event	Description
OPEN	Occurs when the monitored application's File Open dialog opens a document for editing. The Interceptor should monitor the document after opening.
OPENED	Occurs when the core module spots a window that corresponds to a registered document ID.
PRINT	Occurs when a monitored application begins printing a document (the corresponding printer job appears in the printer spooler).
PRINTED	Occurs when the printer finishes printing a document (the corresponding printer job is finished).
SAVEAS	Occurs when the monitored application's Save As dialog saves a document with a different name. The Interceptor should creates a new document and monitor it after saving.
SAVED	Occurs when one of the registered document's files is modified.

Event	Description
IMPORT	Occurs when a monitored application's Insert File (or similar) dialog retrieves the contents of a document but does not open it for editing. The Interceptor should copy the document to a temporary file and open it in the application.
EXPORT	Occurs when the monitored application's Export (or similar) dialog exports a document to another application and then continues editing the document. The Interceptor should create a new document, save its contents to a file, and close the document.
CLOSE	Occurs when a registered document was closed in a monitored application or the application exits.

Processing Interceptor Core Events

This section describes how the Interceptor processes different document-oriented events.

OPEN

- A Quick Retrieve form pops up to select a document. The selected document is downloaded to a local file.
- The Interceptor builds a document ID and starts monitoring the document.
- The Interceptor emulates opening the document's local file.

OPENED

- The document is marked as 'opened'.
- If the document is not monitored, the Interceptor starts monitoring it.

PRINT

- The document is marked as 'being printed'.

PRINTED

- The Interceptor clears its 'being printed' mark.
- If the document is marked as 'being deleted', the Interceptor stops monitoring it.
- The Interceptor adds a print activity to the DM Extensions activity log.

SAVEAS

- If the document being edited is a DM document, the Save As options form pops up to allow the user to replace the current document's version, save the document as a new version/subversion of the same document, or save it as a new document. Otherwise, the document is imported to DM.
- If the user created a new document, version/subversion, the Interceptor builds a new document ID and starts monitoring the new document's version. The old version is marked as 'being closed'.
- The Interceptor emulates saving the document to a local file.

SAVED

- The document is uploaded to the server.
- If the document is marked as 'exported', the Interceptor unlocks the document and stops monitoring it.

IMPORT

- A Quick Retrieve form pops up to select a document. The selected document is downloaded to a local file.
- A copy of the selected document is created and the Interceptor emulates opening this copy in the application.

EXPORT

- A new blank document is created and profiled to DM.
- The Interceptor builds a new document ID and starts monitoring the new document.

- The Interceptor emulates exporting the source document to the local file of the new document.
- The new document is marked as 'exported'.

CLOSE

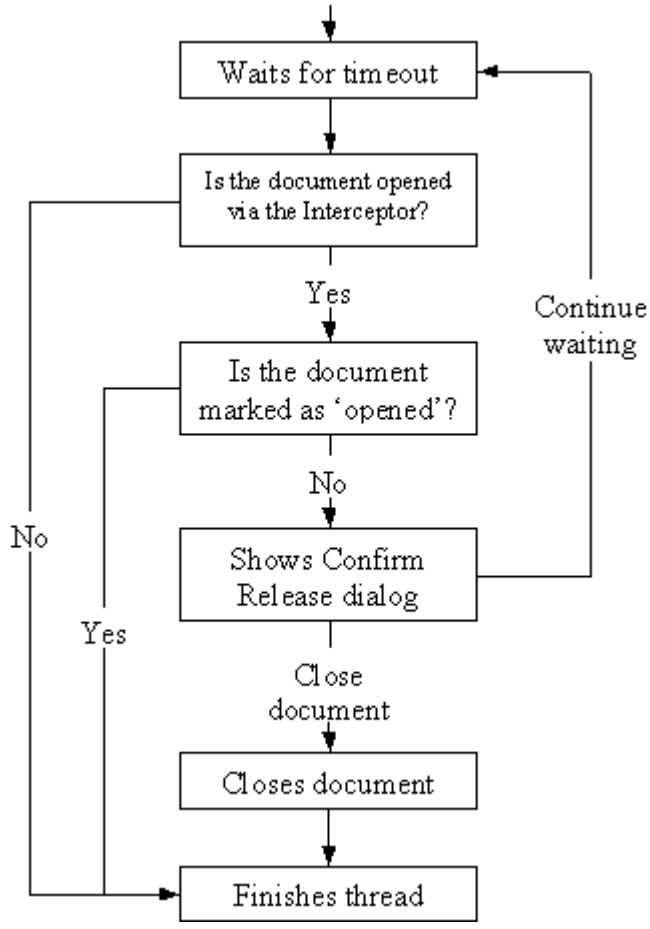
- The document is marked as 'being closed' and the Interceptor unlocks it.
- The Interceptor builds a new document ID and starts monitoring the new document.
- The Interceptor emulates exporting the source document to the local file of the new document.
- The new document is marked as 'exported'.
- If the document is marked as 'being printed', the Interceptor marks it as 'being deleted' to stop monitoring it after printing. Otherwise, the Interceptor stops monitoring the document.

Wait/Release Functionality

Sometimes a monitored application cannot open a document because of its incompatible format. The same situation can arise if the user canceled opening the document but the Interceptor was not able to intercept this action and locked the document. In that case, the document remains locked but the user cannot close and unlocks it.

To prevent this situation, while opening a document, the Interceptor starts a background thread that periodically checks whether the document was actually opened.

The following scheme shows how this thread works:



Chapter

9

Active Integration Implementation

In This Chapter

- "ODMA Document ID" on page 1000
- "ODMA DMS Interfaces" on page 1000
- "Document Attributes" on page 1025

ODMA Document ID

Many of the ODMA functions accept or return an ODMA document ID parameter. A document ID is a persistent, portable identifier for a document. It can be stored and used in a later session, and it can be passed across platforms via e-mail or other processes.

An ODMA document ID is a case insensitive, null-terminated string of printable characters. The format of a DM document ID is

::ODMA\PCDOCS\LibName\DocNumber\VersionLabel

The 'PCDOCS' part of an ODMA document ID identifies DM, the document management system that provides this ID. Other parameters uniquely specify a document version: LibName is the name of the library that contains the document; DocNumber is the number of the document; VersionLabel is the label of the document version.

A special ODMA document ID, ::ODMA\PCDOCS\NULL, is also supported. This document ID is used when creating a new document.

When executing a query, ODMA uses query IDs. The format of a DM query ID is as follows:

::ODMA\PCDOCS\QueryN

where N is the current number of the query. Such IDs are used with the IODMQuery interface.

See Also

"Active (ODMA) Integration" on page 959

ODMA 2.0 Specification

ODMA DMS Interfaces

- "ODMGetODMInterface" on page 1001
- "IODMDocMan" on page 1001
- "IODMDocMan2" on page 1013
- "IODMQuery" on page 1023

ODMGetODMInterface

When an application calls the ODMA connection manager's ODMRegisterApp function, the connection manager calls the appropriate DMS's ODMGetODMInterface function to establish a connection with the DMS.

```
HRESULT ODMGetODMInterface(
  REFIID riid,   LPVOID* ppvObj,   LPUNKNOWN pUnkOuter,   LPVOID
pReserved,   LPSTR lpszAppId,   DWORD dwEnvData
 );
```

Parameters, Return Values

For information please see the ODMA 2.0 specification.

Remarks

During initialization, this function performs the following steps:

- If the ODMA integration scheme is disabled, the function returns E_FAIL.
- Detects what features the application supports by reading the Compatibility registry key (HKLM\Software\Hummingbird\PowerDOCS\AppIntegration\Compatibility).
- Creates an instance of the object implementing the ODMA DMS interfaces.
- Locks the Document Management plug-in by establishing a DDE conversation with this module.
- If no errors occurred, the function returns a pointer to the requested DMS interface.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

See Also

"Active (ODMA) Integration" on page 959

ODMA 2.0 Specification

IODMDocMan

The IODMDocMan interface implements all of the basic document management functionality defined by the ODMA 1.0 specification.

Methods

IODMDocMan Methods	Description
IODMDocMan::SelectDoc	This method causes the DMS to return a document ID representing a document that has been selected for some action.
IODMDocMan::OpenDoc	This method causes the DMS to make a document available to the application.
IODMDocMan::SaveDoc	This method tells the DMS that the document should be saved to the document repository.
IODMDocMan::CloseDoc	An application calls this method when it is finished using the document.
IODMDocMan::NewDoc	This function causes the DMS to create a new document profile and return the document ID for the new document to the calling application.
IODMDocMan::SaveAs	This method causes the DMS to return a new document ID for a document that is based on an existing document.
IODMDocMan::Activate	This method causes the DMS to perform actions that do not require cooperation from the calling application.
IODMDocMan::GetDocInfo	An application uses this method to obtain information about a document from the DMS.
IODMDocMan::SetDocInfo	An application uses this method to pass information about the document to the DMS.
IODMDocMan::GetDMSInfo	This method returns information to the application about the currently active DMS.
IODMDocMan::GetLeadMoniker	This method returns ODM_E_NOSUPPORT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #include <odmacom.h>

See Also

"IODMDocMan2" on page 1013

"Active (ODMA) Integration" on page 959

ODMA 2.0 Specification

IODMDocMan::Activate

C++ Syntax

```
HRESULT Activate(
  WORD action,
  LPSTR lpszDocId);
```

Parameters, Return Values

For information please see the ODMA 2.0 specification.

Remarks

This method causes the DMS to perform actions that do not require cooperation from the calling application.

The implementation passes through the method call as follows:

- If action = ODM_DELETE and the user has delete rights on the document specified by lpszDocID, the method deletes the document.
- If action = ODM_SHOWATTRIBUTES and the user has view profile rights on the document, the method brings up the document's profile in the read-only mode.
- If action = ODM_EDITATTRIBUTES and the user has edit profile rights on the document, the method brings up the document's profile to allow editing a profile. Otherwise, if the user has view profile rights, the method brings up the document's profile in the read-only mode and returns ODM_E_CANCEL.
- If action = ODM_VIEWDOC and the user has view content rights on the document, the method previews the document's contents.

- If action = ODM_OPENDOC and the user has both edit profile and edit content rights on the document, the method opens the document for editing in its native application.
- If action = ODM_NEWDOC and lpszDocId specifies a template, the method creates a new document from the template and opens the new document in its native application.
- If action = ODM_SHOWHISTORY, the method displays the document's history.

All other actions are not supported and the method returns ODM_E_NOSUPPORT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <odmacom.h>

See Also

"IODMDocMan" on page 1001

IODMDocMan::CloseDoc

C++ Syntax

```
HRESULT CloseDoc(
  LPSTR lpszDocId,
  DWORD activeTime,
  DWORD pagesPrinted,
  LPVOID sessionData,
  WORD dataLen);
```

Parameters, Return Values

For information please see the ODMA 2.0 specification.

Remarks

An application that has opened a document by calling ODMOpenDoc then calls this method when it is finished using the document.

The implementation passes this call to IODMDocMan2::CloseDocEx disabling unattended document processing.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <odmacom.h>

See Also

"IODMDocMan" on page 1001

"IODMDocMan2" on page 1013

IODMDocMan::GetDMSInfo

C++ Syntax

```
HRESULT GetDMSInfo(
  LPSTR lpszDmsId,
  LPWORD pwVerNo,
  LPDWORD pdwExtensions);
```

Parameters, Return Values

For information please see the ODMA 2.0 specification.

Remarks

This method returns information to the application about the currently active DMS.

The implementation returns "PCDOCS" as lpszDmsId, ODM_API_VERSION (the DMS supports ODMA 2.0) as *pwVerNo, ODM_EXT_QUERY (the DMS supports the Query Extensions defined in ODMA 1.5) as *pdwExtensions.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <odmacom.h>

See Also

"IODMDocMan" on page 1001

IODMDocMan::GetDocInfo

C++ Syntax

```
HRESULT GetDocInfo(
  LPSTR lpszDocId,
  WORD item,
  LPSTR lpszData,
```

```
WORD dataLen);
```

Parameters, Return Values

For information please see the ODMA 2.0 specification.

Remarks

An application uses this method to obtain information about a document from the DMS.

The implementation passes through the method call as follows:

- If item = ODM_AUTHOR, the method returns the document's author (value of the AUTHOR_ID column).
- If item = ODM_CONTENTFORMAT, the method returns the document's native application identifier (value of the APP_ID column).
- If item = ODM_CREATEDBY, the method returns the document's typist (value of the TYPIST_ID column).
- If item = ODM_NAME, the method returns the document's name. To provide proper call processing, the result can be the same as for item = ODM_TITLETEXT for some applications (MS Word).
- If item = ODM_SUBJECT, the method returns the document abstract (the value of the ABSTRACT column).
- If item = ODM_TYPE, the method returns the document type (value of the TYPE_ID column). To provide proper call processing, the type can be empty for some applications (Quattro Pro).
- If item = ODM_DOCID_LATEST, the method returns the document ID for the latest version of the document represented by lpszDocId. If lpszDocId is the latest version, it will be the same document ID.
- If item = ODM_DOCVERSION, the method returns the label for the document's version represented by lpszDocId.
- If item = ODM_DOCVERSION_LATEST, the method returns the label for the latest version of the document represented by lpszDocId.

- If item = ODM_MODIFYDATE, the method returns the date time of day the document was last edited.
- If item = ODM_MODIFYDATE_LATEST, the method returns the date time of day the latest version for the document represented by lpszDocId was last edited.
- If item = ODM_TITLETEXT, the method returns the name without extension of the document's file.
- If item = ODM_TITLETEXT_RO, the method returns the name without extension of the document's file plus "[Read Only]".
- If item = ODM_DMS_DEFINED, the method returns the value of the document's profile column represented by lpszData.

All other attributes are not supported and the method returns ODM_E_ITEM.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <odmacom.h>

See Also

"IODMDocMan" on page 1001

IODMDocMan::GetLeadMoniker

C++ Syntax

```
HRESULT GetLeadMoniker(
  LPSTR lpszDocId,
  LPMONIKER* ppMoniker);
```

Parameters, Return Values

For information please see the ODMA 2.0 specification.

Remarks

This method returns ODM_E_NOSUPPORT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <odmacom.h>

See Also

"IODMDocMan" on page 1001

IODMDocMan::NewDoc

C++ Syntax

```
HRESULT NewDoc(
  LPSTR lpszDocId,
  DWORD dwFlags,
  LPSTR lpszFormat,
  LPSTR lpszDocLocation);
```

Parameters, Return Values

For information please see the ODMA 2.0 specification.

Remarks

This function causes the DMS to create a new document profile and return the document ID for the new document to the calling application.

The implementation returns the null document ID, ::ODMA\PCDOCS\NULL, if the ODM_SILENT flag is specified. Otherwise, the method returns ODM_E_USERINT. A new profile will be actually created when saving this document.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <odmacom.h>

See Also

"IODMDocMan" on page 1001

IODMDocMan::OpenDoc

C++ Syntax

```
HRESULT OpenDoc(
  DWORD flags,
  LPSTR lpszDocId,
  LPSTR lpszDocLocation);
```

Parameters, Return Values

For information please see the ODMA 2.0 specification.

Remarks

This method causes the DMS to make a document available to the application. It performs any necessary pre-processing (mapping network drives, checking security, etc.) and then returns to the application a temporary filename that can be used to access the document during the current session. Note that this function does not open the document file; it merely makes the file temporarily available to the calling application. The application can then open, read, write, and close the file as needed.

The implementation passes through the method call as follows:

- If lpszDocId represents a new non-profiled document the method returns ODM_E_USERINT.
- If the document is already opened (registered in the document map) and the flag ODM_REFCOPY is not specified, the method returns ODM_E_INUSE. Otherwise, the document is copied to a temporary file.
- If the document is opened by the ODMA integration scheme in the read-only mode and the ODM_MODIFYMODE flag is specified, the method returns ODM_E_ACCESS.
- If the document is not marked as 'being opened' (this mark can be set after a previous call to IODMDocMan2::SaveAsEx) and opened by the ODMA ISM and the ODM_MODIFYMODE flag is specified, the method checks whether the document is locked and the user is able to edit the specified version of the document. Otherwise, an error code is returned.
- The method checks whether user's rights allow performing the request. The user must be granted retrieve document rights. Besides, he/she must have copy document rights if flags = ODM_REFCOPY, edit document rights if flags = ODM_MODIFYMODE, and view document rights if flags = ODM_VIEWMODE.
- The document is downloaded if it has not been yet downloaded from the server.

- If the document is marked as 'being opened' or already opened in the read-write mode by the ODMA ISM and the ODM_VIEWMODE flag is specified, the document is unlocked.
- If the ODM_VIEWMODE flag is specified, the document's file is assigned read-only attributes.
- If the ODM_MODIFYMODE is specified and the document is not already opened by the ODMA ISM or marked as 'being opened', it is locked. If the document is not marked as 'being opened' or is opened by the ODMA ISM, the calling application and the document's native application do not coincide, and lpszDocId does not specify a document's attachment, the document's native application is changed to the calling application. Those changes would not be saved until the profile is saved.
- The document is stored in the document map. The document's mark 'being opened' is cleared.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <odmacom.h>

See Also

"IODMDocMan" on page 1001

"IODMDocMan2" on page 1013

IODMDocMan::SaveAs

C++ Syntax

```
HRESULT SaveAs(
  LPSTR lpszDocId,
  LPSTR lpszNewDocId,
  LPSTR lpszFormat,
  ODMSAVEASCALLBACK pcbCallBack,
  LPVOID pInstanceData);
```

Parameters, Return Values

For information please see the ODMA 2.0 specification.

Remarks

This method causes the DMS to return a new document ID for a document that is based on an existing document. An application typically calls this function in response to selecting a File Save As menu option by the user.

The implementation passes this call to IODMDocMan2::SaveAsEx disabling unattended document creation.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <odmacom.h>

See Also

"IODMDocMan" on page 1001

"IODMDocMan2" on page 1013

IODMDocMan::SaveDoc

C++ Syntax

```
HRESULT SaveDoc(
  LPSTR lpszDocId,
  LPSTR lpszNewDocId);
```

Parameters, Return Values

For information please see the ODMA 2.0 specification.

Remarks

This method tells the DMS that the document should be saved into the document repository. An application typically calls this function after saving changes to the temporary file returned by IODMDocMan::OpenDoc. A new document ID is returned to the application that will be used for all subsequent operations on the document.

The implementation calls IODMDocMan2::SaveDocEx by passing a pointer to ODM_VERSION_SAME as the pdwFlags parameter.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <odmacom.h>

See Also

"IODMDocMan" on page 1001

"IODMDocMan2" on page 1013

IODMDocMan::SelectDoc

C++ Syntax

```
HRESULT SelectDoc(
  LPSTR lpszDocId,
  LPDWORD pdwFlags);
```

Parameters, Return Values

For information please see the ODMA 2.0 specification.

Remarks

This method causes the DMS to return a document ID representing a document that has been selected for some action. An application typically calls this function whenever the user needs to select a document to be opened or imported.

The implementation passes all calls to IODMDocMan2::SelectDocEx that will return one selected document.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <odmacom.h>

See Also

"IODMDocMan" on page 1001

IODMDocMan::SetDocInfo

C++ Syntax

```
HRESULT SetDocInfo(
  LPSTR lpszDocId,
  WORD item,
  LPSTR lpszData);
```

Parameters, Return Values

For information please see the ODMA 2.0 specification.

Remarks

An application uses this method to pass information about the document to the DMS.

The implementation passes through the method call as follows:

- If item = ODM_AUTHOR, the method sets the document's author (value of the AUTHOR_ID column) to lpszData.
- If item = ODM_CREATEDBY, the method sets the document's typist (value of the TYPIST_ID column) to lpszData.
- If item = ODM_NAME, the method sets the document's name to lpszData.
- If item = ODM_SUBJECT, the method sets the document abstract (value of the ABSTRACT column) to lpszData.
- If item = ODM_TYPE, the method sets the document type (value of the TYPE_ID column) to lpszData.
- If item = ODM_DMS_DEFINED, lpszData must contain both the name of the document's profile column to be set and its new value separated by '='. The method assigns the column the new value represented by the second half of lpszData.

All other attributes are not supported and the method returns ODM_E_ITEM.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <odmacom.h>

See Also

"IODMDocMan" on page 1001

IODMDocMan2

The IODMDocMan2 interface implements the extended document management functionality defined in the ODMA 2.0 specification.

Methods

IODMDocMan2 Methods	Description
IODMDocMan2::CloseDocEx	**IODMDocMan2::CloseDocEx** is the same as **IODMDocMan::CloseDoc** except it has a *pwdFlags* parameter.
IODMDocMan2::SaveAsEx	This method extends **IODMDocMan::SaveAs** by adding a *pdwFlags* parameter to allow for unattended document creation.
IODMDocMan2::SaveDocEx	This method tells the DMS that the document should be saved into the document repository.
IODMDocMan2::SelectDocEx	**IODMDocMan2::SelectDocEx** extends **IODMDocMan::SelectDoc**.
IODMDocMan2::QueryCapability	This method is used by a client application to determine if the DMS supports a given ODMA function, or a specific action code in **IODMDocMan::Activate**.
IODMDocMan2::SetDocEvent	An application uses this method to pass event information about the document to the DMS.
IODMDocMan2::GetAlternateContent	This method causes the DMS to return an alternate content file for the specified document.
IODMDocMan2::SetAlternateContent	This method allows an application to create or update an alternate content file for the specified document in the DMS.
IODMDocMan2::GetDocRelation	An application uses this method to retrieve pointers to document versions linked to a particular document ID.
IODMDocMan2::SetDocRelation	An application uses this method to maintain relationship and hierarchy information in the DMS about a compound document structure that it is manipulating externally to the DMS.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #include <odmacom.h>

See Also

"IODMDocMan" on page 1001

"Active (ODMA) Integration" on page 959

ODMA 2.0 Specification

IODMDocMan2::CloseDocEx

C++ Syntax

```
HRESULT CloseDocEx(
  LPSTR lpszDocId,
  LPDWORD pdwFlags,
  DWORD activeTime,
  DWORD pagesPrinted,
  LPVOID sessionData,
  WORD dataLen);
```

Parameters, Return Values

For information please see the ODMA 2.0 specification.

Remarks

IODMDocMan2::CloseDocEx is the same as IODMDocMan::CloseDoc except it has a pwdFlags parameter. There is the ODM_SILENT flag to facilitate unattended document processing.

The implementation passes through the method call as follows:

- If the document's profile needs to be saved, the method saves the profile.
- If pagesPrinted is not 0 or -1, the method updates the document's activity log by adding a print activity.
- If the document was locked, the method unlocks it.
- Removes the document from the internal document map.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <odmacom.h>

See Also

"IODMDocMan2" on page 1013

IODMDocMan2::GetAlternateContent

C++ Syntax

```
HRESULT GetAlternateContent(
  LPSTR lpszDocId,
  LPDWORD pdwFlags,
  LPSTR lpszFormat,
  LPSTR lpszDocLocation);
```

Parameters, Return Values

For information please see the ODMA 2.0 specification.

Remarks

This method causes the DMS to return an alternate content file for the specified document.

The implementation returns ODM_E_NOSUPPORT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <odmacom.h>

See Also

"IODMDocMan2" on page 1013

IODMDocMan2::GetDocRelation

C++ Syntax

```
HRESULT GetDocRelation(
  LPSTR lpszDocId,
  LPDWORD pdwFlags,
  LPSTR lpszLinkedId,
  LPSTR lpszFormat,
  LPSTR lpszPreviousId);
```

Parameters, Return Values

For information please see the ODMA 2.0 specification.

Remarks

An application uses this method to retrieve pointers to document versions linked to a particular document ID.

The implementation returns ODM_E_NOSUPPORT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <odmacom.h>

See Also

"IODMDocMan2" on page 1013

IODMDocMan2::QueryCapability

C++ Syntax

```
HRESULT QueryCapability(
  LPCSTR lpszDmsId,
  DWORD function,
  DWORD item,
  DWORD flags);
```

Parameters, Return Values

For information please see the ODMA 2.0 specification.

Remarks

This method is used by a client application to determine if the DMS supports a given ODMA function, or a specific action code in IODMDocMan::Activate. It can also be used to determine if the DMS supports getting or setting a given document attribute using IODMDocMan::GetDocInfo or IODMDocMan::SetDocInfo.

The implementation returns ODM_SUCCESS for all implemented functions and items (see the description of appropriate methods) or else ODM_E_NOSUPPORT or ODM_E_ITEM.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <odmacom.h>

See Also

"IODMDocMan2" on page 1013

IODMDocMan2::SaveAsEx

C++ Syntax

```
HRESULT SaveAsEx(
  LPSTR lpszDocId,
  LPSTR lpszNewDocId,
  LPSTR lpszFormat,
  ODMSAVEASCALLBACK pcbCallBack,
  LPVOID pInstanceData,
  LPDWORD pdwFlags);
```

Parameters, Return Values

For information please see the ODMA 2.0 specification.

Remarks

This method extends IODMDocMan::SaveAs by adding a pdwFlags parameter to allow for unattended document creation.

The implementation passes through the method call as follows:

- If the ODM_SILENT flag is specified, the method returns ODM_E_USERINT to indicate that the DMS does not support document creation without user interaction.
- If pcbCallBack is not NULL, the method calls the callback function and allows the user to select a document format.
- If lpszDocId is NULL or the null document ID or an existing document was opened in the read-only mode, the method creates a new DM document on the library the user selected, adds the document to the folders the user specified when creating the document, and stores the document in the document map. Otherwise, the method calls IVersion::ShowSaveAsOptionsForm to allow the user to indicate what he/she wants to do.
- If the user refused to create a new DM document or pressed a Cancel button in the DM Extensions Save As dialog and application's native dialogs are not disabled, a message box prompting if the user wants to bring up an application's native Save As dialog pops up.
- If a new version/subversion/document should be created, the method builds a new document ID, assigns it to lpszNewDocId,

adds the document to the document map, and marks it as 'being opened'.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <odmacom.h>

See Also

"IODMDocMan2" on page 1013

IODMDocMan2::SaveDocEx

C++ Syntax

```
HRESULT SaveDocEx(
  LPSTR lpszDocId,
  LPSTR lpszNewDocId,
  LPDWORD pdwFlags);
```

Parameters, Return Values

For information please see the ODMA 2.0 specification.

Remarks

This method tells the DMS that the document should be saved into the document repository. IODMDocMan2::SaveDocEx is basically the same as IODMDocMan::SaveDoc, except it has a pwdFlags parameter. This provides the application with more control during the document creation or save process. There is an ODM_SILENT flag that facilitates unattended document creation or saving.

The implementation passes through the method call as follows:

- If the document's profile needs to be saved, the method saves the profile.
- If the ODM_VERSION_SAME flag is specified, the document is uploaded to the server. Otherwise, the method saves the document as a new version/subversion. The method then locks the newly created version, unlocks the previous one, builds a new document ID for the new version, and replaces the previous version by the new one in the document map.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <odmacom.h>

See Also

"IODMDocMan2" on page 1013

IODMDocMan2::SelectDocEx

C++ Syntax

```
HRESULT SelectDocEx(
  LPSTR lpszDocIds,
  LPWORD pwDocIdsLen,
  LPWORD pwDocCount,
  LPDWORD pdwFlags,
  LPSTR lpszFormatFilter);
```

Parameters, Return Values

For information please see the ODMA 2.0 specification.

Remarks

IODMDocMan2::SelectDocEx extends IODMDocMan::SelectDoc. This method causes the DMS to return one or more document IDs representing documents that have been selected for some action.

The implementation passes through the method call as follows:

- If the ODM_SILENT flag is specified, the method returns ODM_E_USERINT to indicate that the DMS cannot select a document without user interaction.
- Brings up a Quick Retrieve form in order for the user to select one or more documents.
- If the user pressed a Cancel button in the form and application's native dialogs are not disabled, a message box prompting if the user wants to open an application's native dialog pops up.
- Builds document IDs for the first *pdwDocCount selected documents (these document IDs are copied to lpszDocIds). All other selected documents will be opened through the AIM separately.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <odmacom.h>

See Also

"IODMDocMan2" on page 1013

IODMDocMan2::SetAlternateContent

C++ Syntax

```
HRESULT SetAlternateContent(
  LPSTR lpszDocId,
  LPDWORD pdwFlags,
  LPSTR lpszFormat,
  LPSTR lpszDocLocation);
```

Parameters, Return Values

For information please see the ODMA 2.0 specification.

Remarks

This method allows an application to create or update an alternate content file for the specified document in the DMS.

The implementation returns ODM_E_NOSUPPORT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <odmacom.h>

See Also

"IODMDocMan2" on page 1013

IODMDocMan2::SetDocEvent

C++ Syntax

```
HRESULT SetDocEvent(
  LPSTR lpszDocId,
  DWORD flags,
  DWORD event,
  LPVOID lpData,
```

```
  DWORD dwDataLen,
  LPSTR lpszComment);
```

Parameters, Return Values

For information please see the ODMA 2.0 specification.

Remarks

An application uses this method to pass event information about the document to the DMS. Events like printing, sending, posting can be sent by the application to the DMS.

The implementation returns ODM_E_NOSUPPORT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <odmacom.h>

See Also

"IODMDocMan2" on page 1013

IODMDocMan2::SetDocRelation

C++ Syntax

```
HRESULT SetDocRelation(
  LPSTR lpszDocId,
  LPDWORD pdwFlags,
  LPSTR lpszLinkedId,
  LPSTR lpszFormat,
  LPSTR lpszPreviousId);
```

Parameters, Return Values

For information please see the ODMA 2.0 specification.

Remarks

An application uses this method to maintain relationship and hierarchy information in the DMS about a compound document structure that it is manipulating externally to the DMS.

The implementation returns ODM_E_NOSUPPORT.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <odmacom.h>

See Also

"IODMDocMan2" on page 1013

IODMQuery

The IODMQuery interface implements the query extension of the ODMA 1.5 specification.

Methods

IODMQuery Methods	Description
IODMQuery::QueryExecute	This method executes a query represented.
IODMQuery::QueryGetResults	This method fetches a number of documents at a time.
IODMQuery::QueryClose	This method is used by a client application to indicate that it has finished using a query.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <odmacom.h>

See Also

"Active (ODMA) Integration" on page 959

ODMA 2.0 Specification

IODMQuery::QueryClose

C++ Syntax

```
HRESULT QueryClose(LPCSTR dmsQueryId);
```

Parameters, Return Values

For information please see the ODMA 2.0 specification.

Remarks

This method is used by a client application to indicate that it has finished using the query and that the DMS should release any resources and/or memory for the result set.

The implementation releases all the objects it stored and removes the query from the internal map.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <odmacom.h>

See Also

"IODMQuery" on page 1023

IODMQuery::QueryExecute

C++ Syntax

```
HRESULT QueryExecute(
  LPCSTR lpszQuery,
  LPSTR dmsQueryId);
```

Parameters, Return Values

For information please see the ODMA 2.0 specification.

Remarks

This method executes the query represented by lpszQuery.

The implementation passes through the method call as follows:

- Creates a search on the DM primary library.
- Parses the query string lpszQuery to specify the criteria and return columns of the created search.
- Executes the search.
- Builds a query ID and assigns it to the query.
- Stores the query's results in an internal map.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <odmacom.h>

See Also

"IODMQuery" on page 1023

IODMQuery::QueryGetResults

C++ Syntax

```
HRESULT QueryGetResults(
  LPCSTR queryId,
  LPSTR lpszDocId,
  LPSTR lpszDocName,
  WORD docNameLen,
  LPWORD pwDocCount);
```

Parameters, Return Values

For information please see the ODMA 2.0 specification.

Remarks

This method fetches the pwDocCount number of documents at a time.

The implementation returns the specified number of the document IDs of documents satisfied the search referenced in queryId in lpszDocId and their names in lpszDocName.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import <odmacom.h>

See Also

"IODMQuery" on page 1023

Document Attributes

The following table maps document attributes or properties defined by ODMA to DM document columns. Applications can get attribute values for a document by using the IODMDocMan::GetDocInfo method. Some attribute values can be set using the

IODMDocMan::SetDocInfo method. The attributes are also used when specifying query's criteria.

Attribute/Item ID	DM Column
ODM_AUTHOR	PROFILE.AUTHOR.USER_ID; PEOPLE.SYSTEM_ID (AUTHOR_ID)
ODM_CONTENTFORMAT	PROFILE.APPLICATION.APPLICATION; APPS.SYSTEM_ID (APP_ID)
ODM_CREATEDBY	PROFILE.TYPIST.USER_ID; PEOPLE.SYSTEM_ID (TYPIST_ID)
ODM_NAME	PROFILE.DOCNAME
ODM_SUBJECT	PROFILE.ABSTRACT
ODM_TYPE	PROFILE.DOCUMENTTYPE.TYPE_ID; DOCUMENTTYPES.SYSTEM_ID (TYPE_ID)
ODM_DOCVERSION	VERSIONS.VERSION_LABEL
ODM_MODIFYDATE	VERSIONS.LASTEDITDATE + VERSIONS.LASTEDITTIME

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #include <odmacom.h>

See Also

"Active (ODMA) Integration" on page 959

ODMA 2.0 Specification

Chapter

10

CMS Controls

In This Chapter

- "Using CMS Forms" on page 1028
- "CMS Enumerations" on page 1032
- "CMS Objects" on page 1033

Using CMS Forms

- "How to use the CMSForm control" on page 1028
- "How to create a custom form" on page 1028
- "How to replace standard form with your custom form" on page 1031
- "How to register our sample form" on page 1031

How to use the CMSForm control

1. Place the CMSForm ActiveX control on your form.
2. During initialization call its Init() method, passing the QBEMode flag (TRUE if control lives on the QBE form) and a form description. You can obtain a form description from the Library object (Forms collection).
3. Handle all necessary events of the CMSForm control. You may call lookup forms, validate user input and display error message to the user and so on.
4. You can refer to controls on the CMSForm by control's name (CMSForm.Controls("NAME")) in your code.

You can refer to the example in the next section.

How to create a custom form

1. Define the form you want to replace.
2. Create COM object which will implement the interface of the form, you need to replace.
3. Implement methods of that interface as appropriate for your needs. You may have to call DMO methods or modify DM object properties in your code.
4. Register your form as described below.

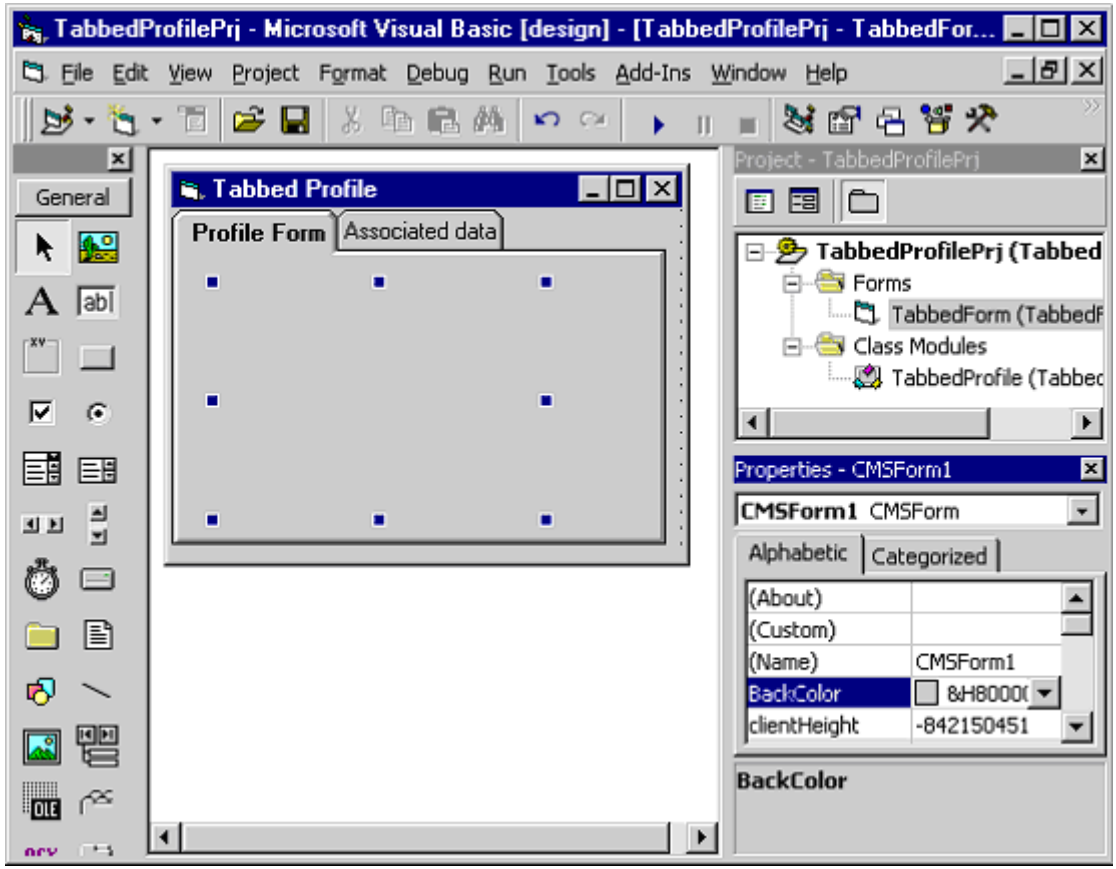
Example

We will now replace a profile form with our own custom profile. Our form will contain the same profile form but we want to put it on the

two-page tab control. Second page will contain the ActiveX Chart control for illustrative purposes.

In our sample we will concentrate on most important details and will not discuss resizing and related issues.

1. Create a new ActiveX DLL project in VB5.
2. Add a new form to your project and add the CMSForm control as well as the Tab control to that form. This is shown below:



3. following is the code for the class module for the form:

```
' Tells VB that we are going to implement an IprofileForm
interface in our module
Implements IProfileForm
```

```
' That interface ontains a single metod – ShowProfileForm
' This is how you should write method declaration in VB:
Public Function IProfileForm_ShowProfileForm(ByVal p As IProfile,
ByVal bAllowEdits As Boolean) As Integer
    ' load and initialize our Profile form
    Load TabbedForm

    TabbedForm.Init p, bAllowEdits

    'show the Profile form modally
    TabbedForm.Show vbModal

    'we are here if user selects OK or Cancel button
    ' do necessary post-processing

    ' return appropriate value
    IProfileForm_ShowProfileForm = FR_OK
End Function__
```

4 and script, associated with TabbedForm (shows how you should process some of the CMSForm's events):

```
Public Sub Init(p As IProfile, bAllowEdits As Boolean)
    ' tell the CMSForm control to recreate the form from its
definition
    CMSForm1.Init False,
p.Library.Forms(p.formName).FormDefinition

    ' should also
    '   store the profile reference so it can be used later
    '   init profile controls with their values from the p.Columns
collection
    '   properly resize form and control on it
End Sub

Private Sub CMSForm1_ButtonPressed(ByVal ControlName As String)
    ' button click event

    If ControlName = "OK" Or ControlName = "CANCEL" Then
        ' should close the form, possibly rollback changes(CANCEL)
and so on
```

```
        Me.Hide

    ElseIf ControlName = "TRUSTEES" Then

        ' Edit Security Dialog requested

        ' should call p.ShowAccessForm ...

    End If

End Sub


Private Sub CMSForm1_DoLookup(ByVal ControlName As String)

    ' should display the lookup form

    MsgBox "lookup request from control " & ControlName

End Sub__
```

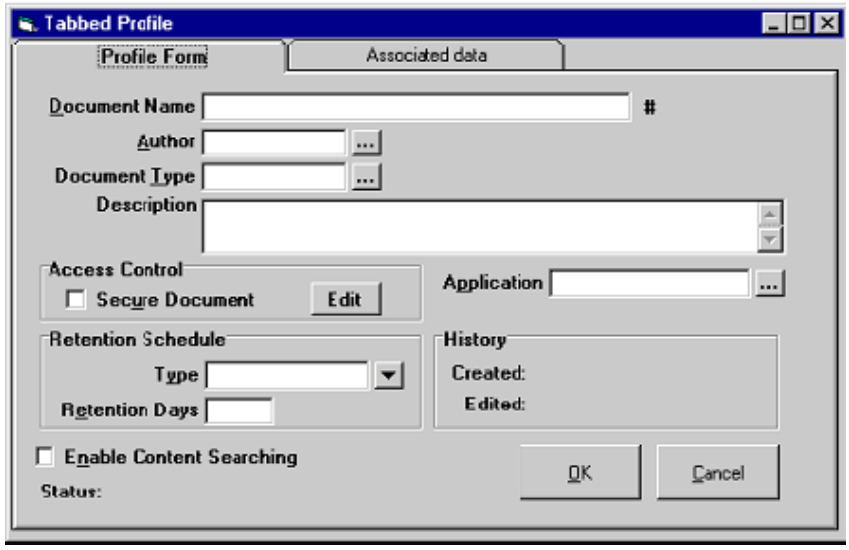
5. Now compile a project to make DLL and proceed to the next section to register and test our form.

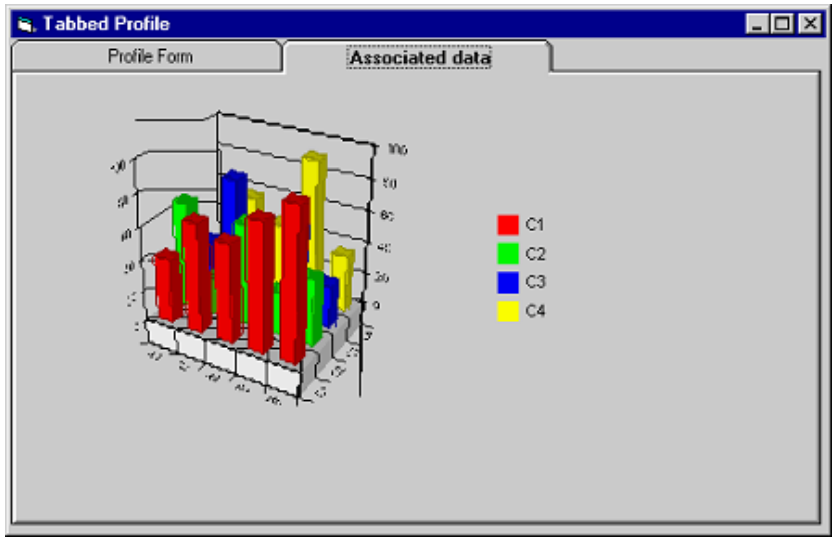
How to replace standard form with your custom form

1. Under HKLM\Software\PCD\Forms find a string value, which corresponds to your form.
2. Replace its value with CLSID or ProgID of your custom form.
3. Restart DM Extensions so your changes will take effect.

How to register our sample form

1. Under HKLM\Software\PCD\Forms replace a ProfileForm's value with the string "TabbedFormPrj.TabbedForm".
2. Restart DM Extensions, select a document and select the Display Profile command.
3. You should see the form, similar to pictured below.
4. On the second page you will see your custom ActiveX control.





CMS Enumerations

- "ControlTypeEnum Enumeration " on page 1032

ControlTypeEnum Enumeration

Types of controls stored on CMSForm.

Elements

ControlTypeEnum Elements	Description
typeCMSBox	Specify CMSBox control type.
typeCMSCheckBox	Specify CMSCheckBox control type.
typeCMSEdit	Specify CMSEdit control type.
typeCMSList	Specify CMSList control type.
typeCMSPush	Specify CMSPush control type.
typeCMSRadioGroup	Specify CMSRadioGroup control type.
typeCMSText	Specify CMSText control type.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSBox::ControlType" on page 1038

"CMSCheckBox::ControlType" on page 1055

"CMSEdit::ControlType" on page 1074

"CMSList::ControlType" on page 1140

"CMSPush::ControlType" on page 1157

"CMSRadioGroup::ControlType" on page 1173

"CMSText::ControlType" on page 1189

CMS Objects

- "CMSBox" on page 1034
- "CMSCheckBox" on page 1049
- "CMSEdit" on page 1068
- "CMSForm" on page 1116

- “CMSList” on page 1136
- “CMSPush” on page 1153
- “CMSRadioGroup” on page 1169
- “CMSText” on page 1184

CMSBox

The CMSBox control provides an identifiable grouping for controls. You can also use a Frame to subdivide a form functionally—for example, to separate groups of CMSRadioGroup controls. Except described the control supports also standard properties, methods, and events described in Visual Basic.

Methods

CMSBox Methods	Description
CMSBox::Init	The method initializes CMSBox.
CMSBox::SetFocus	The method moves the focus to the specified object.

Properties

CMSBox Properties	Description
CMSBox::BackColor	The property returns or sets the background color of the control prompt.
CMSBox::Border	The property returns or sets the draw border.
CMSBox::ControlType	The property returns the type of the control.
CMSBox::Enabled	The property returns or sets a value that determines whether a control can respond to user-generated events.
CMSBox::Font	The property returns or sets a Font object.
CMSBox::ForeColor	The property returns or sets the foreground color used to display text and graphics in an object.
CMSBox::Image	The property returns or sets a handle, provided by Microsoft Windows, to a persistent bitmap.
CMSBox::ImageType	The property is not used.
CMSBox::Name	The property returns the name used in the code to identify an object.
CMSBox::Opaque	The property returns or sets a value that determines whether the control's BackColor will fill the control and obscure all the colors or graphics behind it.

CMSBox Properties	Description
CMSBox::Prompt	The property returns or sets the prompt for the box.
CMSBox::Stretch	The property returns or sets a value, which determines whether or not graphics change a size up to the size of an Image control.
CMSBox::TabStop	The property returns or sets the value indicating whether or not a user can use the TAB key to set the focus to an object.
CMSBox::Visible	The property returns or sets a value that determines whether the control is visible or hidden.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

CMSBox::BackColor

The property returns or sets the background color of the control prompt.

Syntax

```
object.BackColor [ = color ]
```

```
Color
```

An OLE_COLOR value or constant that determines the background colors of an object.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
```

```
CMSBox1.BackColor = vbButtonText
```

```
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSBox::Font" on page 1040

"CMSBox::ForeColor" on page 1041

"CMSBox::Prompt" on page 1045

"CMSBox" on page 1034

"CMSCheckBox" on page 1049

CMSBox::Border

The property returns or sets a draw border.

Syntax

```
object.Border [ = long ]
```

`Long`

A numeric expression specifying the border type. Can be one of the following values:

Value	Description
0	None (Default)
1	Thin
2	Thick
3	Underline
4	Shadow
5	Raised
6	Recessed
7	Thin Raised
8	Thin Recessed
9	Button Up
10	Button Down
11	Default Button Up

Value	Description
12	Default Button Down
13	Combo Up
14	Combo Down

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSBox1.Border = 9
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSBox::BackColor" on page 1036

"CMSBox::Font" on page 1040

"CMSBox::Visible" on page 1048

"CMSBox" on page 1034

"CMSCheckBox" on page 1049

CMSBox::ControlType

The property returns the type of the control.

Syntax

```
object.ControlType
```

A numeric (long) expression specifying the control type. The value of the property is equal to 0 (the constant is typeCMSBox).

Access:

Read-Only

Example

(Visual Basic 5.0)

```
Private Sub ...
whatControlType = CMSBox1.ControlType
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSBox::Enabled" on page 1039

"CMSBox::Opaque" on page 1044

"CMSBox" on page 1034

"CMSCheckBox" on page 1049

CMSBox::Enabled

The property returns or sets a value that determines whether a control can respond to user-generated events.

Syntax

```
object.Enabled [ = boolean ]
Boolean
```

A Boolean value determines that an object can respond to events (Default) or prevents an object from responding to events (False).

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSBox1.Enabled = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSBox::Image" on page 1041

"CMSBox::Name" on page 1044

"CMSBox::Visible" on page 1048

"CMSBox" on page 1034

"CMSCheckBox" on page 1049

CMSBox::Font

The property returns or sets a Font object.

Syntax

```
object.Font
StdFont
```

The StdFont value that determines a Font object.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSBox1.Font.Bold = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSBox::Border" on page 1037

"CMSBox::ForeColor" on page 1041

"CMSBox::Opaque" on page 1044

"CMSBox" on page 1034

“CMSCheckBox” on page 1049

CMSBox::ForeColor

The property returns or sets the foreground color used to display text and graphics in an object.

Syntax

```
object.ForeColor [ = color ]
Color
```

An OLE_COLOR value or constant that determines the foreground colors of an object.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSBox1.ForeColor = vbHighLight
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

“CMSBox::BackColor” on page 1036

“CMSBox::Font” on page 1040

“CMSBox::Image” on page 1041

“CMSBox” on page 1034

“CMSCheckBox” on page 1049

CMSBox::Image

The property returns or sets a handle, provided by Microsoft Windows, to a persistent bitmap.

Syntax

```
object.Image [ = StdPicture ]
```

```
StdPicture
```

The StdPicture object contains the given image for an object.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSBox1.Image = Load(Images(5))
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSBox::ImageType" on page 1042

"CMSBox::Name" on page 1044

"CMSBox::Stretch" on page 1047

"CMSBox" on page 1034

"CMSCheckBox" on page 1049

CMSBox::ImageType

The property is not used.

Syntax

```
object.ImageType [ = long ]
Long
```

A numeric expression specifying the image type.

Access:

Read-write

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSBox::Image" on page 1041

"CMSBox" on page 1034

"CMSCheckBox" on page 1049

CMSBox::Init

The method initializes CMSBox.

Syntax

```
object.Init lFlags, formDescription
```

`lFlags`

Required. A long value indicates the mode of the control. The parameter can be:

Constant	Value	Description
frmProfile	0	Profile mode
frmQBE	1	QBE mode
frmReadOnly	2	Read Only
frmDefaultMode	4	Default mode

`formDescription`

Required. A string contains the CMSBox description.

Return Values

If the numeric (long) value is equal to zero then the method succeeds. If the value is not equal to zero then the method fails.

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim valueReturn as Long
ValueReturn=CMSBox1.Init(4, "You can show this box")
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSBox::SetFocus" on page 1046

"CMSBox" on page 1034

"CMSCheckBox" on page 1049

CMSBox::Name

The property returns the name used in the code to identify an object.

Syntax

```
object.Name [ = string ]
```

A string expression specifying an object.

Access:

Read-Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSBox::Enabled" on page 1039

"CMSBox::Image" on page 1041

"CMSBox::Prompt" on page 1045

"CMSBox" on page 1034

"CMSCheckBox" on page 1049

CMSBox::Opaque

The property returns or sets a value that determines whether the control's BackColor will fill the control and obscure all the colors or graphics behind it.

Syntax

```
object.Opaque [ = boolean ]
```

```
Boolean
```

A Boolean value determines that the object is transmittance (True) else the opaque (Default) .

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSBox1.Opaque = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSBox::BackColor" on page 1036

"CMSBox::Image" on page 1041

"CMSBox::Visible" on page 1048

"CMSBox" on page 1034

"CMSCheckBox" on page 1049

CMSBox::Prompt

The property returns or sets the prompt for the box.

Syntax

```
object.Prompt [ = string ]
String
```

A string value determines a prompt for the object.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim line1 as String
```

```
Line1 = CMSBox1.Prompt
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSBox::Border" on page 1037

"CMSBox::Name" on page 1044

"CMSBox::TabStop" on page 1047

"CMSBox" on page 1034

"CMSCheckBox" on page 1049

CMSBox::SetFocus

The method moves the focus to the specified object.

Syntax

```
object.SetFocus
```

Example

(Visual Basic 5.0)

```
Private Sub Test1()
If MsgBox("Do you want to set focus in the CMSBox control?", _
vbYesNo) = vbYes Then CMSBox1.SetFocus
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSBox::Init" on page 1043

"CMSBox" on page 1034

"CMSCheckBox" on page 1049

CMSBox::Stretch

The property returns or sets a value, which determines whether or not graphics change a size up to the size of an Image control.

Syntax

```
object.Stretch [ = boolean ]
Boolean
```

A Boolean value determines that a graphic resizes (True) else the graphic is shown as described in Settings.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSBox1.Stretch = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSBox::Image" on page 1041

"CMSBox::Name" on page 1044

"CMSBox::Visible" on page 1048

"CMSBox" on page 1034

"CMSCheckBox" on page 1049

CMSBox::TabStop

The property returns or sets the value indicating whether or not a user can use the TAB key to set the focus to an object.

Syntax

```
object.TabStop [ = boolean ]
Boolean
```

If a value is equal to False then bypasses the object when the user is tabbing.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSBox1.TabStop = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSBox::Font" on page 1040

"CMSBox::Name" on page 1044

"CMSBox::Opaque" on page 1044

"CMSBox" on page 1034

"CMSCheckBox" on page 1049

CMSBox::Visible

The property returns or sets a value that determines whether the control is visible or hidden.

Syntax

```
object.Visible [ = boolean ]
Boolean
```

A Boolean value determines that the control is visible (Default) or hidden (False).

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSBox1.Visible = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSBox::Enabled" on page 1039

"CMSBox::Name" on page 1044

"CMSBox::Prompt" on page 1045

"CMSBox" on page 1034

"CMSCheckBox" on page 1049

CMSCheckBox

If the form (CMSForm) is in the QBE mode then the Control has three conditions - two standards and one additional. The additional condition - «undetermined». This condition is ignored when searching. Except described the control supports also standard properties, methods, and events described in Visual Basic.

Methods

CMSCheckBox Methods	Description
CMSCheckBox::Init	The method initializes CMSCheckBox.
CMSCheckBox::SetFocus	The method moves the focus to the specified object.

Properties

CMSCheckBox Properties	Description
CMSCheckBox::BackColor	The property returns or sets the background color of the control prompt.
CMSCheckBox::Border	The property returns or sets the draw border.
CMSCheckBox::CheckedTrigger	The property returns a value, which should be equal to a value of the Value property, if being in the Checked state.
CMSCheckBox::ControlType	The property returns the type of the control.
CMSCheckBox::Enabled	The property returns or sets a value that determines whether a control can respond to user-generated events.
CMSCheckBox::Font	The property returns or sets a Font object.
CMSCheckBox::ForeColor	The property returns or sets the foreground color used to display text and graphics in an object.
CMSCheckBox::hasValue	The property returns or sets a value that determines whether the condition of the Control is “undetermined”.
CMSCheckBox::HelpID	The property returns or sets the name of the context sensitive help for this button.

CMSCheckBox Properties	Description
CMSCheckBox::LeftText	The property returns or sets a value that determines whether the caption exists (True) or not (Default).
CMSCheckBox::Name	The property returns the name used in the code to identify an object.
CMSCheckBox::Prompt	The property returns or sets the prompt for the box.
CMSCheckBox::SQLInfo	Internal property, which contains the index on a main database.
CMSCheckBox::TabStop	The property returns or sets the value indicating whether or not a user can use the TAB key to set the focus to an object.
CMSCheckBox::UncheckedTrigger	The property returns a value, which should be equal to a value of the Value property, if being in the Unchecked state.
CMSCheckBox::Value	The property returns the value of the control. Selects from the values of the checkedTrigger and uncheckedTrigger properties.
CMSCheckBox::Visible	The property returns or sets a value that determines whether the control is visible or hidden.

Events

CMSCheckBox Events	Description
CMSCheckBox::DataChanged	Occurs when the data is changed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

CMSCheckBox::BackColor

The property returns or sets the background color of the control prompt.

Syntax

```
object.BackColor [ = color ]
```

```
Color
```

An OLE_COLOR value or constant that determines the background colors of an object.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSCheckBox1.BackColor = vbButtonText
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSCheckBox::Font" on page 1057

"CMSCheckBox::ForeColor" on page 1057

"CMSCheckBox::Prompt" on page 1063

"CMSBox" on page 1034

"CMSCheckBox" on page 1049

CMSCheckBox::Border

The property returns or sets the draw border.

Syntax

```
object.Border [ = long ]
```

```
Long
```

A numeric expression specifying the border type. Can be one of the following values:

Value	Description
0	None (Default)
1	Thin
2	Thick
3	Underline
4	Shadow
5	Raised
6	Recessed
7	Thin Raised
8	Thin Recessed
9	Button Up
10	Button Down
11	Default Button Up
12	Default Button Down
13	Combo Up
14	Combo Down

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSCheckBox1.Border = 4
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSCheckBox::BackColor" on page 1052

"CMSCheckBox::Font" on page 1057

"CMSCheckBox::Visible" on page 1067

"CMSBox" on page 1034

"CMSCheckBox" on page 1049

CMSCheckBox::CheckedTrigger

The property returns a value, which should be equal to a value of the Value property, if being in the Checked state.

Syntax

```
object.CheckedTrigger [ = variant ]
```

Variant

The value, which should be equal to a value of the Value property, if being in the Checked state.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim mvarValue
MvarValue = CMSCheckBox1.CheckedTrigger
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSCheckBox::Font" on page 1057

"CMSCheckBox::Name" on page 1062

"CMSCheckBox::Value" on page 1066

"CMSBox" on page 1034

“CMSCheckBox” on page 1049

CMSCheckBox::ControlType

The property returns the type of the control.

Syntax

```
object.ControlType
```

A numeric (long) expression specifying the control type. The value of the property is equal to 1 (the constant is typeCMSCheckBox).

Access:

Read-Only

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim whatControlType as Long
whatControlType = CMSCheckBox1.ControlType
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

“CMSCheckBox::Enabled” on page 1056

“CMSCheckBox::Name” on page 1062

“CMSBox” on page 1034

“CMSCheckBox” on page 1049

CMSCheckBox::DataChanged

Occurs when the data is changed.

Syntax

```
Private Sub object_DataChanged()
```

Example

(Visual Basic 5.0)

```
Private Sub CMSCheckBox1_DataChanged()
MsgBox "Contents was changed"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSBox" on page 1034

"CMSCheckBox" on page 1049

CMSCheckBox::Enabled

The property returns or sets a value that determines whether a control can respond to user-generated events.

Syntax

```
object.Enabled [ = boolean ]
Boolean
```

A Boolean value determines that an object can respond to events (Default) or prevents an object from responding to events (False).

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSCheckBox1.Enabled = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSCheckBox::ControlType" on page 1055

"CMSCheckBox::Name" on page 1062

“CMSCheckBox::Visible” on page 1067

“CMSBox” on page 1034

“CMSCheckBox” on page 1049

CMSCheckBox::Font

The property returns or sets a Font object.

Syntax

```
object.Font
StdFont
```

The StdFont value that determines a Font object.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSCheckBox1.Font.Bold = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

“CMSCheckBox::Border” on page 1052

“CMSCheckBox::ForeColor” on page 1057

“CMSCheckBox::Visible” on page 1067

“CMSBox” on page 1034

“CMSCheckBox” on page 1049

CMSCheckBox::ForeColor

The property returns or sets the foreground color used to display text and graphics in an object.

Syntax

```
object.ForeColor [ = color ]
```

```
Color
```

The OLE_COLOR value or constant that determines the foreground colors of an object.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSCheckBox1.ForeColor = vbHighLight
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSCheckBox::BackColor" on page 1052

"CMSCheckBox::Font" on page 1057

"CMSCheckBox::Name" on page 1062

"CMSBox" on page 1034

"CMSCheckBox" on page 1049

CMSCheckBox::hasValue

The property returns or sets a value that determines whether the condition of the control is "undetermined".

Syntax

```
object.hasValue [ = boolean ]
```

```
Boolean
```

A Boolean value that determines whether the condition of the Control is "undetermined" (Default) or not (False).

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSCheckBox1.hasValue = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSCheckBox::Enabled" on page 1056

"CMSCheckBox::Name" on page 1062

"CMSCheckBox::Value" on page 1066

"CMSBox" on page 1034

"CMSCheckBox" on page 1049

CMSCheckBox::HelpID

The property returns or sets the name of the context sensitive help for this button.

Syntax

```
object.HelpID [ = string ]
String
```

The string expression specifying a name of the context sensitive help for this button.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSCheckBox1.HelpID = "txtHelpPush"
```

```
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSCheckBox::hasValue" on page 1058

"CMSCheckBox::Name" on page 1062

"CMSCheckBox::Prompt" on page 1063

"CMSBox" on page 1034

"CMSCheckBox" on page 1049

CMSCheckBox::Init

The method initializes CMSCheckBox.

Syntax

```
object.Init lFlags, formDescription
```

`lFlags`

Required. A long value indicates the mode of the control. The parameter can be:

Constant	Value	Description
frmProfile	0	Profile mode
frmQBE	1	QBE mode
frmReadOnly	2	Read Only
frmDefaultMode	4	Default mode. Values of the CMSCheckBox control have three states: checked, unchecked, and undefined.

`formDescription`

Required. A string contains the CMSCheckBox description.

Return Values

If the numeric (long) value is equal to zero then the method succeeds. If the value is not equal to zero then the method fails.

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim valueReturn as Long
ValueReturn=CMSCheckBox1.Init(4, "You can show this control")
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSCheckBox::SetFocus" on page 1063

"CMSBox" on page 1034

"CMSCheckBox" on page 1049

CMSCheckBox::LeftText

The property returns or sets a value that determines whether the caption exists (True) or not (Default).

Syntax

```
object.LeftText [ = boolean ]
Boolean
```

A Boolean value determines whether the caption exists or not.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSCheckBox1.LeftText = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSCheckBox::Font" on page 1057

"CMSCheckBox::HelpID" on page 1059

"CMSCheckBox::Value" on page 1066

"CMSBox" on page 1034

"CMSCheckBox" on page 1049

CMSCheckBox::Name

The property returns the name used in the code to identify this control.

Syntax

```
object.Name
```

The string expression specifying the name of the control.

Access:

Read-Only

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim currentName as String
CurrentName = CMSCheckBox1.Name
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSCheckBox::Enabled" on page 1056

"CMSCheckBox::hasValue" on page 1058

“CMSCheckBox::Prompt” on page 1063

“CMSBox” on page 1034

“CMSCheckBox” on page 1049

CMSCheckBox::Prompt

The property returns or sets the prompt for the button.

Syntax

```
object.Prompt [ = string ]
```

```
String
```

A string value determines the prompt for an object.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim line1 as String
Line1 = CMSCheckBox1.Prompt
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

“CMSCheckBox::Border” on page 1052

“CMSCheckBox::Name” on page 1062

“CMSCheckBox::TabStop” on page 1065

“CMSBox” on page 1034

“CMSCheckBox” on page 1049

CMSCheckBox::SetFocus

The method moves the focus to the specified object.

Syntax

```
object.SetFocus
```

Example

(Visual Basic 5.0)

```
Private Sub Test1()
If MsgBox("Do you want to set focus in the CMSPush control?", _
vbYesNo) = vbYes Then CMSCheckBox1.SetFocus
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSCheckBox::Init" on page 1060

"CMSBox" on page 1034

"CMSCheckBox" on page 1049

CMSCheckBox::SQLInfo

Internal property, which contains the index on a main database.

Syntax

```
object.SQLInfo [ = string ]
String
```

The string expression specifies the index on a main database.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim info as String
info = CMSCheckBox1.SQLInfoEnd Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSCheckBox" on page 1049

CMSCheckBox::TabStop

The property returns or sets the value indicating whether or not a user can use the TAB key to set the focus to an object.

Syntax

```
object.TabStop [ = boolean ]
```

```
Boolean
```

If a value is equal to False then bypasses the object when the user is tabbing.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSCheckBox1.TabStop = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSCheckBox::Font" on page 1057

"CMSCheckBox::Name" on page 1062

"CMSCheckBox::Visible" on page 1067

"CMSBox" on page 1034

"CMSCheckBox" on page 1049

CMSCheckBox::UncheckedTrigger

The property returns a value, which should be equal to a value of the Value property, if being in the Unchecked state.

Syntax

```
object.UncheckedTrigger [ = variant ]
Variant
```

The value, which should be equal to a value of the Value property, if being in the Unchecked state.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim mvarValue
mvarValue = CMSCheckBox1.UncheckedTrigger
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSCheckBox::Enabled" on page 1056

"CMSCheckBox::hasValue" on page 1058

"CMSCheckBox::Value" on page 1066

"CMSBox" on page 1034

"CMSCheckBox" on page 1049

CMSCheckBox::Value

The property returns the value of the control. Selects from the values of the checkedTrigger and uncheckedTrigger properties.

Syntax

```
object.Value [ = variant ]
```

```
Variant
```

The value of the control.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim mvarValue
mvarValue = CMSCheckBox1.Value
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSCheckBox::hasValue" on page 1058

"CMSCheckBox::Name" on page 1062

"CMSCheckBox::Visible" on page 1067

"CMSBox" on page 1034

"CMSCheckBox" on page 1049

CMSCheckBox::Visible

The property returns or sets a value that determines whether the control is visible or hidden.

Syntax

```
object.Visible [ = boolean ]
Boolean
```

A Boolean value determines that the control is visible (Default) or hidden (False).

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSCheckBox1.Visible = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSCheckBox::Enabled" on page 1056

"CMSCheckBox::Name" on page 1062

"CMSCheckBox::Prompt" on page 1063

"CMSBox" on page 1034

"CMSCheckBox" on page 1049

CMSEdit

This field is intended to input information with a prompt. It can be displayed as ComboBox or lookup (the button with « … ») depending on flags. Multiple values and multiple selections in ComboBox in the QBE mode are allowed. Except described the control supports also standard properties, methods, and events described in Visual Basic.

Methods

CMSEdit Methods	Description
CMSEdit::AddItem	The method adds an item to the CMSEdit control.
CMSEdit::Init	The method initializes CMSEdit control.
CMSEdit::RemoveItem	The method removes an item from the CMSEdit control.
CMSEdit::SetFocus	The method moves the focus to the specified object.

Properties

CMSEdit Properties	Description
CMSEdit::ComboBox	The property returns or sets a value determines that the control represents such control as ComboBox or ListBox.
CMSEdit::ControlType	The property returns the type of the control.
CMSEdit::EditAlignment	The property returns or sets a value alignment of the text box.
CMSEdit::EditBackColor	The property returns or sets the background color of the text box.
CMSEdit::EditBorder	The property returns or sets the border type of the text box.
CMSEdit::EditFont	The property returns or sets the font for the text box.
CMSEdit::EditForeColor	The property returns or sets the foreground color of the text box.
CMSEdit::EditStyles	Used to return or set values of all properties of the control.
CMSEdit::Enabled	The property returns or sets a value that determines whether a control can respond to user-generated events.

CMSEdit Properties	Description
CMSEdit::ErrorText	The property returns or sets the text of the message with information about the error in input.
CMSEdit::Format	The property returns or sets data type for the CMSEidtBox.
CMSEdit::HasValidValue	The property returns or sets the value indicating that field is empty or not.
CMSEdit::HelpID	The property returns or sets the name of the context sensitive help for this button.
CMSEdit::Hscroll	The property returns or sets the value indicating that the horizontal scrolling is permitted or not.
CMSEdit::hWnd	The property returns a handle (from Microsoft Windows) to control's window.
CMSEdit::ListIndex	The property returns or sets selection in the list box.
CMSEdit::List	The property returns or sets item specified by index in the list box.
CMSEdit::LookupName	The property returns or sets the form name for look it up.
CMSEdit::LowerCase	The property returns or sets the value indicating whether or not the output returns the field text as “Lower case” one.
CMSEdit::MaxChars	The property returns or sets the maximal length of the text in a field of input.
CMSEdit::MultiLine	The property returns or sets the value indicating the multi-line condition.
CMSEdit::Name	The property returns the name used in the code to identify an object.

CMSEdit Properties	Description
CMSEdit::Password	The property returns or sets the value that determines whether a control is in the condition “Password”.
CMSEdit::Prompt	The property returns or sets the prompt for the box.
CMSEdit::PromptAlignment	The property returns or sets a value of the prompt alignment.
CMSEdit::PromptBackColor	The property returns or sets the background color of the prompt.
CMSEdit::PromptBorder	The property returns or sets the border type of the prompt.
CMSEdit::PromptFont	The property returns or sets the text font of the prompt.
CMSEdit::PromptForeColor	The property returns or sets the foreground color of the prompt.
CMSEdit::PromptLocation	The property returns or sets a location of the prompt near the text box.
CMSEdit::PromptWidth	The property returns or sets the width of the prompt.
CMSEdit::ReadOnly	The property returns or sets the value determining that control has the “read only” condition.
CMSEdit::Required	The property returns or sets the value determines that control has required field or not.
CMSEdit::SQLInfo	The internal property, which contains the index to the main database.
CMSEdit::TabStop	The property returns or sets the value indicating whether or not a user can use the TAB key to set the focus to an object.
CMSEdit::Type	The property returns or sets a data type of CMSEditbox.

CMSEdit Properties	Description
CMSEdit::UpperCase	The property returns or sets the value indicating whether or not the output returns the field text as “Upper case” one.
CMSEdit::Validation	The property returns or sets a value that determines the validation criterion for a column - a range of entries available for using in the column.
CMSEdit::Value	The property returns or sets a value that contains contents of a text box.
CMSEdit::Visible	The property returns or sets a value that determines whether the control is visible or hidden.
CMSEdit::Vscroll	The property returns or sets the value indicating that the vertical scrolling is permitted or not.
CMSEdit::WantReturn	The property returns or sets the flag of the “Want return".

Events

CMSEdit Events	Description
CMSEdit::DataChanged	Occurs when the data changed.
CMSEdit::DoLookup	Occurs when lookup was pressed.
CMSEdit::InvalidInput	Occurs when a user enters the incorrect meaning.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

CMSEdit::AddItem

The method initializes the CMSEdit control.

Syntax

```
object.AddItem varItem
varItem
```

Required. This parameter (string) is the text of the add item or value of the property a format for this item.

Return Values

If the (boolean) value is True then the method succeeded. If the value is False then the method failed.

Example

(Visual Basic 5.0)

```
Private Sub ...
' Value mvItem was set before
if CMSEdit1.AddItem(mvItem) Then
MsgBox "Item was removed"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit" on page 1068

CMSEdit::ComboBox

The property returns or sets a value specifying that the control represents ComboBox or ListBox.

Syntax

```
object.ComboBox [ = boolean ]
Boolean
```

A Boolean value determines that an object is represented as ComboBox (True) or as ListBox (Default).

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.ComboBox = True
```

```
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::Name" on page 1095

"CMSEdit::Type" on page 1110

"CMSEdit::Value" on page 1112

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::ControlType

The property returns the type of the control.

Syntax

```
object.ControlType
```

A numeric (long) expression specifying the control type. The value of the property is equal to 2 (the constant is typeCMSEdit).

Access:

Read-Only

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim whatControlType as Long
whatControlType = CMSEdit1.ControlType
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::Name" on page 1095

"CMSEdit::Type" on page 1110

"CMSEdit::Value" on page 1112

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::DataChanged

Occurs when the data is changed.

Syntax

```
Private Sub object_DataChanged()
```

Example

(Visual Basic 5.0)

```
Private Sub CMSEdit1_DataChanged()
MsgBox "Contents was changed"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::DoLookup

Occurs when the lookup button is pressed.

Syntax

```
Private Sub object_DoLookup()
```

Example

(Visual Basic 5.0)

```
Private Sub CMSEdit1_DoLookup()
MsgBox "Lookup button is pressed"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::EditAlignment

The property returns or sets an alignment value of the text box.

Syntax

```
object.EditAlignment [ = long ]
Long
```

This is a numeric value defining an alignment of the text box.

Value	Description
0	Left justify
1	Center
2	Right justify

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.EditAlignment = 1
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::EditBackColor" on page 1077

"CMSEdit::EditBorder" on page 1078

"CMSEdit::EditFont" on page 1079

"CMSEdit::EditForeColor" on page 1080

"CMSEdit::EditStyles" on page 1081

"CMSEdit::Value" on page 1112

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::EditBackColor

The property returns or sets the background color of the text box.

Syntax

```
object.EditBackColor [ = color ]
Color
```

The OLE_COLOR value or constant that determines the background color of the text box.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.EditBackColor = vbButtonText
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::EditAlignment" on page 1076

"CMSEdit::EditBorder" on page 1078

"CMSEdit::EditFont" on page 1079

"CMSEdit::EditForeColor" on page 1080

“CMSEdit::EditStyles” on page 1081

“CMSEdit::Value” on page 1112

“CMSBox” on page 1034

“CMSEdit” on page 1068

CMSEdit::EditBorder

The property returns or sets the border type of the text box.

Syntax

```
object.EditBorder [ = long ]
```

```
Long
```

This is a numeric expression specifying the border type. Can be one of the following values:

Value	Description
0	None (Default)
1	Thin
2	Thick
3	Underline
4	Shadow
5	Raised
6	Recessed
7	Thin Raised
8	Thin Recessed
9	Button Up
10	Button Down
11	Default Button Up
12	Default Button Down
13	Combo Up
14	Combo Down

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.EditBorder = 5
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::EditAlignment" on page 1076

"CMSEdit::EditBackColor" on page 1077

"CMSEdit::EditFont" on page 1079

"CMSEdit::EditForeColor" on page 1080

"CMSEdit::EditStyles" on page 1081

"CMSEdit::Value" on page 1112

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::EditFont

The property returns or sets the font for the text box.

Syntax

```
object.EditFont
StdFont
```

The StdFont value that determines the font for the text box.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.EditFont.Bold = True
```

```
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::EditAlignment" on page 1076

"CMSEdit::EditBackColor" on page 1077

"CMSEdit::EditBorder" on page 1078

"CMSEdit::EditForeColor" on page 1080

"CMSEdit::EditStyles" on page 1081

"CMSEdit::Value" on page 1112

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::EditForeColor

The property returns or sets the foreground color of the text box.

Syntax

```
object.EditForeColor [ = color ]
```

```
Color
```

The OLE_COLOR value or constant that determines the foreground colors of the text box.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.EditForeColor = vbButtonText
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::EditAlignment" on page 1076

"CMSEdit::EditBackColor" on page 1077

"CMSEdit::EditBorder" on page 1078

"CMSEdit::EditFont" on page 1079

"CMSEdit::EditStyles" on page 1081

"CMSEdit::Value" on page 1112

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::EditStyles

Used to return or set values of all properties of the control.

Syntax

```
object.EditStyles [ = long ]
```

```
Long
```

This is a numeric expression. It specifies the border type. Can be combination of the following values:

Constant	Value	Description
Disabled	0x02000000	Returns or sets a value that determines whether a form or control cannot respond to user-generated events
Hide	0x01000000	Returns or sets a value that determines whether a control is hidden
ReadOnly	0x00080000	Returns or sets a value that determines whether a control values have the property "read only"

Constant	Value	Description
Password	0x00000020	Returns or sets a value that determines whether a control is a user for input data in condition “Password”
UpperCase	0x00000008	The text on the field output in condition “Upper case”
LowerCase	0x00000010	The text on the field output in condition “Lower case”
VscrollMultiLine	0x00000080	Permits the vertical scrolling and multi-line condition
Hscroll	0x08000200	Permits the horizontal scrolling
WantReturn	0x00000400	The flag of the “Want return”

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.EditStyles = Hscroll And UpperCase
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

“CMSEdit::EditAlignment” on page 1076

“CMSEdit::EditBackColor” on page 1077

“CMSEdit::EditBorder” on page 1078

“CMSEdit::EditFont” on page 1079

“CMSEdit::EditForeColor” on page 1080

“CMSEdit::Value” on page 1112

“CMSBox” on page 1034

“CMSEdit” on page 1068

CMSEdit::Enabled

The property returns or sets a value that determines whether a control can respond to user-generated events.

Syntax

```
object.Enabled [ = boolean ]
Boolean
```

A Boolean value determines that an object can respond to events (Default) or prevents an object from responding to events (False).

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.Enabled = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

“CMSEdit::HasValidValue” on page 1086

“CMSEdit::Name” on page 1095

“CMSEdit::Value” on page 1112

“CMSBox” on page 1034

“CMSEdit” on page 1068

CMSEdit::ErrorText

The property returns or sets the text of the message with information about the error in input.

Syntax

```
object.ErrorText [ = string ]
String
```

The string expression specifying the message with information about the error in input.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.ErrorText = "Input error"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::HelpID" on page 1086

"CMSEdit::Type" on page 1110

"CMSEdit::Value" on page 1112

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::Format

The property returns or sets the data type for CMSEdit.

Syntax

```
object.Format [ = string ]
String
```

The string expression specifying the data type:

Value	Description
0	Available

Value	Description
1	Document Being Edited
2	Profile Being Edited
3	Checked out
4	Not Available
5	Being Indexed
6	Archived
16	Being Archived
18	Deleted

Remarks:

If the value is more than 6, the text box has the property “Not Available”.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.Format = "3"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

“CMSEdit::Name” on page 1095

“CMSEdit::Type” on page 1110

“CMSEdit::Value” on page 1112

“CMSBox” on page 1034

“CMSEdit” on page 1068

CMSEdit::HasValidValue

The property returns or sets the value indicating that a field is empty or not.

Syntax

```
object.HasValidValue [ = boolean ]
Boolean
```

A Boolean value determines whether the field is required or the field is not required (default). If a value is equal to False, the required field is empty.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim hasValid as Boolean
HasValid = CMSEdit1.HasValidValue
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::Format" on page 1084

"CMSEdit::Type" on page 1110

"CMSEdit::Value" on page 1112

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::HelpID

The property returns or sets the name of the context sensitive help for this button.

Syntax

```
object.HelpID [ = string ]
```

```
String
```

The string expression specifying a name of the context sensitive help for this button.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.HelpID = "txtHelpEdit"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::EditBorder" on page 1078

"CMSEdit::MaxChars" on page 1093

"CMSEdit::Prompt" on page 1097

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::Hscroll

The property returns or sets the value indicating that the horizontal scrolling is permitted or not.

Syntax

```
object.Hscroll [ = boolean ]
```

```
Boolean
```

A Boolean value that indicating that the horizontal scrolling is permitted (True) or not (Default).

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.Hscroll = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::Name" on page 1095

"CMSEdit::Vscroll" on page 1114

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::hWnd

The property returns a handle (from Microsoft Windows) to control's window.

Syntax

```
object.hWnd
```

A numeric (long) value, which contains a handle.

Access:

Read-Only

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim Hndl as Long
Hndl = CMSEdit1.hWnd
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::Name" on page 1095

"CMSEdit::Type" on page 1110

"CMSEdit::Value" on page 1112

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::Init

The method initializes the CMSEdit control.

Syntax

```
object.Init lFlags, formDescription
```

`lFlags`

Required. A long value indicates the mode of the control. The parameter can be:

Constant	Value	Description
frmProfile	0	Profile mode
frmQBE	1	QBE mode
frmReadOnly	2	Read Only
frmDefaultMode	4	Default mode

`formDescription`

Required. A string contains the CMSEdit description.

Return Values

If the numeric (long) value is equal to zero then the method succeeds. If the value is not equal to zero then the method fails.

Example

(Visual Basic 5.0)

```
Private Sub ...
```

```
Dim valueReturn as Long
ValueReturn=CMSEdit1.Init(4, "You can show this control")
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::RemoveItem" on page 1106

"CMSEdit::SetFocus" on page 1108

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::InvalidInput

Occurs when a user enters the incorrect meaning.

Syntax

```
Private Sub object_InvalidInput()
```

Example

(Visual Basic 5.0)

```
Private Sub CMSEdit1_InvalidInput()
MsgBox "You enter the incorrect meaning"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::List

The property returns or sets item specified by index in the list box.

Syntax

```
object.List (index) [ = string ]
```

```
index
```

The numeric expression specifying the index of the list box item.

```
string
```

The string expression specifying the value of the list box item.

Access:

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.List(0) = "Value"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit" on page 1068

CMSEdit::ListIndex

The property returns or sets selection in the list box.

Syntax

```
object.ListIndex [= index ]
```

```
index
```

The numeric expression specifying the index.

Access:

Read-Write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.ListIndex = 0
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit" on page 1068

CMSEdit::LookupName

The property returns or sets the form name for look up.

Syntax

```
object.LookupName [ = string ]
```

```
String
```

The string expression that specifies the form name.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.LookupName = "frmLookUp"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::ErrorText" on page 1083

"CMSEdit::LowerCase" on page 1093

"CMSEdit::MultiLine" on page 1094

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::LowerCase

The property returns or sets the value indicating whether or not the output returns the field text as “Lower case” one.

Syntax

```
object.LowerCase [ = boolean ]
Boolean
```

A Boolean value determines that the output must be shown in condition "Lower Case" (True) or not (Default).

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.LowerCase = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

“CMSEdit::Enabled” on page 1083

“CMSEdit::MaxChars” on page 1093

“CMSEdit::UpperCase” on page 1111

“CMSBox” on page 1034

“CMSEdit” on page 1068

CMSEdit::MaxChars

The property returns or sets the maximal length of the text in a field of input.

Syntax

```
object.MaxChars [ = long ]
Long
```

The numeric expression specifies the maximal length of the text in a field of input.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.MaxChars = 3
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::Format" on page 1084

"CMSEdit::Password" on page 1096

"CMSEdit::Prompt" on page 1097

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::MultiLine

The property returns or sets the value indicating the multi-line condition.

Syntax

```
object.MultiLine [ = boolean ]
Boolean
```

A Boolean value indicates the multi-line condition (True) or not (Default).

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.MultiLine = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::HasValidValue" on page 1086

"CMSEdit::MaxChars" on page 1093

"CMSEdit::Value" on page 1112

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::Name

The property returns the name used in the code to identify this control.

Syntax

```
object.Name
```

The string expression specifying the name of the control.

Access:

Read-Only

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim currentName as String
CurrentName = CMSEdit1.Name
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

“CMSEdit::EditFont” on page 1079

“CMSEdit::Enabled” on page 1083

“CMSEdit::Value” on page 1112

“CMSBox” on page 1034

“CMSEdit” on page 1068

CMSEdit::Password

The property returns or sets the value that determines whether a control is in condition “Password”.

Syntax

```
object.Password [ = boolean ]
Boolean
```

A Boolean value indicates this condition (True) or not (Default).

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.Password = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

“CMSEdit::Name” on page 1095

“CMSEdit::Type” on page 1110

“CMSEdit::Value” on page 1112

“CMSBox” on page 1034

“CMSEdit” on page 1068

CMSEdit::Prompt

The property returns or sets the prompt for the button.

Syntax

```
object.Prompt [ = string ]
String
```

A string value determines the prompt for an object.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim line1 as String
Line1 = CMSEdit1.Prompt
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::PromptAlignment" on page 1098

"CMSEdit::PromptBackColor" on page 1099

"CMSEdit::PromptBorder" on page 1100

"CMSEdit::PromptFont" on page 1101

"CMSEdit::PromptForeColor" on page 1102

"CMSEdit::PromptLocation" on page 1103

"CMSEdit::PromptWidth" on page 1104

"CMSEdit::Value" on page 1112

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::PromptAlignment

The property returns or sets a value of the prompt alignment.

Syntax

```
object.PromptAlignment [ = long ]
Long
```

A numeric value of the prompt alignment:

Value	Description
0	Left justify
1	Center
2	Right justify

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.PromptAlignment = 1
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::Prompt" on page 1097

"CMSEdit::PromptBackColor" on page 1099

"CMSEdit::PromptBorder" on page 1100

"CMSEdit::PromptFont" on page 1101

"CMSEdit::PromptForeColor" on page 1102

"CMSEdit::PromptLocation" on page 1103

"CMSEdit::PromptWidth" on page 1104

“CMSEdit::Value” on page 1112

“CMSBox” on page 1034

“CMSEdit” on page 1068

CMSEdit::PromptBackColor

The property returns or sets the background color of the prompt.

Syntax

```
object.PromptBackColor [ = color ]
Color
```

The OLE_COLOR value or constant that determines the background colors of the prompt.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.PromptBackColor = vbButtonText
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

“CMSEdit::Prompt” on page 1097

“CMSEdit::PromptAlignment” on page 1098

“CMSEdit::PromptBorder” on page 1100

“CMSEdit::PromptFont” on page 1101

“CMSEdit::PromptForeColor” on page 1102

“CMSEdit::PromptLocation” on page 1103

“CMSEdit::PromptWidth” on page 1104

“CMSEdit::Value” on page 1112

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::PromptBorder

The property returns or sets the border type of the prompt.

Syntax

```
object.PromptBorder [ = long ]
```

```
Long
```

This is a numeric expression specifying the border type. Can be one of the following values:

Value	Description
0	None (Default)
1	Thin
2	Thick
3	Underline
4	Shadow
5	Raised
6	Recessed
7	Thin Raised
8	Thin Recessed
9	Button Up
10	Button Down
11	Default Button Up
12	Default Button Down
13	Combo Up
14	Combo Down

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
```

```
CMSEdit1.PromptBorder = 1
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::Prompt" on page 1097

"CMSEdit::PromptAlignment" on page 1098

"CMSEdit::PromptBackColor" on page 1099

"CMSEdit::PromptFont" on page 1101

"CMSEdit::PromptForeColor" on page 1102

"CMSEdit::PromptLocation" on page 1103

"CMSEdit::PromptWidth" on page 1104

"CMSEdit::Value" on page 1112

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::PromptFont

The property returns or sets the text font of the prompt.

Syntax

```
object.PromptFont
StdFont
```

The StdFont value determines the text font of the prompt.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.PromptFont.Bold = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::Prompt" on page 1097

"CMSEdit::PromptAlignment" on page 1098

"CMSEdit::PromptBackColor" on page 1099

"CMSEdit::PromptBorder" on page 1100

"CMSEdit::PromptForeColor" on page 1102

"CMSEdit::PromptLocation" on page 1103

"CMSEdit::PromptWidth" on page 1104

"CMSEdit::Value" on page 1112

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::PromptForeColor

The property returns or sets the foreground color of the prompt.

Syntax

```
object.PromptForeColor [ = color ]
```

Color

The OLE_COLOR value or constant that determines the foreground colors of the prompt.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.PromptForeColor = vbButtonText
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::Prompt" on page 1097

"CMSEdit::PromptAlignment" on page 1098

"CMSEdit::PromptBackColor" on page 1099

"CMSEdit::PromptBorder" on page 1100

"CMSEdit::PromptFont" on page 1101

"CMSEdit::PromptLocation" on page 1103

"CMSEdit::PromptWidth" on page 1104

"CMSEdit::Value" on page 1112

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::PromptLocation

The property returns or sets a location of the prompt near the text box.

Syntax

```
object.PromptLocation [ = long ]
```

```
Long
```

This is a numeric expression. It specifies the location of the prompt of the text box. Can be one of the following values:

Constant	Value	Description
None	0	Without the label
Above	1	The label above the text box
Left	2	The label to the left of the text box

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.PromptLocation = 1
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::Prompt" on page 1097

"CMSEdit::PromptAlignment" on page 1098

"CMSEdit::PromptBackColor" on page 1099

"CMSEdit::PromptBorder" on page 1100

"CMSEdit::PromptFont" on page 1101

"CMSEdit::PromptForeColor" on page 1102

"CMSEdit::PromptWidth" on page 1104

"CMSEdit::Value" on page 1112

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::PromptWidth

The property returns or sets the width of the prompt.

Syntax

```
object.PromptWidth [ = long ]
Long
```

A numeric expression specifies the width of the prompt.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.PromptWidth = 550
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::Prompt" on page 1097

"CMSEdit::PromptAlignment" on page 1098

"CMSEdit::PromptBackColor" on page 1099

"CMSEdit::PromptBorder" on page 1100

"CMSEdit::PromptFont" on page 1101

"CMSEdit::PromptForeColor" on page 1102

"CMSEdit::PromptLocation" on page 1103

"CMSEdit::Value" on page 1112

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::ReadOnly

The property returns or sets the value determines that the control has "read only" condition.

Syntax

```
object.ReadOnly [ = boolean ]
Boolean
```

A Boolean value determines that whether a values control has property "read only" (True).

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.ReadOnly = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::Enabled" on page 1083

"CMSEdit::Validation" on page 1112

"CMSEdit::Visible" on page 1113

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::RemoveItem

The method removes an item from the CMSEdit control.

Syntax

```
object.RemoveItem varItem
```

varItem

Required. This parameter (string) is an identifier of the removed item. The identifier itself depends on description of the form field.

Return Values

If the (Boolean) value is True then the method succeeded. If the value is False then the method failed.

Example

(Visual Basic 5.0)

```
Private Sub ...
' Value mvItem was set before
if CMSEdit1.RemoveItem(mvItem) Then MsgBox "Item was removed"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::Init" on page 1089

"CMSEdit::SetFocus" on page 1108

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::Required

The property returns or sets the value determining that the control has required field or not.

Syntax

```
object.Required [ = boolean ]
Boolean
```

A Boolean value determines that control has required field (True) or not (Default).

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.Required = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::Password" on page 1096

"CMSEdit::Type" on page 1110

“CMSEdit::WantReturn” on page 1115

“CMSBox” on page 1034

“CMSEdit” on page 1068

CMSEdit::SetFocus

The method moves the focus to the specified object.

Syntax

```
object.SetFocus
```

Example

(Visual Basic 5.0)

```
Private Sub Test1()
If MsgBox("Do you want to set focus in the CMSEdit control?", _
vbYesNo) = vbYes Then CMSEdit1.SetFocus
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

“CMSEdit::Init” on page 1089

“CMSEdit::RemoveItem” on page 1106

“CMSBox” on page 1034

“CMSEdit” on page 1068

CMSEdit::SQLInfo

The internal property, which contains the index to the main database.

Syntax

```
object.SQLInfo [ = string ]
String
```

The string expression specifies the index to the main database.

Access:

Read-write

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::Name" on page 1095

"CMSEdit::Type" on page 1110

"CMSEdit::Value" on page 1112

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::TabStop

The property returns or sets the value indicating whether or not a user can use the TAB key to set the focus to an object.

Syntax

```
object.TabStop [ = boolean ]
Boolean
```

If a value is equal to False then bypasses the object when the user is tabbing.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.TabStop = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::Name" on page 1095

"CMSEdit::Type" on page 1110

"CMSEdit::Value" on page 1112

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::Type

The property returns or sets the data type of CMSEdit.

Syntax

```
object.Type [ = long ]
Long
```

A numeric expression specifies the data type of this control. Can be one of the following values:

Constant	Value	Description
String	0	text format
Date	1	date format
Time	2	time format
Real	3	real number
Integer	4	integer number
Currency	5	currency format
CMS Comment	6	CMS comment
Comment	7	comment

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.Type = 2
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::Name" on page 1095

"CMSEdit::Prompt" on page 1097

"CMSEdit::Value" on page 1112

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::UpperCase

The property returns or sets the value indicating whether or not the output returns the field text as "Upper case" one.

Syntax

```
object.UpperCase [ = boolean ]
Boolean
```

A Boolean value determines that the output must be shown in condition "Upper Case" (True) or not (Default).

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.UpperCase = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::LowerCase" on page 1093

"CMSEdit::Value" on page 1112

"CMSEdit::Visible" on page 1113

"CMSBox" on page 1034

“CMSEdit” on page 1068

CMSEdit::Validation

The property returns or sets a value that determines the validation criterion for a column - a range of entries available for using in the column.

Syntax

```
object.Validation [ = string ]
String
```

The string expression specifies the validation range of entries accepted in the column.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.Validation = "Range 0 999"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

“CMSEdit::Enabled” on page 1083

“CMSEdit::Required” on page 1107

“CMSEdit::Value” on page 1112

“CMSBox” on page 1034

“CMSEdit” on page 1068

CMSEdit::Value

The property returns or sets a value with contents of a text box.

Syntax

```
object.Value [ = variant ]
```

```
Variant
```

The expression specifies the contents of a text box.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim valueSelect
ValueSelect = CMSEdit1.Value
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::Name" on page 1095

"CMSEdit::Type" on page 1110

"CMSEdit::Visible" on page 1113

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::Visible

The property returns or sets a value that determines whether the control is visible or hidden.

Syntax

```
object.Visible [ = boolean ]
Boolean
```

A Boolean value determines that the control is visible (Default) or hidden (False).

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.Visible = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::Enabled" on page 1083

"CMSEdit::Type" on page 1110

"CMSEdit::Value" on page 1112

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::Vscroll

The property returns or sets the value indicating the vertical scrolling is permitted or not.

Syntax

```
object.Vscroll [ = boolean ]
Boolean
```

A Boolean value indicates that the vertical scrolling is permitted (True) or not (Default).

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.Vscroll = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::Hscroll" on page 1087

"CMSEdit::Prompt" on page 1097

"CMSEdit::Visible" on page 1113

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSEdit::WantReturn

The property returns or sets the "Want return" flag.

Syntax

```
object.WantReturn [ = boolean ]
Boolean
```

A Boolean value of the "Want return" flag. If its value is True, the flag is set else (Default) the flag is not set.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSEdit1.WantReturn = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSEdit::Name" on page 1095

"CMSEdit::Type" on page 1110

"CMSEdit::Value" on page 1112

"CMSBox" on page 1034

"CMSEdit" on page 1068

CMSForm

CMSForm is the special control, which allows displaying and working with DM Forms. This control uses other CMS controls (CMSPush, CMSEdit, CMSBox, CMSCheckBox, CMSList, CMSText, and CMSRadioGroup) when a form is created. CMSForm works like interpreter of the DM Forms. Except described the control supports also standard properties, methods, and events described in Visual Basic.

Methods

CMSForm Methods	Description
CMSForm::GetControl	Gets the CMS Control by its name.
CMSForm::GetValue	Gets a value of the CMS control by its name.
CMSForm::Init	The method initializes CMSForm.
CMSForm::SetValue	Sets a new value to the CMS control.

Properties

CMSForm Properties	Description
CMSForm::BackColor	The property returns or sets the background color of the control.
CMSForm::ClientHeight	The property returns or sets the height of the form.
CMSForm::ClientWidth	The property returns or sets the width of the form.
CMSForm::Controls	The property returns a reference to a collection of Control (CMS Controls) objects.
CMSForm::FormStyles	The property returns or sets styles of the form (whether there is a heading, disabled and etc.)
CMSForm::Prompt	The property returns or sets the caption of the form.

Events

CMSForm Events	Description
CMSForm::ButtonPressed	Occurs when the button (the CMSPush control) is pressed.
CMSForm::CMSGotFocus	Occurs when the CMS control receives the focus.
CMSForm::CMSLostFocus	Occurs when the CMS control loses the focus.
CMSForm::DataChanged	Occurs when a value of the CMS control changes the data.

CMSForm Events	Description
CMSForm::DoLookup	Occurs when the CMSEdit control lookup is opened.
CMSForm::DoubleClick	Occurs when the user double clicks on the CMS control.
CMSForm::InvalidInput	Occurs when the CMSEdit control raises an InvalidInput event. For example, incorrect value was typed in the field with the lookup button.
CMSForm::LastRowRetrieved	Occurs when the CMSList control has raised the FireLastRowRetrieved event.
CMSForm::RowRetrieved	Occurs when some control of CMSList has caused the FireRowRetrieved event.
CMSForm::SelectionChanged	Occurs when CMSList control has raised the SelectionChanged event.
CMSForm::ViewModeChanged	Occurs when CMSList control has raised ViewModeChanged event.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CMSForm.ocx"

CMSForm::BackColor

The property returns or sets the background color of the control.

Syntax

```
object.BackColor [ = color ]
```

```
Color
```

The OLE_COLOR value or constant that determines the background color of an object.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
```

```
CMSForm1.BackColor = vbButtonText
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CMSForm.ocx"

See Also

"CMSForm::ClientHeight" on page 1120

"CMSForm::Controls" on page 1123

"CMSForm::Prompt" on page 1132

"CMSForm" on page 1116

"CMSCheckBox" on page 1049

CMSForm::ButtonPressed

Occurs when the button (the CMSPush control) is pressed.

Syntax

```
Private Sub object_ButtonPressed(ControlName)
```

ControlName

Name (string) of a CMS Control, which has caused event.

Example

(Visual Basic 5.0)

```
Private Sub CMSForm1_ButtonPressed(NameControl as String)
MsgBox "You click control " + NameControl
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CMSForm.ocx"

See Also

"CMSForm::CMSGotFocus" on page 1121

"CMSForm::DataChanged" on page 1124

“CMSForm::InvalidInput” on page 1131

“CMSForm” on page 1116

“CMSCheckBox” on page 1049

CMSForm::ClientHeight

The property returns or sets the height of the form.

Syntax

```
object.ClientHeight [ = single ]
Single
```

A numeric expression specifying the height of the form.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSForm1.ClientHeight = 1200
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CMSForm.ocx"

See Also

“CMSForm::ClientWidth” on page 1120

“CMSForm::Controls” on page 1123

“CMSForm::Prompt” on page 1132

“CMSForm” on page 1116

“CMSCheckBox” on page 1049

CMSForm::ClientWidth

The property returns or sets the width of the form.

Syntax

```
object.ClientWidth [ = single ]
```

```
Single
```

A numeric expression specifying the width of the form.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSForm1.ClientWidth = 750
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CMSForm.ocx"

See Also

"CMSForm::ClientHeight" on page 1120

"CMSForm::FormStyles" on page 1126

"CMSForm::Prompt" on page 1132

"CMSForm" on page 1116

"CMSCheckBox" on page 1049

CMSForm::CMSGotFocus

Occurs when the CMS control receives the focus.

Syntax

```
Private Sub object_CMSGotFocus(ControlName)
ControlName
```

A name (string) of the CMS Control, which has caused event.

Example

(Visual Basic 5.0)

```
Private Sub CMSForm1_CMSGotFocus(NameControl as String)
MsgBox "Control " + NameControl + " got focus"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CMSForm.ocx"

See Also

"CMSForm::CMSLostFocus" on page 1122

"CMSForm::DoubleClick" on page 1125

"CMSForm::InvalidInput" on page 1131

"CMSForm" on page 1116

"CMSCheckBox" on page 1049

CMSForm::CMSLostFocus

Occurs when the CMS control loses the focus.

Syntax

```
Private Sub object_CMSLostFocus(ControlName)
ControlName
```

A name (string) of the CMS Control, which has caused event.

Example

(Visual Basic 5.0)

```
Private Sub CMSForm1_CMSLostFocus(NameControl as String)
MsgBox "Control " + NameControl + " lost focus"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CMSForm.ocx"

See Also

"CMSForm::CMSGotFocus" on page 1121

"CMSForm::DataChanged" on page 1124

"CMSForm::RowRetrieved" on page 1133

"CMSForm" on page 1116

“CMSCheckBox” on page 1049

CMSForm::Controls

The property returns a reference to a collection of Control (CMS Controls) objects.

Syntax

```
object.Controls
```

This property contains the control's collection.

Access:

Read-Only

Remarks:

You can manipulate Control objects using the reference returned by the Controls property.

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim objPush as CMSPush
. . .
Set objPush = CMSForm1.Controls(i)
' Parameter "i" has the Integer type
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CMSForm.ocx"

See Also

“CMSForm::ClientWidth” on page 1120

“CMSForm::FormStyles” on page 1126

“CMSForm::Prompt” on page 1132

“CMSForm” on page 1116

“CMSCheckBox” on page 1049

CMSForm::DataChanged

Occurs when a value of the CMS control changes the data.

Syntax

```
Private Sub object_DataChanged(ControlName)
ControlName
```

A name (string) of the CMS Control, which has caused the event.

Example

(Visual Basic 5.0)

```
Private Sub CMSForm1_DataChanged(NameControl as String)
MsgBox "Control " + NameControl + " lost focus"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CMSForm.ocx"

See Also

"CMSForm::CMSGotFocus" on page 1121

"CMSForm::DoLookup" on page 1124

"CMSForm::InvalidInput" on page 1131

"CMSForm" on page 1116

"CMSCheckBox" on page 1049

CMSForm::DoLookup

Occurs when the CMSEdit control lookup is opened.

Syntax

```
Private Sub object_DoLookup(ControlName)
ControlName
```

A name (string) of the CMS Control, which has caused the event.

Example

(Visual Basic 5.0)

```
Private Sub CMSForm1_DoLookup(NameControl as String)
MsgBox "Control " + NameControl + " had caused DoLookup event"
```

```
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CMSForm.ocx"

See Also

"CMSForm::CMSLostFocus" on page 1122

"CMSForm::DoubleClick" on page 1125

"CMSForm::SelectionChanged" on page 1134

"CMSForm" on page 1116

"CMSCheckBox" on page 1049

CMSForm::DoubleClick

Occurs when the user double clicks on CMS control.

Syntax

```
Private Sub object_DoubleClick(ControlName)
```

`ControlName`

A name (string) of the CMS Control, which has caused the event.

Example

(Visual Basic 5.0)

```
Private Sub CMSForm1_DoubleClick(NameControl as String)
MsgBox "You click " + NameControl + " control"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CMSForm.ocx"

See Also

"CMSForm::CMSGotFocus" on page 1121

"CMSForm::DataChanged" on page 1124

"CMSForm::InvalidInput" on page 1131

"CMSForm" on page 1116

"CMSCheckBox" on page 1049

CMSForm::FormStyles

The property returns or sets styles of the form (whether there is a heading, disabled and etc.)

Syntax

```
object.FormStyles [ = Integer ]
```

`Integer`

This is a numeric expression. It specifies the styles of the form. Can be combination of the following values:

Constant	Value	Description
CAPTION	0x00000001	Determines whether a form has a caption
MINIMIZEBOX	0x00000008	Determines whether a form has a minimize button
SYSMENU	0x00000004	Determines whether a form has a system menu
VSCROLL	0x00000040	Determines whether a form has a vertical scroll bar
HSCROLL	0x00000080	Determines whether a form has a horizontal scroll bar
TABORDER	0x00000100	Sets the tab order of most objects within their parent form from left to the right
VISIBLE	0x01000000	Determines whether a form is visible
DISABLED	0x02000000	Determines whether a form is disabled
MINIMIZE	0x00000400	Determines whether a form is minimized

Constant	Value	Description
CHILD	0x00000010	Determines whether a form has children
GRADIENT	0x00000020	Determines whether a form sets the background color from BackColor to Gradient Fill
BORDER	0x00000000	Determines whether a form has a border
SIZEBOX	0x00000002	Determines whether a form has a Size box
DLGFRAME	0x00000800	Determines whether a form is created with a double border but no title

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSForm1.FormStyles = 12
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CMSForm.ocx"

See Also

“CMSForm::BackColor” on page 1118

“CMSForm::ClientWidth” on page 1120

“CMSForm::Prompt” on page 1132

“CMSForm” on page 1116

“CMSCheckBox” on page 1049

CMSForm::GetControl

Gets the CMS Control by its name.

Syntax

```
object.GetControl ControlName
```

`ControlName`

A name (string) of the Control. It can be one of the CMS controls of the following types:

ControlName
CMSBox
CMSCheckBox
CMSEdit
CMSList
CMSPush
CMSRadioGroup
CMSText

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim objPush as CMSPush
. . .
Set objPush = CMSForm1.GetControl("CMSPush1")
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CMSForm.ocx"

See Also

"CMSForm::GetValue" on page 1129

"CMSForm::Init" on page 1129

"CMSForm::SetValue" on page 1134

"CMSForm" on page 1116

“CMSCheckBox” on page 1049

CMSForm::GetValue

Gets a value of the CMS control by its name.

Syntax

```
object.GetValue ControlName
```

`ControlName`

A name (string) of the Control.

Return Values

The object type.

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim objPush as CMSPush
. . .
Set objPush = CMSForm1.GetValue("CMSPush1")
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CMSForm.ocx"

See Also

“CMSForm::GetControl” on page 1128

“CMSForm::Init” on page 1129

“CMSForm::SetValue” on page 1134

“CMSForm” on page 1116

“CMSCheckBox” on page 1049

CMSForm::Init

The method initializes CMSForm.

Syntax

```
object.Init lFlags, formDescription
```

`lFlags`

Required. A long value indicates the mode of the control. The parameter can be:

Constant	Value	Description
frmProfile	0	Profile mode
frmQBE	1	QBE mode
frmReadOnly	2	Read Only
frmDefaultMode	4	Default mode. Values of CMSForm have three states: checked, unchecked, and undefined.

`formDescription`

Required. A string contains the CMSForm description.

Return Values

If the numeric (long) value is equal to zero then the method succeeds. If the value is not equal to zero then the method fails.

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim valueReturn as Long
ValueReturn=CMSForm1.Init(4, "String1")
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CMSForm.ocx"

See Also

"CMSForm::GetControl" on page 1128

"CMSForm::GetValue" on page 1129

"CMSForm::SetValue" on page 1134

"CMSForm" on page 1116

"CMSCheckBox" on page 1049

CMSForm::InvalidInput

Occurs when the CMSEdit control raises an InvalidInput event. For example, incorrect value was typed in the field with the lookup button.

Syntax

```
Private Sub object_InvalidInput(ControlName)
```

`ControlName`

A name (string) of the CMS Control, which has caused the event.

Example

(Visual Basic 5.0)

```
Private Sub CMSForm1_InvalidInput(NameControl as String)
MsgBox NameControl + " had InvalidInput event"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CMSForm.ocx"

See Also

"CMSForm::CMSGotFocus" on page 1121

"CMSForm::DoLookup" on page 1124

"CMSForm::RowRetrieved" on page 1133

"CMSForm" on page 1116

"CMSCheckBox" on page 1049

CMSForm::LastRowRetrieved

Occurs when the CMSList control has raised the FireLastRowRetrieved event.

Syntax

```
Private Sub object_LastRowRetrieved(sReason, nTotalLines)
```

`sReason`

Required. The enumeration (integer) was finished normally (equal to zero) or is interrupted (equal to 1).

`nTotalLines`

Required. The total number (long) of the received lines.

Example

(Visual Basic 5.0)

```
Private Sub CMSForm1_LastRowRetrieved(Reason, TotalLines)
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CMSForm.ocx"

See Also

"CMSForm::ButtonPressed" on page 1119

"CMSForm::DoubleClick" on page 1125

"CMSForm::RowRetrieved" on page 1133

"CMSForm" on page 1116

"CMSCheckBox" on page 1049

CMSForm::Prompt

The property returns or sets the caption of the form.

Syntax

```
object.Prompt [ = string ]
String
```

A string value determines a caption of the form.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim line1 as String
. . .
Line1 = CMSForm1.Prompt
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CMSForm.ocx"

See Also

"CMSForm::ClientHeight" on page 1120

"CMSForm::Controls" on page 1123

"CMSForm::FormStyles" on page 1126

"CMSForm" on page 1116

"CMSCheckBox" on page 1049

CMSForm::RowRetrieved

Occurs when some control of CMSList has caused the FireRowRetrieved event.

Syntax

```
Private Sub object_RowRetrieved(nCurrentLine)
```

`nCurrentLine`

This is a number (long) of the received line.

Example

(Visual Basic 5.0)

```
Private Sub CMSForm1_RowRetriaved(CurrentLine)
MsgBox "Prosess was stopped. The received line " + _
CStr(CurrentLine)
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CMSForm.ocx"

See Also

"CMSForm::CMSGotFocus" on page 1121

"CMSForm::DataChanged" on page 1124

"CMSForm::InvalidInput" on page 1131

“CMSForm” on page 1116

“CMSCheckBox” on page 1049

CMSForm::SelectionChanged

Occurs when the CMSList control has raised the SelectionChanged event.

Syntax

```
Private Sub object_SelectionChanged(ControlName)
```

ControlName

A name (string) of the CMS Control, which has caused the event.

Example

(Visual Basic 5.0)

```
Private Sub CMSForm1_SelectionChanged(NameControl as String)
MsgBox NameControl + " had SelectionChanged event"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CMSForm.ocx"

See Also

“CMSForm::CMSLostFocus” on page 1122

“CMSForm::DoubleClick” on page 1125

“CMSForm::RowRetrieved” on page 1133

“CMSForm” on page 1116

“CMSCheckBox” on page 1049

CMSForm::SetValue

Sets a new value to the CMS control.

Syntax

```
object.SetValue ControlName, newValue
```

ControlName

Control's name (string).

`newValue`

A new value (unknown) for the CMS control.

Example

(Visual Basic 5.0)

```
Private Sub ...
. . .
CMSForm1.SetValue("CMSPush1", 1)
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CMSForm.ocx"

See Also

"CMSForm::GetControl" on page 1128

"CMSForm::GetValue" on page 1129

"CMSForm::Init" on page 1129

"CMSForm" on page 1116

"CMSCheckBox" on page 1049

CMSForm::ViewModeChanged

Occurs when the CMSList control has raised the ViewModeChanged event.

Syntax

```
Private Sub object_ViewModeChanged(ControlName)
```

`ControlName`

A name (string) of the CMS Control, which has caused the event.

Example

(Visual Basic 5.0)

```
Private Sub CMSForm1_ViewModeChanged(NameControl as String)
MsgBox NameControl + " had ViewModeChanged event"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CMSForm.ocx"

See Also

"CMSForm" on page 1116

"CMSList::ViewModeChanged" on page 1152

CMSList

The CMSList control displays a list of items from which the user can select one item or more. Except described the control supports also standard properties, methods, and events described in Visual Basic.

Methods

CMSList Methods	Description
CMSList::Init	The method initializes CMSList.
CMSList::Refresh	The method refreshes the content control.
CMSList::SetFocus	The method moves the focus to the specified object.
CMSList::SetSearch	The method sets the search.

Properties

CMSList Properties	Description
CMSList::Border	The property returns or sets a border type of the list box.
CMSList::ControlType	The property returns or sets the type of the control.
CMSList::Enabled	The property returns or sets a value that determines whether a control can respond to user-generated events.
CMSList::MaxSelectionSize	The property returns or sets a value that determines whether a user can make multiple selections in a control.
CMSList::Name	The property returns the name used in code to identify this control.
CMSList::PreviewCommand	The property returns or sets whether the commands should be previewed .
CMSList::Selection	The property returns the obtained current selection.
CMSList::SID	The property returns the SID (session identifier) used in this control.
CMSList::TabStop	The property returns or sets the value indicating whether or not a user can use the TAB key to set the focus to an object.

CMSList Properties	Description
CMSList::Visible	The property returns or sets a value that determines whether the control is visible or hidden.

Events

CMSList Events	Description
CMSList::DblClick	Occurs when the user double clicks on control.
CMSList::FireLastRowRetrieved	Occurs when retrieved the last row.
CMSList::FireRowRetrieved	Occurs when retrieved any row.
CMSList::SelectionChanged	Occurs when the selection has been changed.
CMSList::ViewModeChanged	Occurs when the view mode has been changed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

CMSList::Border

The property returns or sets a border type of the list box.

Syntax

```
object.Border [ = long ]
```

```
Long
```

A numeric expression specifying the border type. Can be one of the following values:

Value	Description
0	None (Default)
1	Thin
2	Thick
3	Underline
4	Shadow

Value	Description
5	Raised
6	Recessed
7	Thin Raised
8	Thin Recessed
9	Button Up
10	Button Down
11	Default Button Up
12	Default Button Down
13	Combo Up
14	Combo Down

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSList1.Border = 10
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSList::MaxSelectionSize" on page 1145

"CMSList::Name" on page 1145

"CMSList::Visible" on page 1152

"CMSForm" on page 1116

"CMSList" on page 1136

CMSList::ControlType

The property returns the type of the control.

Syntax

```
object.ControlType
```

A numeric (long) expression specifying the control type. The value of the property is equal to 3 (the constant is typeCMSList).

Access:

Read-Only

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim whatControlType as Long
whatControlType = CMSList1.ControlType
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSList::MaxSelectionSize" on page 1145

"CMSList::Name" on page 1145

"CMSList::Visible" on page 1152

"CMSForm" on page 1116

"CMSList" on page 1136

CMSList::DblClick

Occurs when the user double clicks on the control.

Syntax

```
Private Sub object_DblClick()
```

Object is the CMSList control.

Example

(Visual Basic 5.0)

```
Private Sub CMSList1_DblClick()
MsgBox "You double-click in CMSList control"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSList::FireLastRowRetrieved" on page 1142

"CMSList::FireRowRetrieved" on page 1143

"CMSList::SelectionChanged" on page 1148

"CMSForm" on page 1116

"CMSList" on page 1136

CMSList::Enabled

The property returns or sets a value that determines whether a control can respond to user-generated events.

Syntax

```
object.Enabled [ = boolean ]
boolean
```

A boolean value determines that object can respond to events (Default) or prevents object from respond to events (False).

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSList1.Enabled = Fals
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSList" on page 1136

CMSList::FireLastRowRetrieved

Occurs when the last row is retrieved.

Syntax

```
Private Sub object_FireLastRowRetrieved(sReason, nTotalLines)
```

`sReason`

Required. The enumeration (integer) was finished normally (equal to zero) or is interrupted (equal to 1) - was suspended by a user.

`nTotalLines`

Required. The total number (long) of the received lines.

Example

(Visual Basic 5.0)

```
Private Sub CMSList1_FireLastRowRetrieved(Reason, TotalLines)
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSList::DblClick" on page 1140

"CMSList::FireRowRetrieved" on page 1143

"CMSList::SelectionChanged" on page 1148

"CMSForm" on page 1116

"CMSList" on page 1136

CMSList::FireRowRetrieved

Occurs when any row is retrieved.

Syntax

```
Private Sub object_FireRowRetrieved(nCurrentLine)
```

`nCurrentLine`

Required. This is a number (long) of the received line or as value of second parameter in the FireLastRowRetrieved method if there was an event, which has cancelled this event.

Example

(Visual Basic 5.0)

```
Private Sub CMSList1_FireRowRetriaved(CurrentLine)
MsgBox "Process was stopped. The received line " + _
CStr(CurrentLine)
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSList::DblClick" on page 1140

"CMSList::FireLastRowRetrieved" on page 1142

"CMSList::SelectionChanged" on page 1148

"CMSForm" on page 1116

"CMSList" on page 1136

CMSList::Init

The method initializes CMSList control.

Syntax

```
object.Init lFlags, formDescription
```

`lFlags`

Required. A long value indicates the mode of the control. The parameter can be:

Constant	Value	Description
frmProfile	0	Profile mode
frmQBE	1	QBE mode
frmReadOnly	2	Read Only
frmDefaultMode	4	Default mode

`formDescription`

Required. A string contains the CMSList description.

Return Values

If the numeric (long) value is equal to zero then the method succeeds, else the method fails.

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim valueReturn as Long
ValueReturn=CMSList1.Init(4, "You can show this control")
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSList::Refresh" on page 1147

"CMSList::SetFocus" on page 1149

"CMSList::SetSearch" on page 1149

"CMSForm" on page 1116

"CMSList" on page 1136

CMSList::MaxSelectionSize

The property returns or sets a value that determines whether a user can make multiple selections in a control.

Syntax

```
object.MaxSelectionSize
```

A numeric (long) expression specifies the setting multiple selections (1) or single selection (0).

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim whatControlType as Long
CMSList1.MaxSelectionSize = 1
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSList::Border" on page 1138

"CMSList::ControlType" on page 1140

"CMSList::Selection" on page 1147

"CMSForm" on page 1116

"CMSList" on page 1136

CMSList::Name

The property returns the name used in the code to identify this control.

Syntax

```
object.Name
```

The string expression specifies the name of the control.

Access:

Read-Only

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim currentName as String
CurrentName = CMSList1.Name
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSList::Border" on page 1138

"CMSList::Selection" on page 1147

"CMSList::TabStop" on page 1151

"CMSForm" on page 1116

"CMSList" on page 1136

CMSList::PreviewCommand

The property returns or sets whether the commands should be previewed .

Syntax

```
object.PreviewCommand
```

The boolean expression specifies whether the commands should be previewed.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSList1.PreviewCommand= True
```

```
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSList" on page 1136

CMSList::Refresh

The method refreshes the content control.

Syntax

```
object.Refresh
```

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSList1.Refresh
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSList::Init" on page 1143

"CMSList::SetFocus" on page 1149

"CMSList::SetSearch" on page 1149

"CMSForm" on page 1116

"CMSList" on page 1136

CMSList::Selection

The property returns the obtained current selection.

Syntax

```
object.Selection
```

The unknown expression specifies the obtained current selection.

Access:

Read-Only

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim currentSelectionName
CurrentSelectionName = CMSList1.Selection
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSList::MaxSelectionSize" on page 1145

"CMSList::Name" on page 1145

"CMSList::Visible" on page 1152

"CMSForm" on page 1116

"CMSList" on page 1136

CMSList::SelectionChanged

Occurs when the selection has been changed.

Syntax

```
Private Sub object_SelectionChanged()
```

Example

(Visual Basic 5.0)

```
Private Sub CMSList1_SelectionChanged()
MsgBox "The selection has been changed"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSList::DblClick" on page 1140

"CMSList::FireLastRowRetrieved" on page 1142

"CMSList::FireRowRetrieved" on page 1143

"CMSForm" on page 1116

"CMSList" on page 1136

CMSList::SetFocus

The method moves the focus to the specified object.

Syntax

```
object.SetFocus
```

Example

(Visual Basic 5.0)

```
Private Sub Test1()
If MsgBox("Do you want to set focus in the CMSList control?", _
vbYesNo) = vbYes Then CMSList1.SetFocus
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSList::Init" on page 1143

"CMSList::Refresh" on page 1147

"CMSList::SetSearch" on page 1149

"CMSForm" on page 1116

"CMSList" on page 1136

CMSList::SetSearch

The method sets the search.

Syntax

```
object.SetSearch pSearch
```

`pSearch`

Required. The object type.

Return Values

If the numeric (long) value is equal to zero then the method succeeded, else the method failed.

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim valueReturn as Long
Dim objSearch as Document
If CMSList1.SetSearch(objSearch)= 0 then Exit Sub
. . .
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

“CMSList::Init” on page 1143

“CMSList::Refresh” on page 1147

“CMSList::SetFocus” on page 1149

“CMSForm” on page 1116

“CMSList” on page 1136

CMSList::SID

The property returns the SID (session identifier) used in this control.

Syntax

```
object.SID
```

The string expression specifies the SID of the control.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSList1.SID = SID
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSList" on page 1136

CMSList::TabStop

The property returns or sets the value indicating whether or not a user can use the TAB key to set the focus to an object.

Syntax

```
object.TabStop [ = boolean ]
Boolean
```

If value is equal to False then bypasses the object when the user is tabbing.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSList1.TabStop = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSList::MaxSelectionSize" on page 1145

"CMSList::Name" on page 1145

"CMSList::Visible" on page 1152

"CMSForm" on page 1116

"CMSList" on page 1136

CMSList::ViewModeChanged

Occurs when the view mode has been changed.

Syntax

```
Private Sub object_ViewModeChanged()
```

Example

(Visual Basic 5.0)

```
Private Sub CMSList1_ViewModeChanged()
MsgBox "The view mode has been changed"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSForm" on page 1116

"CMSList" on page 1136

CMSList::Visible

The property returns or sets a value that determines whether the control is visible or hidden.

Syntax

```
object.Visible [ = boolean ]
Boolean
```

A Boolean value determines that the control is visible (Default) or hidden (False).

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSList1.Visible = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSList::ControlType" on page 1140

"CMSList::Selection" on page 1147

"CMSList::TabStop" on page 1151

"CMSForm" on page 1116

"CMSList" on page 1136

CMSPush

The Control has the CommandButton type. Except described the control supports also standard properties, methods, and events described in Visual Basic.

Methods

CMSPush Methods	Description
CMSPush::Init	The method initializes the CMSPush control.
CMSPush::SetFocus	The method moves the focus to the specified object.

Properties

CMSPush Properties	Description
CMSPush::BitmapAt	The property returns or sets a position of the bitmap on a push button.
CMSPush::ControlType	The property returns or sets the type of the control.
CMSPush::DownBitmap	The property returns or sets the path to the icon for the pressed button.
CMSPush::Enabled	The property returns or sets a value that determines whether a control can respond to user-generated events.
CMSPush::Font	The property returns or sets a Font object.
CMSPush::HelpID	The property returns or sets the name of the context sensitive help for this button.
CMSPush::IconSizeBitmap	The property returns or sets a value that determines whether a bitmap is compressed or stretched to the size of an icon.
CMSPush::Name	The property returns the name used in the code to identify this control.
CMSPush::Prompt	The property returns or sets the prompt for the button.
CMSPush::TabStop	The property returns or sets the value indicating whether or not a user can use the TAB key to set the focus to an object.

CMSPush Properties	Description
CMSPush::TextAt	This property returns and sets a position of the text on the CMSPush control.
CMSPush::UpBitmap	The property returns or sets the path to the bitmap for the unpressed button.
CMSPush::Visible	The property returns or sets a value that determines whether the control is visible or hidden.

Events

CMSPush Events	Description
CMSPush::ButtonPressed	Occurs when the button is pressed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

CMSPush::BitmapAt

The property returns or sets a position of the bitmap on a push button.

Syntax

```
object.BitmapAt [ = long ]
```

`Long`

A numeric expression specifies the position of this control. Can be one of the following values:

Value	Description
0	Upper left
1	Middle left
2	Lower left
3	Upper middle
4	Centered
5	Lower middle
6	Upper right

Value	Description
7	Middle right
8	Lower right

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSPush1.BitmapAt = 4
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSPush::Enabled" on page 1158

"CMSPush::Prompt" on page 1164

"CMSPush::TextAt" on page 1166

"CMSList" on page 1136

"CMSPush" on page 1153

CMSPush::ButtonPressed

Occurs when the button is pressed.

Syntax

```
Private Sub object_ButtonPressed()
```

Object is the CMSPush control.

Example

(Visual Basic 5.0)

```
Private Sub CMSPush1_ButtonPressed()
MsgBox "You click in CMSPush control"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSPush::Init" on page 1162

"CMSPush::SetFocus" on page 1164

"CMSList" on page 1136

"CMSPush" on page 1153

CMSPush::ControlType

The property returns the type of the control.

Syntax

```
object.ControlType
```

A numeric (long) expression specifies the control type. The value of the property is equal to 4 (the constant is typeCMSPush).

Access:

Read-Only

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim whatControlType as Long
whatControlType = CMSPush1.ControlType
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSPush::DownBitmap" on page 1158

"CMSPush::Name" on page 1163

“CMSPush::TabStop” on page 1165

“CMSList” on page 1136

“CMSPush” on page 1153

CMSPush::DownBitmap

The property returns or sets the path to the icon for the pressed button.

Syntax

```
object.DownBitmap [ = string ]
```

```
String
```

The string expression specifies a path to the icon or bitmap for the pressed button.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSPush1.DownBitmap = "c:\project\images\iconDown.gif"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

“CMSPush::Font” on page 1159

“CMSPush::IconSizeBitmap” on page 1161

“CMSPush::UpBitmap” on page 1167

“CMSList” on page 1136

“CMSPush” on page 1153

CMSPush::Enabled

The property returns or sets a value that determines whether a control can respond to user-generated events.

Syntax

```
object.Enabled [ = boolean ]
Boolean
```

A Boolean value determines that an object can respond to events (Default) or prevents an object from responding to events (False).

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSPush1.Enabled = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSPush::HelpID" on page 1160

"CMSPush::Name" on page 1163

"CMSPush::TabStop" on page 1165

"CMSList" on page 1136

"CMSPush" on page 1153

CMSPush::Font

The property returns or sets a Font object.

Syntax

```
object.Font
StdFont
```

The StdFont value that determines a Font object.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSPush1.Font.Bold = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSPush::Enabled" on page 1158

"CMSPush::Prompt" on page 1164

"CMSPush::TextAt" on page 1166

"CMSList" on page 1136

"CMSPush" on page 1153

CMSPush::HelpID

The property returns or sets the name of the context sensitive help for this button.

Syntax

```
object.HelpID [ = string ]
String
```

The string expression specifies the name of the context sensitive help for this button.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSPush1.HelpID = "txtHelpPush"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSPush::Enabled" on page 1158

"CMSPush::Prompt" on page 1164

"CMSPush::TextAt" on page 1166

"CMSList" on page 1136

"CMSPush" on page 1153

CMSPush::IconSizeBitmap

The property returns or sets a value that determines whether a bitmap is compressed or stretched to the size of an icon.

Syntax

```
object.IconSizeBitmap [ = boolean ]
Boolean
```

A Boolean value, which determines that graphics will be resized to fit the size of an icon (True) or not (Default).

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSPush1.IconSizeBitmap = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSPush::BitmapAt" on page 1155

"CMSPush::UpBitmap" on page 1167

"CMSPush::Visible" on page 1168

"CMSList" on page 1136

"CMSPush" on page 1153

CMSPush::Init

The method initializes the CMSPush control.

Syntax

```
object.Init lFlags, formDescription
```

`lFlags`

Required. A long value indicates the mode of the control. The parameter can be:

Constant	Value	Description
frmProfile	0	Profile mode
frmQBE	1	QBE mode
frmReadOnly	2	Read Only
frmDefaultMode	4	Default mode

`formDescription`

Required. A string contains the CMSPush description.

Return Values

If the numeric (long) value is equal to zero then the method succeeds. If the value is not equal to zero then the method fails.

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim valueReturn as Long
ValueReturn=CMSPush1.Init(4, "You can show this control")
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSPush::ButtonPressed" on page 1156

"CMSPush::SetFocus" on page 1164

"CMSList" on page 1136

"CMSPush" on page 1153

CMSPush::Name

The property returns the name used in the code to identify this control.

Syntax

```
object.Name
```

The string expression specifies the name of the control.

Access:

Read-Only

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim currentName as String
CurrentName = CMSPush1.Name
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSPush::ControlType" on page 1157

"CMSPush::Prompt" on page 1164

"CMSPush::TextAt" on page 1166

"CMSList" on page 1136

"CMSPush" on page 1153

CMSPush::Prompt

The property returns or sets the prompt for the button.

Syntax

```
object.Prompt [ = string ]
String
```

A string value determines a prompt for the object.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim line1 as String
Line1 = CMSPush1.Prompt
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSPush::BitmapAt" on page 1155

"CMSPush::Name" on page 1163

"CMSPush::Visible" on page 1168

"CMSList" on page 1136

"CMSPush" on page 1153

CMSPush::SetFocus

The method moves the focus to the specified object.

Syntax

```
object.SetFocus
```

Example

(Visual Basic 5.0)

```
Private Sub Test1()
If MsgBox("Do you want to set focus in the CMSPush control?", _
vbYesNo) = vbYes Then CMSPush1.SetFocus
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSPush::ButtonPressed" on page 1156

"CMSPush::Init" on page 1162

"CMSList" on page 1136

"CMSPush" on page 1153

CMSPush::TabStop

The property returns or sets the value indicating whether or not a user can use the TAB key to set the focus to an object.

Syntax

```
object.TabStop [ = boolean ]
Boolean
```

If a value is equal to False then bypasses the object when the user is tabbing.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSPush1.TabStop = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSPush::Enabled" on page 1158

"CMSPush::Prompt" on page 1164

"CMSPush::TextAt" on page 1166

"CMSList" on page 1136

"CMSPush" on page 1153

CMSPush::TextAt

This property returns and sets a position of the text on the CMSPush control.

Syntax

```
object.TextAt [ = long ]
```

```
Long
```

A numeric expression specifies the position of this text for this control. Can be one of the following values:

Value	Description
0	Upper left
1	Middle left
2	Lower left
3	Upper middle
4	Centered
5	Lower middle
6	Upper right
7	Middle right
8	Lower right

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
```

```
CMSPush1.TextAt = 5
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSPush::Font" on page 1159

"CMSPush::IconSizeBitmap" on page 1161

"CMSPush::UpBitmap" on page 1167

"CMSList" on page 1136

"CMSPush" on page 1153

CMSPush::UpBitmap

The property returns or sets the path to the bitmap for the unpressed button.

Syntax

```
object.UpBitmap [ = string ]
String
```

The string expression specifies a path to the bitmap for the unpressed button.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSPush1.UpBitmap = "c:\project\images\iconUp.gif"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSPush::DownBitmap" on page 1158

"CMSPush::Prompt" on page 1164

"CMSPush::TextAt" on page 1166

"CMSList" on page 1136

"CMSPush" on page 1153

CMSPush::Visible

The property returns or sets a value that determines whether the control is visible or hidden.

Syntax

```
object.Visible [ = boolean ]
Boolean
```

A Boolean value determines that the control is visible (Default) or hidden (False).

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSPush1.Visible = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSPush::Enabled" on page 1158

"CMSPush::Prompt" on page 1164

"CMSPush::TextAt" on page 1166

"CMSList" on page 1136

"CMSPush" on page 1153

CMSRadioGroup

The CMSRadioGroup control displays a list of items from which the user can select one item or more. Except described the control supports also standard properties, methods, and events described in Visual Basic.

Methods

CMSRadioGroup Methods	Description
CMSRadioGroup::Init	The method initializes the CMSRadioGroup control.
CMSRadioGroup::SetFocus	The method moves the focus to the specified object.

Properties

CMSRadioGroup Properties	Description
CMSRadioGroup::BackColor	The property returns or sets the background color of the control prompt.
CMSRadioGroup::Border	The property returns or sets a border type of the box.
CMSRadioGroup::ControlType	The property returns or sets the type of the control.
CMSRadioGroup::Enabled	The property returns or sets a value that determines whether a control can respond to user-generated events.
CMSRadioGroup::Font	The property returns or sets a Font object.
CMSRadioGroup::ForeColor	The property returns or sets the foreground color used to display text and graphics in an object.
CMSRadioGroup::Items	The property returns or sets the line description of the radio button. Default three buttons are set.
CMSRadioGroup::Name	The property returns the name used in the code to identify this control.
CMSRadioGroup::Prompt	The property returns or sets the prompt for the box.
CMSRadioGroup::SQLInfo	The internal property, which contains the index to the main database.

CMSRadioGroup Properties	Description
CMSRadioGroup::TabStop	The property returns or sets the value indicating whether or not a user can use the TAB key to set the focus to an object.
CMSRadioGroup::Value	The property returns/sets the value of the Control.
CMSRadioGroup::Visible	The property returns or sets a value that determines whether the control is visible or hidden.

Events

CMSRadioGroup Events	Description
CMSRadioGroup::DataChanged	Occurs when the data is changed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

CMSRadioGroup::BackColor

The property returns or sets the background color of the control prompt.

Syntax

```
object.BackColor [ = color ]
```

```
Color
```

The OLE_COLOR value or constant that determines the background colors of a control.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSRadioGroup1.BackColor = vbButtonText
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSRadioGroup::ControlType" on page 1173

"CMSRadioGroup::Items" on page 1179

"CMSRadioGroup::Visible" on page 1183

"CMSPush" on page 1153

"CMSRadioGroup" on page 1169

CMSRadioGroup::Border

The property returns or sets the border type of a box.

Syntax

```
object.Border [ = long ]
```

```
Long
```

A numeric expression specifies the border type of this control. Can be one of the following values:

Value	Description
0	None (Default)
1	Thin
2	Thick
3	Underline
4	Shadow
5	Raised
6	Recessed
7	Thin Raised
8	Thin Recessed
9	Button Up
10	Button Down
11	Default Button Up

Value	Description
12	Default Button Down
13	Combo Up
14	Combo Down

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSRadioGroup1.Border = 10
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSRadioGroup::ControlType" on page 1173

"CMSRadioGroup::Items" on page 1179

"CMSRadioGroup::Visible" on page 1183

"CMSPush" on page 1153

"CMSRadioGroup" on page 1169

CMSRadioGroup::ControlType

The property returns the type of the control.

Syntax

```
object.ControlType
```

A numeric (long) expression specifying the control type. The value of the property is equal to 5 (the constant is typeCMSRadioGroup).

Access:

Read-Only

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim whatControlType as Long
whatControlType = CMSRadioGroup1.ControlType
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSRadioGroup::Enabled" on page 1175

"CMSRadioGroup::ForeColor" on page 1176

"CMSRadioGroup::Prompt" on page 1180

"CMSPush" on page 1153

"CMSRadioGroup" on page 1169

CMSRadioGroup::DataChanged

Occurs when the data is changed.

Syntax

```
Private Sub object_DataChanged()
```

Example

(Visual Basic 5.0)

```
Private Sub CMSRadioGroup1_DataChanged()
MsgBox "Contents was changed"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSRadioGroup::Init" on page 1177

“CMSPush” on page 1153

“CMSRadioGroup” on page 1169

CMSRadioGroup::Enabled

The property returns or sets a value that determines whether a control can respond to user-generated events.

Syntax

```
object.Enabled [ = boolean ]
Boolean
```

A Boolean value determines that an object can respond to events (Default) or prevents an object from responding to events (False).

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSRadioGroup1.Enabled = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

“CMSRadioGroup::ControlType” on page 1173

“CMSRadioGroup::Items” on page 1179

“CMSRadioGroup::Visible” on page 1183

“CMSPush” on page 1153

“CMSRadioGroup” on page 1169

CMSRadioGroup::Font

The property returns or sets a Font object.

Syntax

```
object.Font
StdFont
```

The StdFont value that determines a Font object.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSRadioGroup1.Font.Bold = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSRadioGroup::Border" on page 1172

"CMSRadioGroup::TabStop" on page 1182

"CMSRadioGroup::Visible" on page 1183

"CMSPush" on page 1153

"CMSRadioGroup" on page 1169

CMSRadioGroup::ForeColor

The property returns or sets the foreground color used to display text and graphics in an object.

Syntax

```
object.ForeColor [ = color ]
Color
```

The OLE_COLOR value or constant that determines the foreground colors of an object.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSRadioGroup1.ForeColor = vbHighLight
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSRadioGroup::ControlType" on page 1173

"CMSRadioGroup::Items" on page 1179

"CMSRadioGroup::Visible" on page 1183

"CMSPush" on page 1153

"CMSRadioGroup" on page 1169

CMSRadioGroup::Init

The method initializes the CMSRadioGroup control.

Syntax

```
object.Init lFlags, formDescription
```

`lFlags`

Required. A long value indicates the mode of the control. The parameter can be:

Constant	Value	Description
frmProfile	0	Profile mode
frmQBE	1	QBE mode
frmReadOnly	2	Read Only
frmDefaultMode	4	Default mode

`formDescription`

Required. A string contains the CMSRadioGroup description.

Return Values

If the numeric (long) value is equal to zero then the method succeeds, else the method fails.

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim valueReturn as Long
ValueReturn=CMSRadioGroup1.Init(4,"You can show this control")
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSRadioGroup::SetFocus" on page 1180

"CMSPush" on page 1153

"CMSRadioGroup" on page 1169

CMSRadioGroup::Name

The property returns the name used in the code to identify an object.

Syntax

```
object.Name
```

The string expression specifies the name of the control.

Access:

Read-Only

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim currentName as String
CurrentName = CMSRadioGroup1.Name
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSRadioGroup::Enabled" on page 1175

"CMSRadioGroup::Prompt" on page 1180

"CMSCheckBox" on page 1049

"CMSRadioGroup" on page 1169

CMSRadioGroup::Items

The property returns or sets the line description of the radio button. Default three buttons are set.

Syntax

```
object.Items
```

The string expression specifies the line description of the Radio button of the control.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSRadioGroup1.Item = "Second"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSRadioGroup::BackColor" on page 1171

"CMSRadioGroup::Border" on page 1172

“CMSRadioGroup::Font” on page 1175

“CMSPush” on page 1153

“CMSRadioGroup” on page 1169

CMSRadioGroup::Prompt

The property returns or sets the prompt for the box.

Syntax

```
object.Prompt [ = string ]
String
```

A string value determines the prompt for an object.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim line1 as String
Line1 = CMSRadioGroup1.Prompt
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

“CMSRadioGroup::ControlType” on page 1173

“CMSRadioGroup::Items” on page 1179

“CMSRadioGroup::Visible” on page 1183

“CMSPush” on page 1153

“CMSRadioGroup” on page 1169

CMSRadioGroup::SetFocus

The method moves the focus to the specified object.

Syntax

```
object.SetFocus
```

Example

(Visual Basic 5.0)

```
Private Sub Test1()
If MsgBox("Do you want to set focus in CMSRadioGroup control?" _
, vbYesNo) = vbYes Then CMSRadioGroup1.SetFocus
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSRadioGroup::Init" on page 1177

"CMSPush" on page 1153

"CMSRadioGroup" on page 1169

CMSRadioGroup::SQLInfo

The internal property, which contains the index to the main database.

Syntax

```
object.SQLInfo [ = string ]
String
```

The string expression specifies the index to the main database.

Access:

Read-write

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSRadioGroup::ControlType" on page 1173

"CMSRadioGroup::Items" on page 1179

"CMSRadioGroup::Visible" on page 1183

"CMSPush" on page 1153

"CMSRadioGroup" on page 1169

CMSRadioGroup::TabStop

The property returns or sets the value indicating whether or not a user can use the TAB key to set the focus to an object.

Syntax

```
object.TabStop [ = boolean ]
Boolean
```

If a value is equal to False then bypasses the object when the user is tabbing.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSRadioGroup1.TabStop = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSRadioGroup::Enabled" on page 1175

"CMSRadioGroup::Font" on page 1175

"CMSRadioGroup::Prompt" on page 1180

"CMSPush" on page 1153

"CMSRadioGroup" on page 1169

CMSRadioGroup::Value

The property returns/sets the value of the Control.

Syntax

```
object.Value [ = variant ]
variant
```

The value of the control

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim mvarValue
mvarValue = CMSRadioGroup1.ValueEnd Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSRadioGroup" on page 1169

CMSRadioGroup::Visible

The property returns or sets a value that determines whether the control is visible or hidden.

Syntax

```
object.Visible [ = boolean ]
Boolean
```

A Boolean value determines that the control is visible (Default) or hidden (False).

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSRadioGroup1.Visible = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSRadioGroup::ForeColor" on page 1176

"CMSRadioGroup::Prompt" on page 1180

"CMSRadioGroup::TabStop" on page 1182

"CMSPush" on page 1153

"CMSRadioGroup" on page 1169

CMSText

The CMSText control can be of the Label type, allows multiple selections and can have any border. Except described the control supports also standard properties, methods, and events described in Visual Basic.

Methods

CMSText Methods	Description
CMSText::Init	The method initializes CMSText.
CMSText::SetFocus	The method moves the focus to the specified object.

Properties

CMSText Properties	Description
CMSText::Alignment	The property returns or sets the text alignment in the place of the control.
CMSText::BackColor	The property returns or sets the background color of the control prompt.
CMSText::Border	The property returns or sets draw border.
CMSText::ControlType	The property returns the type of the control.
CMSText::Enabled	The property returns or sets a value that determines whether a control can respond to user-generated events.
CMSText::Font	The property returns or sets a Font control.
CMSText::ForeColor	The property returns or sets the foreground color used to display text and graphics in a control.
CMSText::Name	The property returns the name used in the code to identify this control.
CMSText::Opaque	The property returns or sets a value that determines whether the control's BackColor will fill the control and obscure all the colors or graphics behind it.
CMSText::TabStop	The property returns or sets the value indicating whether or not a user can use the TAB key to set the focus to a control.

CMSText Properties	Description
CMSText::Text	The property returns or sets the text contained in the control.
CMSText::Visible	The property returns or sets a value that determines whether the control is visible or hidden.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

CMSText::Alignment

The property returns or sets the text alignment in the place of the control.

Syntax

```
object.Alignment
Long
```

A numeric (long) expression specifies the text alignment in the place. Can be one of the following values:

Value	Description
0	Aligns left edges
1	Aligns in the center
2	Aligns right edges

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSText1.Alignment = 1
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSText::Font" on page 1190

"CMSText::ForeColor" on page 1191

"CMSText::Text" on page 1196

"CMSText" on page 1184

"CMSRadioGroup" on page 1169

CMSText::BackColor

The property returns or sets the background color of the control prompt.

Syntax

```
object.BackColor [ = color ]
Color
```

The OLE_COLOR value or constant that determines the background colors of a control.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSText1.BackColor = vbButtonText
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSText::Font" on page 1190

"CMSText::ForeColor" on page 1191

"CMSText::Text" on page 1196

"CMSText" on page 1184

"CMSRadioGroup" on page 1169

CMSText::Border

The property returns or sets draw border.

Syntax

```
object.Border [ = long ]
```

```
Long
```

A numeric expression specifying the border type. Can be one of the following values:

Value	Description
0	None (Default)
1	Thin
2	Thick
3	Underline
4	Shadow
5	Raised
6	Recessed
7	Thin Raised
8	Thin Recessed
9	Button Up
10	Button Down
11	Default Button Up
12	Default Button Down
13	Combo Up
14	Combo Down

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSText1.Border= 9
```

```
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSText" on page 1184

CMSText::ControlType

The property returns the type of the control.

Syntax

```
object.ControlType
```

A numeric (long) expression specifies the control type. The value of the property is equal to 6 (the constant is typeCMSText).

Access:

Read-Only

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim whatControlType as Long
whatControlType = CMSText1.ControlType
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSText::Enabled" on page 1190

"CMSText::Opaque" on page 1194

"CMSText" on page 1184

"CMSRadioGroup" on page 1169

CMSText::Enabled

The property returns or sets a value that determines whether a control can respond to user-generated events.

Syntax

```
object.Enabled [ = boolean ]
Boolean
```

A Boolean value determines that an object can respond to events (Default) or prevents an object from responding to events (False).

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSText1.Enabled = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSText::Alignment" on page 1186

"CMSText::Name" on page 1193

"CMSText::Text" on page 1196

"CMSText" on page 1184

"CMSRadioGroup" on page 1169

CMSText::Font

The property returns or sets a Font object.

Syntax

```
object.Font
StdFont
```

The StdFont value that determines a Font object.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSText1.Font.Bold = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSText::Alignment" on page 1186

"CMSText::ForeColor" on page 1191

"CMSText::Opaque" on page 1194

"CMSText" on page 1184

"CMSRadioGroup" on page 1169

CMSText::ForeColor

The property returns or sets the foreground color used to display text and graphics in a control.

Syntax

```
object.ForeColor [ = color ]
```

```
Color
```

The OLE_COLOR value or constant that determines the foreground colors of a control.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSText1.ForeColor = vbHighLight
```

```
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSText::BackColor" on page 1187

"CMSText::Font" on page 1190

"CMSText::Text" on page 1196

"CMSText" on page 1184

"CMSRadioGroup" on page 1169

CMSText::Init

The method initializes the CMSText control.

Syntax

```
object.Init lFlags, formDescription
```

`lFlags`

Required. A long value indicates the mode of the control. The parameter can be:

Constant	Value	Description
frmProfile	0	Profile mode
frmQBE	1	QBE mode
frmReadOnly	2	Read Only
frmDefaultMode	4	Default mode

`formDescription`

Required. A string contains the CMSText description.

Return Values

If the numeric (long) value is equal to zero then the method succeeds. If the value is not equal to zero then the method fails.

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim valueReturn as Long
ValueReturn=CMSText1.Init(4, "You can show this control")
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSText::SetFocus" on page 1195

"CMSText" on page 1184

"CMSRadioGroup" on page 1169

CMSText::Name

The property returns the name used in the code to identify this control.

Syntax

```
object.Name [ = string ]
```

A string expression specifies the name of the control.

Access:

Read-Only

Example

(Visual Basic 5.0)

```
Private Sub ...
Dim currentName as String
CurrentName = CMSText1.Name
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSText::Enabled" on page 1190

“CMSText::ForeColor” on page 1191

“CMSText::Text” on page 1196

“CMSText” on page 1184

“CMSRadioGroup” on page 1169

CMSText::Opaque

The property returns or sets a value that determines whether the control's BackColor will fill the control and obscure all the colors or graphics behind it.

Syntax

```
object.Opaque [ = boolean ]
Boolean
```

A Boolean value determines that the object is transmittance (True) else the opaque (Default).

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSText1.Opaque = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

“CMSText::BackColor” on page 1187

“CMSText::Font” on page 1190

“CMSText::Text” on page 1196

“CMSText” on page 1184

“CMSRadioGroup” on page 1169

CMSText::SetFocus

The method moves the focus to the specified object.

Syntax

```
object.SetFocus
```

Example

(Visual Basic 5.0)

```
Private Sub Test1()
If MsgBox("Do you want to set focus in the CMSText control?", _
vbYesNo) = vbYes Then CMSText1.SetFocus
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSText::Init" on page 1192

"CMSText" on page 1184

"CMSRadioGroup" on page 1169

CMSText::TabStop

The property returns or sets the value indicating whether or not a user can use the TAB key to set the focus to a control.

Syntax

```
object.TabStop [ = boolean ]
Boolean
```

If a value is equal to False then bypasses the object when the user is tabbing.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
```

```
CMSText1.TabStop = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSText::Font" on page 1190

"CMSText::Name" on page 1193

"CMSText::Opaque" on page 1194

"CMSText" on page 1184

"CMSRadioGroup" on page 1169

CMSText::Text

The property returns or sets the text contained in the control.

Syntax

```
object.Text [ = String ]
```

The string expression specifies the text.

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
CMSText1.Text = "This control"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSText::ControlType" on page 1189

"CMSText::Name" on page 1193

"CMSText::TabStop" on page 1195

"CMSText" on page 1184

"CMSRadioGroup" on page 1169

CMSText::Visible

The property returns or sets a value that determines whether the control is visible or hidden.

Syntax

```
object.Visible [ = boolean ]
```

```
Boolean
```

A boolean value determines that the control is visible (Default) or hidden (False).

Access:

Read-write

Example

(Visual Basic 5.0)

```
Private Sub ...
```

```
CMSText1.Visible= False
```

```
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "Cmsctrls.ocx"

See Also

"CMSText" on page 1184

Chapter

11

Add-ons

In This Chapter

- "About Add-ons" on page 1200
- "Using Add-Ons" on page 1200
- "Creation of new Add-on" on page 1202
- "Add-On Objects" on page 1206

About Add-ons

Add-ons is a way of providing users with additional functionality without making changes to the Explorer's code. There is a lot of uses for the add-ons pane. For example:

- Document preview
- Profile form preview
- Versions preview
- Log viewer (mostly for troubleshooting reasons)

Add-ons are not intended to dramatically change the Explorer's UI. They provide an easy way to enhance it. If users need a different functionality they are free to create their own ActiveX controls on top of the Explorer Core and use these controls on the Explorer instead of standard ones.

Note that Explorer Add-ons extend the user's interface functionality as opposed to the plugs in, which extend the overall object model.

See Also

"Viewer OCX Control" on page 1221

"Profile View Add-on" on page 1218

"Versions Add-on" on page 1240

"Logger Add-On" on page 1243

"Add-ons" on page 1199

Using Add-Ons

- "Add-on Container Control" on page 1201
- "Creation of new Add-on" on page 1202
- "Registration of new Add-on" on page 1204

Add-on Container Control

Add-ons allow the developer to extend the user interface by providing additional information about objects selected by the user. The add-on container control provides the ability to use them in any user interface in the same manner.

Examples of the add-ons include the quick viewer and profile preview add-ons shipped with the DM. They supply users with the document preview and profile preview functionality right from the DM Desktop and Explorer, without requiring them to perform additional operations.

In its simplest form any ActiveX control can be used as an add-on. Developers just have to register their controls properly to turn them into an add-on. However, in most cases you need to provide information depending on the user's activity. For example, the profile preview add-on should display the profile data for the currently selected document, i.e. each time the user changes the selection the profile information should be updated in response to that change. For that reason the add-on container control (which loads and hosts all registered add-ons) notifies an add-on every time when selection changes.

Explorer add-ons are standard ActiveX controls, loaded and managed by the Adds Container control. Several extensions can be loaded simultaneously - a user can easily switch between them using small tabs at the bottom of the container control. Each tab has extension name on it for easy user access.

Explorer by default includes Add-on Container as an optional pane under its Contents pane.

Since Add-on Container is a separate ActiveX control, it can also be used without Explorer.

Except described the control supports also standard properties, methods, and events described in Visual Basic.

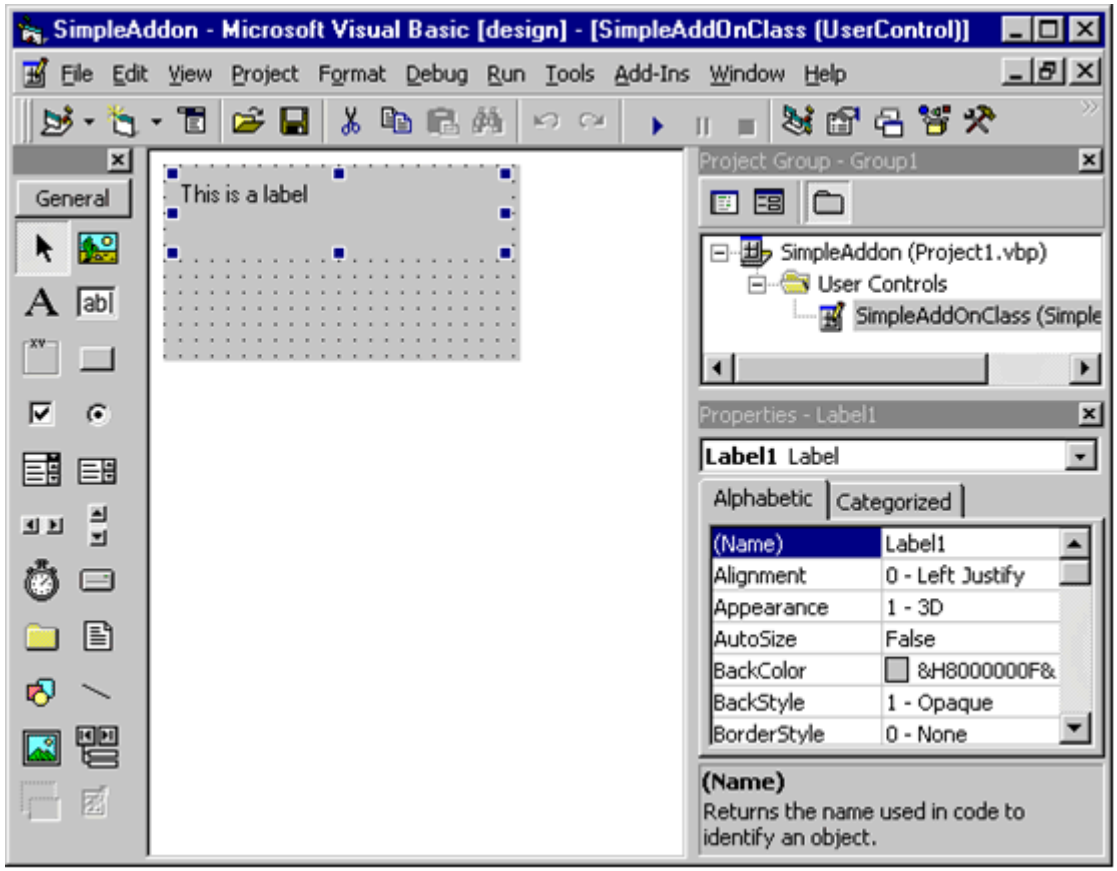
1. Create an ActiveX control, which will expose the desired functionality and become a server for AddOnCnt container control.
2. If you want to provide different information depending on the selection, implement the SetSelection method.
3. If you need to be notified about parent object changes, implement the SetParent method.
4. If you want to provide visual information to the user about events, happening to your add-on you can fire the SetStatus event in different places in your code.
5. Register your add-on as described in "Registration of new Add-on" on page 1204.

Example

As an example consider the following add-on, created in VB5. It displays a number of objects, selected by the user.

1. Create a new ActiveX Control project in VB5.

2 Add the Label control to the form as shown below.



3 Write a SetSelection handler code as shown below.

```
' Declares event that will be raised by the code to draw user's
attention
Event ChangeStatus(b As Boolean)


' SetSelecion handler
' As an argument you are getting an array of objects, selected by
the user
Public Sub SetSelection(ByVal oa As BCObjectArray)


    ' Display the size of the array (the number of objects
selected)
    Label1.Caption = "You have selected " & oa.Size & " items"
```

```
    ' If there is more than five objects, let user know about that
    If oa.Size >= 5 Then
        ' by fireing appropriate event
        ' TRUE means that we need to draw user's attention to our
page
        RaiseEvent ChangeStatus(True)
    End If
End Sub
' end of addon code_
```

4 Now proceed to the Registration of new Add-on section to register your add-on.

See Also

"AddOnCnt Control" on page 1207

Registration of new Add-on

Before Hummingbird 5.0 version

1 Under HKEY_LOCAL_MACHINE\Software\Hummingbird\PowerDOCS\Explorer\AddOns create a key which will hold all the information about your add-on.

2 Under that new key you can specify the following information:

Key	Description
@(Default)	(required) ProgID or CLSID of your ActiveX control
Description	(optional) Text to appear on your add-on's tab
WorksWith	(optional) Ids of plug-ins, your add-on will work with. If no plug-ins are present your add-on will not be loaded
InsertAt	(optional) letter from A to Z which indicates the position of the add-on, A - first, B - second and so on

Key	Description
Disable	(optional) 1 for temporary disabling this add-on (add-on will not be loaded)
ObjectsSupported	(optional) Classes of objects, your add-on will work with. If object with not supported class will be selected in left (hierarchy) pane your add-on will be temporary hidden

Example

1. Under HKEY_LOCAL_MACHINE\Software\Hummingbird\PowerDOCS\Explorer\AddOns create a "SimpleAddon" subkey.

2. Under that key create the following values:

Key	Value	Comments
@(Default)	"SimpleAddon.SimpleAddonClass"	ProgID of your add-on
Description	"Simple"	(optional) Text to appear on your add-on's tab

Hummingbird 5.0 or later version

IBCRegistrarExt::RegisterExtObject method should be used to register new add-on. See Remarks section in this method for possible values of additional parameters.

Example

This example registers "SimpleAddon".

```
Dim pCtx As New BCCoreCtx

Dim pRegistrar As New BCRegistrar

pCtx.RegisterVariantParam "Description", "Simple"

pCtx.RegisterVariantParam "InsertAt", "B"

pRegistrar.RegisterExtObject rcRegister, "AddOn", "SimpleAddon",
"SimpleAddon.SimpleAddonClass", pCtx
```

Add-On Objects

- "AddOnCnt Control" on page 1207
- "Profile View Add-on" on page 1218
- "Viewer OCX Control" on page 1221
- "Versions Add-on" on page 1240
- "Logger Add-On" on page 1243
- "LoggerOCX Control" on page 1243
- "LoggerOCXNotificationSink Control" on page 1244

AddOnCnt Control

Properties

Name	Description
AddOnCnt::AddOnCount	Used by the Add-on container's owner to find out how many Add-ons were loaded by the Init() method (should be called only after Init()).
AddOnCnt::SelectedAddon	Returns or sets selected add-on.
AddOnCnt::Font	The property returns or sets a Font for the Tabs text.
AddOnCnt::hWnd	The property returns a handle of the object's window.
AddOnCnt::Enabled	The property returns or sets enabled state for add-on.
AddOnCnt::Caption	The property returns or sets the text displayed in a control title bar.
AddOnCnt::AddonName	The property returns name (registry key) for specified add-on. This name can be useful when information about add-ons is being saved.
AddOnCnt::AddonDescription	The property returns/sets add-on description that appears on the Tab for specified add-on. This name can be useful when information about add-ons is being saved.
AddOnCnt::AddonVisible	The property returns/sets add-on visible state for specified add-on.
AddOnCnt::Enabled	The property returns add-on enabled state for specified add-on.

Methods

Name	Description
AddOnCnt::Init	The method loads all the Add-ons for specifies plug-ins using the string pluginString as a reference list of all the loaded plug-ins. pluginString is a semicolon-separated string containing the names for all the plug-ins registered in the system. If it is empty add-on container will load all add-ons. Standard DM plug-ins are "DOCSOPEN;ATTACHE".
AddOnCnt::SetParent	Should be called by container owner when parent object in the tree of the Explorer was changed. Container will notify all loaded Add-ons about new parent object by calling SetParent method of the Add-ons.
AddOnCnt::SetSelection	Called by Add-on container owner when selection in the List control was changed. Add-on container will notify all loaded Add-ons about selection change by calling SetSelection method of the Add-ons.

Events

Name	Description
AddOnCnt::ActiveAddonChanged	This event is called when selected add-on was changed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

"About Add-ons" on page 1200

"Add-ons" on page 1199

AddOnCnt::ActiveAddonChanged

This event is called when selected add-on is changed.

Example

(Visual Basic 6.0)

```
Private Sub AddOnCnt1_ActiveAddonChanged(ByVal nActiveAddon As
Long)
...
End Sub
```

See Also

"AddOnCnt Control" on page 1207

"About Add-ons" on page 1200

"Add-ons" on page 1199

AddOnCnt::AddOnCount

It is used by the Add-on container's owner to find out how many Add-ons are loaded by the Init() method (should be called only after Init()).

Syntax

```
object.AddOnCount
```

A numeric (long) expression specifying the count of the loaded Tabs.

Access

Read-Only

Example

(Visual Basic 6.0)

```
Private Sub ...
whatCount = AddOn1.AddonCount
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

"AddOnCnt Control" on page 1207

“About Add-ons” on page 1200

“Add-ons” on page 1199

AddOnCnt::AddonDescription

The property returns/sets add-on description that appears on the Tab for specified add-on. This description can be useful when information about add-ons is being saved.

Syntax

```
object.AddonDescription (nAddonNumber)
```

A string value (String) - add-on name.

```
nAddonNumber
```

Add-on order (0 ... AddOnCount - 1)

Access

Read-Write

Example

(Visual Basic 6.0)

```
Private Sub ...
Dim addonDescription As String
AddonDescription = AddOn1.AddonDescription(0)
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

“AddOnCnt Control” on page 1207

“About Add-ons” on page 1200

“Add-ons” on page 1199

AddOnCnt::AddonEnabled

The property returns add-on enabled state for specified add-on.

Syntax

```
object.AddonEnabled(nAddonNumber)
```

A Boolean value (Boolean) - add-on enabled state (add-on can be disabled if it does not support the current parent object).

`nAddonNumber`

Add-on order (0 ... AddOnCount - 1)

Access

Read-Only

Example

(Visual Basic 6.0)

```
Private Sub ...
AddOn1.AddonVisible(0) = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

"AddOnCnt Control" on page 1207

"About Add-ons" on page 1200

"Add-ons" on page 1199

AddOnCnt::AddonName

The property returns a name (registry key) for specified add-on. This name can be useful when information about add-ons is being saved.

Syntax

```
object.AddonName(nAddonNumber)
```

A string value (String) - add-on name.

`nAddonNumber`

Add-on order (0 ... AddOnCount - 1)

Access

Read-Only

Example

(Visual Basic 6.0)

```
Private Sub ...
Dim addonName As String
AddonName = AddOn1.AddonName(0)
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

"AddOnCnt Control" on page 1207

"About Add-ons" on page 1200

"Add-ons" on page 1199

AddOnCnt::AddonVisible

The property returns/sets add-on visible state for specified add-on.

Syntax

```
object.AddonVisible(nAddonNumber)
```

A Boolean value (Boolean) - add-on visible state.

```
nAddonNumber
```

Add-on order (0 ... AddOnCount - 1)

Access

Read-Write

Example

(Visual Basic 6.0)

```
Private Sub ...
AddOn1.AddonVisible(0) = False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

“AddOnCnt Control” on page 1207

“About Add-ons” on page 1200

“Add-ons” on page 1199

AddOnCnt::Caption

The property returns or sets the text displayed in a control title bar.

Syntax

```
object.Caption [=  string]
String
```

A string value determines a text for the title bar.

Access

Read-Write

Example

(Visual Basic 6.0)

```
Private Sub ...
AddOn1.Caption = "Caption"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

“AddOnCnt Control” on page 1207

“About Add-ons” on page 1200

“Add-ons” on page 1199

AddOnCnt::Enabled

The property returns or sets the enabled state for add-ons.

Syntax

```
object.Enabled
```

A Boolean value (Boolean) that determines the state Tabs.

Access

Read-Write

Example

(Visual Basic 6.0)

```
Private Sub ...
AddOn1.Enabled = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

"AddOnCnt Control" on page 1207

"About Add-ons" on page 1200

"Add-ons" on page 1199

AddOnCnt::Font

The property returns or sets a Font for the Tabs text.

Syntax

```
object.Font
```

A Visual Basic StdFont object that determines a Font object. It is interpreted via the IFontDisp interface in Visual C++.

Access

Read-Write

Example

(Visual Basic 6.0)

```
Private Sub ...
AddOn1.Font.Bold = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

"AddOnCnt Control" on page 1207

"About Add-ons" on page 1200

"Add-ons" on page 1199

AddOnCnt::hWnd

The property returns a handle of the object's window.

Syntax

```
object.hWnd
```

A numeric (long)value contents handle.

Access

Read-Only

Example

(Visual Basic 6.0)

```
Private Sub ...
Dim Hndl as Long
Hndl = AddOn1.hWndEnd Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

"AddOnCnt Control" on page 1207

"About Add-ons" on page 1200

"Add-ons" on page 1199

AddOnCnt::Init

The method loads all the Add-ons for specifies plug-ins using the string pluginString as a reference list of all the loaded plug-ins. pluginString is a semicolon-separated string containing the names for all the plug-ins registered in the system. If it is empty add-on container will load all add-ons.

Standard DM plug-ins are "DOCSOPEN;ATTACHE".

Syntax

```
object.Init pluginString
```

`pluginString`

Required. Semicolon-separated string containing the names for all the plug-ins registered in the system.

Return Values

Numeric (long) value - the count of loaded add-ons.

Example

(Visual Basic 6.0)

```
Private Sub ...
Dim valuePlugins as Long
ValuePlugins=AddOn1.Init("DOCSOPEN")
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

“AddOnCnt Control” on page 1207

“About Add-ons” on page 1200

“Add-ons” on page 1199

AddOnCnt::SelectedAddon

Returns or sets selected add-on.

Syntax

```
object.SelectedAddon
```

A numeric (long) value specifying the selected add-on.

Access

Read-Write

Example

(Visual Basic 6.0)

```
Private Sub ...
whatSelected = AddOn1.SelectedAddon
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

"AddOnCnt Control" on page 1207

"About Add-ons" on page 1200

"Add-ons" on page 1199

AddOnCnt::SetParent

Should be called by a container owner when a parent object in the tree of the Explorer was changed. Container will notify all loaded Add-ons about new parent object by calling the SetParent method of the Add-ons.

Syntax

```
object.SetParent Unknown
```

`Unknown`

It is a IBCObject.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

"AddOnCnt Control" on page 1207

"About Add-ons" on page 1200

"Add-ons" on page 1199

AddOnCnt::SetSelection

Called by Add-on container owner when a selection in the List control was changed. Add-on container will notify all loaded Add-ons about a selection change by calling the SetSelection method of the Add-ons.

Syntax

```
object.SetSelection ObjectArray
```

`ObjectArray`

It is a BCObjectArray.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

"AddOnCnt Control" on page 1207

"About Add-ons" on page 1200

"Add-ons" on page 1199

Profile View Add-on

This add-on previews a document profile.

Except described the control supports also standard properties, methods, and events described in Visual Basic.

This add-on is implemented via the ProfView control:

Properties

Name	Description
ProfView::Font	The property returns or sets a Font object.
ProfView::Caption	The property returns or sets the text displayed in a control title bar.

Methods

Name	Description
ProfView::SetSelection	This method used by Add-on container control, which notifies all loaded Add-ons that selection was changed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

"Add-on Container Control" on page 1201

"Viewer OCX Control" on page 1221

"Versions Add-on" on page 1240

"Logger Add-On" on page 1243

"Add-ons" on page 1199

ProfView::Caption

The property returns or sets the text displayed in the control title bar.

Syntax

```
object.Caption
```

A string value that determines a text for the title bar.

Access

Read-Write

Example

(Visual Basic 6.0)

```
Private Sub ...
ProFile1.Caption = "Caption"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

"Profile View Add-on" on page 1218

"Add-On Objects" on page 1206

AddOnCnt interface

Add-ons

ProfView::Font

The property returns or sets a Font object.

Syntax

```
object.Font
```

A Visual Basic StdFont object that determines a Font object. It is interpreted via the IFontDisp interface in Visual C++.

Access

Read-Write

Example

(Visual Basic 6.0)

```
Private Sub ...
ProFile1.Font.Bold = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

Profile form preview

Add-on Container

AddOnCnt interface

Add-ons

ProfView::SetSelection

This method is used by Add-on container control, which notifies all loaded Add-ons that a selection has been changed.

Syntax

```
object.SetSelection ObjectArray
```

`ObjectArray`

It is a IBCObjectArray.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

Profile form preview

Add-on Container

Add-on container control

Add-ons

Viewer OCX Control

The control is intended for a choice of the documents viewing.

Except described the control supports also standard properties, methods, and events described in Visual Basic.

This add-on is implemented via the FulViewOCX control:

Properties

Name	Description
FulViewOCX::Font	The property returns or sets a Font object.
FulViewOCX::Caption	The property returns or sets the text displayed in the control title bar.
FulViewOCX::BackColor	The property returns or sets a background color.
FulViewOCX::ForeColor	The property returns or sets a foreground color.

Methods

Name	Description
FulViewOCX::CopyAll	The method copies a document text to the clipboard.
FulViewOCX::CopySelection	The method copies the selected document text to the clipboard.
FulViewOCX::PrintCurrentDoc	The method prints the selected document.
FulViewOCX::PrintSetup	The method sets margins, paper source, paper size, paper orientation, and other layout options for the active document/its version.
FulViewOCX::Search	The method returns the Boolean expression as result of the search.
FulViewOCX::SearchDialog	The method sets criteria for search: string expression sought, expression that sets the starting position for each search, and specifies the type of a string comparison.
FulViewOCX::SearchNext	The method continues searching and returns the Boolean expression as result of the search.
FulViewOCX::SelectAll	The method selects all document contents.

Name	Description
FulViewOCX::SetSelection	This method is used by Add-on container control, which notifies all loaded Add-ons that selection was changed.
FulViewOCX::ViewFile	The method shows the document.
FulViewOCX::ViewVersion	The method shows a given version of the document.
FulViewOCX::FirstHit	The method finds the first full text search criteria occurrence in the document.
FulViewOCX::LastHit	The method finds the last full text search criteria occurrence in the document.
FulViewOCX::NextHit	The method finds the next full text search criteria occurrence in the document.
FulViewOCX::PreviousHit	The method finds the previous full text search criteria occurrence in the document.
FulViewOCX::ShowStatusBar	The method shows or hides the status bar.
FulViewOCX::SetStatusBarText	The method shows the specified text in the status bar.

Events

Name	Description
FulViewOCX::Completed	Control fires the Completed event after download is completed.
FulViewOCX::HitsEnabled	This event called when hits can be enabled.
FulViewOCX::SelChanged	This event called when selection is changed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

Add-on container control

Profile form preview

Versions preview

Log viewer

Add-ons

FulViewOCX::BackColor

The property returns or sets a background color.

Syntax

```
object.BackColor
```

A hexadecimal number defines the color for a background.

Access

Read-Write

Example

(Visual Basic 6.0)

```
Private Sub ...
Viewer1.BackColor = &H8000000F&
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

Document preview

Add-on Container

AddOnCnt interface

Add-ons

FulViewOCX::Caption

The property returns or sets the text displayed in a control title bar.

Syntax

```
object.Caption
```

A string value that determines a text for the title bar.

Access

Read-Write

Example

(Visual Basic 6.0)

```
Private Sub ...
Viewer1.Caption = "Caption"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

Document preview

Add-on Container

AddOnCnt interface

Add-ons

FulViewOCX::Completed

Control fires the Completed event after download is completed.

Syntax

```
Private Sub object_Completed ()
```

Example

(Visual Basic 6.0)

```
Private Sub FulViewOCX1_Completed ()
...
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

FulViewOCX Control

FulViewOCX::CopyAll

The method copies a document text to the clipboard.

Syntax

```
object.CopyAll
```

Example

(Visual Basic 6.0)

```
Private Sub ...
Viewer1.CopyAll
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

Document preview

Add-on Container

AddOnCnt interface

Add-ons

FulViewOCX::CopySelection

The method copies the selected document text to the clipboard.

Syntax

```
object.CopySelection
```

Example

(Visual Basic 6.0)

```
Private Sub ...
Viewer1.CopySelectionEnd Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

Document preview

Add-on Container

AddOnCnt interface

Add-ons

FulViewOCX::FirstHit

The method finds the first full text search criteria occurrence in the document.

Syntax

```
object.FirstHit
```

Return value

Boolean

If the hit is found, the value is equal to True.

Example

(Visual Basic 6.0)

```
Private Sub ...
Dim bFound As Boolean
bFound = Viewer1.FirstHit
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

Document preview

Add-on Container

AddOnCnt interface

Add-ons

FulViewOCX::Font

The property returns or sets a Font object.

Syntax

```
object.Font
```

A Visual Basic StdFont object that determines a Font object. It is interpreted via the IFontDisp interface in Visual C++.

Access

Read-Write

Example

(Visual Basic 6.0)

```
Private Sub ...
Viewer1.Font.Bold = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

Document preview

Add-on Container

AddOnCnt interface

Add-ons

FulViewOCX::ForeColor

The property returns or sets the foreground color.

Syntax

```
object.ForeColor
```

A hexadecimal number defines the foreground color.

Access

Read-Write

Example

(Visual Basic 6.0)

```
Private Sub ...
Viewer1.ForeColor = &H80000012&
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

Document preview

Add-on Container

AddOnCnt interface

Add-ons

FulViewOCX::HitsEnabled

This event called when hits can be enabled.

Syntax

```
Private Sub object_HitsEnabled (bEnabled)
bEnabled
```

The parameter has the Boolean type and becomes True when the control can be enabled. Otherwise the parameter is False.

Example

(Visual Basic 6.0)

```
Private Sub FulViewOCX1_HitsEnabled (bEnabled As Boolean)
...
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

FulViewOCX Control

FulViewOCX::LastHit

The method finds the last full text search criteria occurrence in the document.

Syntax

```
object.LastHit
```

Return value

Boolean

If the hit is found, the value is equal to True.

Example

(Visual Basic 6.0)

```
Private Sub ...
Dim bFound As Boolean
bFound = Viewer1.LastHit
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

Document preview

Add-on Container

AddOnCnt interface

Add-ons

FulViewOCX::NextHit

The method finds the next full text search criteria occurrence in the document.

Syntax

```
object.NextHit
```

Return value

Boolean

If the hit is found, the value is equal to True.

Example

(Visual Basic 6.0)

```
Private Sub ...
Dim bFound As Boolean
bFound = Viewer1.NextHit
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

Document preview

Add-on Container

AddOnCnt interface

Add-ons

FulViewOCX::PreviousHit

The method finds the previous full text search criteria occurrence in the document.

Syntax

```
object.PreviousHit
```

Return value

Boolean

If the hit is found, the value is equal to True.

Example

(Visual Basic 6.0)

```
Private Sub ...
Dim bFound As Boolean
bFound = Viewer1.PreviousHit
```

```
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

Document preview

Add-on Container

AddOnCnt interface

Add-ons

FulViewOCX::PrintCurrentDoc

The method prints the selected document.

Syntax

```
object.PrintCurrentDoc
```

Example

(Visual Basic 6.0)

```
Private Sub ...
Viewer1.PrintCurrentDoc
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

Document preview

Add-on Container

AddOnCnt interface

Add-ons

FulViewOCX::PrintSetup

The method sets margins, paper source, paper size, paper orientation, and other layout options for the active document/its version.

Syntax

```
object.PrintSetup
```

Example

(Visual Basic 6.0)

```
Private Sub ...
Viewer1.PrintSetup
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

Document preview

Add-on Container

AddOnCnt interface

Add-ons

FulViewOCX::Search

The method returns the Boolean expression as result of the search.

Syntax

```
object.Search
```

Return value

Boolean

If the text is found, the value is equal to True.

Example

(Visual Basic 6.0)

```
Private Sub ...
Dim bSearch As Boolean
bSearch = Viewer1.Search
```

```
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

Document preview

Add-on Container

AddOnCnt interface

Add-ons

FulViewOCX::SearchDialog

The method sets criteria for search: string expression sought, expression that sets the starting position for each search, and specifies the type of a string comparison.

Syntax

```
object.SearchDialog lpszText, bFromTop, bCaseSensitive
```

`lpszText`

String expression sought.

`bFromTop`

The Boolean expression that sets the starting position for each search from beginning of the documents (True).

`bCaseSensitive`

If this parameter (Boolean) is True, the search is conducted with a binary comparison (True).

Example

(Visual Basic 6.0)

```
Private Sub ...
Viewer1.SearchDialog "Test1", True, False
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

Document preview

Add-on Container

AddOnCnt interface

Add-ons

FulView0CX::SearchNext

The method continues searching and returns the Boolean expression as result of the search.

Syntax

```
object.SearchNext
```

Return value

Boolean

If the text is found, the value is equal to True.

Example

(Visual Basic 6.0)

```
Private Sub ...
Dim bSearch As Boolean
bSearch = Viewer1.SearchNext
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

Document preview

Add-on Container

AddOnCnt interface

Add-ons

FulViewOCX::SelChanged

This event called when selection is changed.

Syntax

```
Private Sub object_SelChanged (bcanCopy)
bcanCopy
```

The parameter has the Boolean type and becomes True when the selected item can be copied to the clipboard. Otherwise the parameter is False.

Example

(Visual Basic 6.0)

```
Private Sub FulViewOCX1_SelChanged (bcanCopy As Boolean)
...
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

FulViewOCX Control

FulViewOCX::SelectAll

The method selects all document contents.

Syntax

```
object.SelectAll
```

Example

(Visual Basic 6.0)

```
Private Sub ...
Viewer1.SelectAll
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

Document preview

Add-on Container

AddOnCnt interface

Add-ons

FulView0CX::SetSelection

This method is used by Add-on container control, which notifies all loaded Add-ons that a selection was changed.

```
Syntax
object.SetSelection ObjectArray
ObjectArray
```

It is IBCObjectArray.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

Document preview

Add-on Container

Add-on container control

Add-ons

FulView0CX::SetStatusBarText

The method shows the specified text in the status bar.

Syntax

```
object.SetStatusBarText lpszString
lpszString
```

The string value for the text that should be shown in the status bar.

Example

(Visual Basic 6.0)

```
Private Sub ...
Viewer1.SetStatusBarText "Info"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

Document preview

Add-on Container

AddOnCnt interface

Add-ons

FulViewOCX::ShowStatusBar

The method shows or hides the status bar.

Syntax

```
object.ShowStatusBar bShow
```

bShow

The Boolean expression.

Example

(Visual Basic 6.0)

```
Private Sub ...
Viewer1.ShowStatusBar True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

Document preview

Add-on Container

AddOnCnt interface

Add-ons

FulViewOCX::ViewFile

The method shows a document.

Syntax

```
object.ViewFile lpszFileName
```

`lpszFileName`

Required. The parameter is a string value. It is a filename.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

Document preview

Add-on Container

AddOnCnt interface

Add-ons

FulViewOCX::ViewVersion

The method shows a given version of the document.

Syntax

```
object.ViewVersion Version
```

`Version`

Required. The parameter of the Unknown type. It is a version of the document.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

Document preview

Add-on Container

AddOnCnt interface

Add-ons

Versions Add-on

This control previews a document versions.

Except described the control supports also standard properties, methods, and events described in Visual Basic.

This add-on is implemented via the VersionsAddon control:

Properties

Name	Description
VersionsAddon::Font	The property returns or sets a Font object.
VersionsAddon::Caption	The property returns or sets the text displayed in a control title bar.

Methods

Name	Description
VersionsAddon::SetSelection	This method used by Add-on container, which notifies all loaded Add-ons that a selection was changed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

AddOnCnt Interface

Document preview

Profile form preview

Log viewer

Add-ons

VersionsAddon::Caption

The property returns or sets the text displayed in the control title bar.

Syntax

```
object.Caption
```

A string value that determines a text for the title bar.

Access

Read-Write

Example

(Visual Basic 6.0)

```
Private Sub ...
VersionsAddon1.Caption = "Caption"
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

Versions preview

Add-on Container

AddOnCnt interface

Add-ons

VersionsAddon::Font

The property returns or sets a Font object.

Syntax

```
object.Font
```

A Visual Basic StdFont object that determines a Font object. It is interpreted via the IFontDisp interface in Visual C++.

Access

Read-Write

Example

(Visual Basic 6.0)

```
Private Sub ...
VersionsAddon1.Font.Bold = True
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

Versions preview

Add-on Container

AddOnCnt interface

Add-ons

VersionsAddon::SetSelection

This method used by Add-on container, which notifies all loaded Add-ons that a selection was changed.

Syntax

```
object.SetSelection ObjectArray
```

`ObjectArray`

It is a IBCObjectArray.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

Versions preview

Add-on Container

AddOnCnt interface

Add-ons

Logger Add-On

This control logs DM Extensions errors.

The control is shown in Add-on panel of the DM Extensions for observing the errors. The log file (filename, and other) is specified separately in the special settings of Registry.

This add-on is implemented via the LoggerOCX and LoggerOCXNotificationSink controls.

See Also

"LoggerOCX Control" on page 1243

"LoggerOCXNotificationSink Control" on page 1244

LoggerOCX Control

The object contains only one event, which is fired after an appropriate notification message.

Except described the control supports also standard properties, methods, and events of the ActiveX described in Visual Basic.

For the information about registration of this control refer to the Add-on container.

Events

Name	Description
ChangeStatus	Control fires the ChangeStatus event after an appropriate notification message from the Explorer Core.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

AddOnCnt Interface

Document preview

Profile form preview

Versions preview

Add-ons

LoggerOCX::ChangeStatus

Control fires the ChangeStatus event after an appropriate notification message from the Explorer Core.

Syntax

```
Private Sub object_ChangeStatus (bParam)
```

`bParam`

The parameter has the Boolean type and becomes True when the control gets the message with a fatal error. Otherwise the parameter is False.

Example

(Visual Basic 6.0)

```
Private Sub History1_ChangeStatus (bParam As Boolean)
...
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

LoggerOCX Control

Log viewer

Add-on Container

AddOnCnt interface

Add-ons

LoggerOCXNotificationSink Control

The object contains only one method. This method saves the error message.

This object should be registered in the Registry:

1. It is registered under the registry key HKEY_LOCAL_MACHINE\Software\Hummingbird\PowerDOCS\Core.
2. This key has the following values:

Key	Value	Description
EnabledLogger	0 (Default)	This object is not available.
1	Object is available.	LoggerFile
"C:\Program Files\Hummingbird\PowerDOCS\PowerDOCS.log"	The name of the log file.	

Methods

Name	Description
LoggerOCXNotificationSink::EventFired	The method saves an error message with some complementary information about the error.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

AddOnCnt Interface

Document preview

Profile form preview

Versions preview

Add-ons

LoggerOCXNotificationSink::EventFired

The method saves error message with some complementary information about the error.

Syntax

```
Private Sub object_EventFired (pszCategory, lEventID,
lpszDescription, nSeverity)
```

`pszCategory`

The parameter has the string type. This parameter is a name of the module, which brings an event.

`lEventID`

The parameter of the long type is identifier of the event. There is the Thread ID, in which an event has taken place.

`lpszDescription`

The parameter of the string type is a description of the bug event.

`nSeverity`

The parameter has the integer type. The value specifies the event severity. It can be following:

Constant	Value	Description
logTrace	0	Debug trace
logInfo	1	Information message
logSevereInfo	2	Severe information message - always logged
logWarning	3	Warning
logError	4	Error
logFatal	5	Fatal error

Example

(Visual Basic 6.0)

```
Sub ...
Dim IDEvent
'Some operation with IDEvent
...
LoggerOCXNotificationSink.EventFired "FsPlugin", IDEvent,
"Division by ziro", logFatal
End Sub
```

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "AddOnCnt.ocx"

See Also

LoggerOCXNotificationSink Control

Log viewer

Add-on Container

AddOnCnt interface

Add-ons

Chapter

12

E-Mail Integration

In This Chapter

- "About E-Mail Integration" on page 1250
- "Using E-Mail Integration" on page 1250
- "E-Mail Integration Interfaces" on page 1257
- "Email Integration Enumerations" on page 1288

About E-Mail Integration

DM Extensions E-Mail Integration allows to integrate DM into e-mail client (MS Outlook, Groupwise, Lotus Notes). All e-mail client modules work with DM using the E-Mail integration Core module. Microsoft Outlook DM Extension uses the E-Mail integration core module for drag-and-drop support. API for the Core module is described in this section.

Using E-Mail Integration

Scenario of using E-Mail integration API in your application.

This scenario provides simple illustration of using DM Extensions E-Mail integration API.

Step 1 through Step 3 of this tutorial provide the introduction to the following framework features.

- Outlook MailItem object
- DM Extensions EMICORELib.EIAttachment inteface
- DM Extensions EMICORELib.EIMessage interface
- Creation of a new E-Mail message
- Attaching documents from DM
- Storing an e-mail message in DM

See Also

"Using E-Mail Integration Step 1" on page 1250

"Using E-Mail Integration Step 2" on page 1252

"Using E-Mail Integration Step 3" on page 1255

Using E-Mail Integration Step 1

Scenario of using E-Mail integration API in your application.

Step 1

This step shows how to create own class for the EMICORELib.EIAttachment interface implementation.

1. Create a new VB project, choose New Project from the File menu, then select Standard EXE and press OK. Open Project References and select EMICore 1.0 Type Library and Microsoft Outlook 9.0 object library components.

2. Choose Add Class Module from the Project menu. Name it as AttImpl.cls.

3. Declare EMICORELib.EIAttachment interface implementation.

```
Implements EMICORELib.IAttachment
```

4. Declare the private Outlook.Attachment variable for class encapsulation. Declare attachment type and number variables.

```
Private parent As Outlook.Attachment
Private vType As EMICORELib.AttachmentType
Private lNumber As Long
```

5. Create a method for class initializing

```
Public Sub Cloak(ByRef mailAtt As Outlook.Attachment, _
    ByVal vAttType As EMICORELib.AttachmentType, _
    ByVal attNumber As Long)
  Set parent = mailAtt
  vType = vAttType
  lNumber = attNumber
End Sub
```

6. Add EIAttachment properties and method implementation

```
Public Property Get EIAttachment_Type() As
EMICORELib.AttachmentType
  EIAttachment_Type = vType
End Property

Public Property Get EIAttachment_Name() As String
  If vType = AT_FILE Then
    EIAttachment_Name = parent.FileName
  Else
    EIAttachment_Name = parent.DisplayName
  End If
End Property

Public Property Get EIAttachment_Extension() As String
```

```
  If vType = AT_MESSAGE Then
    EIAttachment_Extension = "E-Mail"
  Else
    EIAttachment_Extension = Right(parent.FileName, 3)
  End If
End Property

Public Property Get EIAttachment_Number() As Long
  EIAttachment_Number = lNumber
End Property

Public Sub EIAttachment_SaveToFile(ByVal bstrFilePath As String)
  parent.SaveAsFile bstrFilePath
End Sub
```

See Also

"Using E-Mail Integration" on page 1250

"Using E-Mail Integration Step 2" on page 1252

"Using E-Mail Integration Step 3" on page 1255

Using E-Mail Integration Step 2

Scenario of using E-Mail integration API in your application.

Step 2

This step shows how to create own class for the EMICORELib.EIMessage interface implementation.

1. Choose Add Class Module from the Project menu. Name it as MsgImpl.cls.
2. Declare EMICORELib.EIMessage interface implementation.

```
Private WithEvents vMail As Outlook.MailItem
Dim pQR As New QuickRetrieve
```

3. Declare Outlook application variable.

```
Private objOL As Outlook.Application
```

4. Create class initialization and finalization scenarios

```
Private Sub Class_Initialize()
```

```
  Set objOL = CreateObject("Outlook.Application")
  Set vMail = objOL.CreateItem(olMailItem)
End Sub

Private Sub Class_Terminate()
  vMail.Close olDiscard
  Set vMail = Nothing
  Set objOL = Nothing
End Sub
```

5 Add code for properties implementation. Values should be retrieved from correspondent Outlook.MailItem properties. Next are just a few examples.

```
Public Property Get EIMessage_From() As String
  EIMessage_From = vMail.SenderName
End Property

Public Property Get EIMessage_To() As String
  EIMessage_To = vMail.To
End Property

Public Property Get EIMessage_ThreadPosition() As Long
  EIMessage_ThreadPosition = vMail.ConversationIndex
End Property
```

6 Add code for Attachments property implementation. You will use an attachment class made in Step 1.

```
' Transform Outlook.Attachments collection to EIAttachments
Public Property Get EIMessage_Attachments() As
EMICORELib.EIAttachments
  Dim oAtts As EMICORELib.EIAttachments
  Set oAtts = New EIAttachments

  Dim i As Integer
  For i = 1 To vMail.Attachments.Count
    ' Visual Basic Polymorphism
    Dim oAtt As EMICORELib.EIAttachment
    Dim oExtAtt As AttImpl
    Set oAtt = New AttImpl
    Set oExtAtt = oAtt
```

```
    ' Initialize attachment properties
    oExtAtt.Cloak vMail.Attachments.Item(i), AT_FILE, i

    ' Add to collection
    oAtts.Add oAtt
  Next

  Set EIMessage_Attachments = oAtts
End Property
```

7 Create the method for saving message to a file

```
Public Sub EIMessage_SaveToFile(ByVal eFileFormat As
EISaveFileFormat, _
                                ByVal bstrFilePath As String)
  Select Case eFileFormat
    ' EISFF_RTF_BODY = 1
    ' EISFF_RTF_ATTACHMENTS = 2
    Case 1, 2
        vMail.SaveAs bstrFilePath, olRTF
    ' EISFF_MSG_BODY = 3
    ' EISFF_MSG_ATTACHMENTS = 4
    Case 3, 4
      vMail.SaveAs bstrFilePath
  End Select
End Sub
```

8 Add setters to fill in Outlook.MailItem properties. Next are just a few examples.

```
Public Function setTo(ByVal vNewValue As String)
  vMail.To = vNewValue
End Function

Public Function setBCC(ByVal vNewValue As String)
  vMail.BCC = vNewValue
End Function

Public Function setSubject(ByVal vNewValue As Variant)
  vMail.Subject = vNewValue
End Function
```

9 Add a message Body, Attachments property and Display method

```
Public Property Get Body() As String
  Body = vMail.Body
End Property

Public Property Let Body(ByVal vNewValue As String)
  vMail.Body = vNewValue
End Property

Public Sub Display()
  vMail.Display
End Sub

Public Property Get Attachments() As Outlook.Attachments
  Set Attachments = vMail.Attachments
End Property
```

See Also

"Using E-Mail Integration" on page 1250

"Using E-Mail Integration Step 1" on page 1250

"Using E-Mail Integration Step 3" on page 1255

Using E-Mail Integration Step 3

Scenario of using E-Mail integration API in your application.

Step 3

This step shows how to use created classes to interact with DM.

1 Choose a single form automatically added while creating a new project. Name it as frmMain.frm.

2 Declare a variable for the EMICore library.

```
Dim objEICore As EMICORELib.EICore
```

3 Declare variable for the MsgImpl object.

```
Private oMailItem As EMICORELib.EIMessage
```

4. Drop three buttons to form in visual designer. Set their captions to "New E-mail Message", "Insert document", "Save message". Name them as cmdNew, cmdAttach, cmdSave.

5. Click the cmdNew button and write next as button handler procedure. This handler will create a new E-Mail message, fill in fields and display it.

```
' Create new Outlook message
Private Sub cmdNew_Click()
  If oMailItem Is Nothing Then
    Set oMailItem = New MsgImpl

    Dim oExtMail As MsgImpl
    Set oExtMail = oMailItem

    oExtMail.setTo "PowerDOCS_developer@example.com"
    oExtMail.setSubject "E-Mail integration message sample"
    oExtMail.Body = "This is an example of message, " & vbCrLf & _
                            "created using DM Extensions E-Mail
integration features" & vbCrLf
    oExtMail.Display
  End If
End Sub
```

6. Click the cmdAttach button and write next as button handler procedure. This handler will retrieve selected documents from the DM storage; save them as files and attach to the current message.

```
' Attach selected DM documents to new Outlook message
' New Outlook message will be created if needed
Private Sub cmdAttach_Click()
  Dim varFiles As Variant

  ' Create message if needed
  If oMailItem Is Nothing Then
    cmdNew_Click
  End If

  ' Logon to DM
  If objEICore Is Nothing Then
```

```
    Set objEICore = New EMICORELib.EICore
  End If

  ' Request documents from DM
  varFiles = objEICore.InsertDocuments(Me.hWnd, False)

  Set objEICore = Nothing

  Dim oExtMail As MsgImpl
  Set oExtMail = oMailItem

  Dim i As Integer
  ' Attach them to the message and add file description line to
message body
  If Not IsEmpty(varFiles) Then
    For i = LBound(varFiles) To UBound(varFiles)
      oExtMail.Attachments.Add (varFiles(i))
      oExtMail.Body = oExtMail.Body & "File " & varFiles(i) & "
attached" & vbCrLf
    Next
  End If

End Sub
```

7 At last, you can run application to see it in action. Select Start item from VB menu Run or press the F5 button.

See Also

"Using E-Mail Integration" on page 1250

"Using E-Mail Integration Step 1" on page 1250

"Using E-Mail Integration Step 2" on page 1252

E-Mail Integration Interfaces

- "IEICore" on page 1258
- "IEIMessage" on page 1261
- "IEIMessage::BCC" on page 1264

- “IEIAttachment” on page 1272
- “IEIMessages” on page 1277
- “IEIAttachments” on page 1282

IEICore

The interface is implemented by the e-mail integration core module and used by e-mail client modules.

Methods

IEICore Methods	Description
IEICore::SaveMessages	Saves E-mail messages to the DM system.
IEICore::InsertDocuments	Selects documents to insert into the e-mail message.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

IEICore::SaveMessages

Saves message(s) to DM.

C++ Syntax

```
HRESULT SaveMessages(
  long nWnd,  long nFlags  IEIMessages* pMessages,  VARIANT
vParent);
```

Visual Basic Syntax

```
[  nWnd = ] EICore.SaveMessages
```

Parameters

nWnd	[in] Specifies the HWND handle of an active window.
nFlags	[in] Specifies how a message should be saved. This parameter can be combination of EISaveFlags enumeration values.
pMessages	[in] Specifies messages that should be saved.
vParent	[in, optional] Specifies a parent object. At this time the following parent objects are supported:

Supported Interface	Description
ILibrary	Messages will be saved into this library. In other case messages will be saved into the current library.
IProject	Saved messages will be added to this folder.

Return Values

This method supports the standard return values E_POINTER and E_UNEXPECTED as well as the following:

S_OK	The messages have been saved successfully.
S_FALSE	The messages have not been saved.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEICore" on page 1258

"IEIMessages" on page 1277

"EISaveFlags Enumeration" on page 1288

IEICore::InsertDocuments

Inserts document(s) into e-mail messages. This method returns an array of file names that should be inserted into the e-mail message.

C++ Syntax

```
HRESULT InsertDocuments(
  long nWnd,  VARIANT vMaxDocs,  VARIANT* pFiles);
```

Visual Basic Syntax

```
[  pFiles  = ] EICore.InsertDocuments ( nWnd,  vMaxDocs)
```

Parameters

pFiles	[out, retval] Returns an array of strings that contain file names of documents that should be inserted.
nWnd	[in] Specifies the HWND handle of an active window.
vMaxDocs	[in, optional] Specifies whether or not multiple documents can be inserted. If this parameter contains a Boolean variable:

Value	Description
True	More than one document can be inserted.
False	Only one document can be inserted.

If this parameter contains an integer variable:

Value	Description
> 1	More than one document can be inserted.
<= 1	Only one document can be inserted.

If this parameter is not specified then more than one document can be inserted.

Return Values

This method supports the standard return values E_POINTER and E_UNEXPECTED as well as the following:

S_OK	The files for documents have been created and stored in a temporary directory.
S_FALSE	An error occurs during inserting documents or operation was cancelled by user.

Remarks

The returned files are created in a temporary directory. Files and temporary subdirectory are removed when the Core module is unloaded from the memory. E-mail integration modules itself can delete these files after using to free a disk space.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEICore" on page 1258

IEIMessage

The interface is implemented by the e-mail client module and used to communicate with the Core module.

Methods

IEIMessage Methods	Description
IEIMessage::SaveToFile	Saves the message to a file using specified file format.
IEIMessage::Delete	Deletes the message from a message store.

Properties

IEIMessage Properties	Description
IEIMessage::From	From message field.
IEIMessage::To	To message recipient(s).
IEIMessage::CC	CC message recipient(s).
IEIMessage::BCC	BCC message recipient(s).
IEIMessage::Subject	Message subject.
IEIMessage::SendDate	Message send date.
IEIMessage::Attachments	Message attachments collection.
IEIMessage::ID	Message identifier.
IEIMessage::ParentID	Parent message identifier.
IEIMessage::ThreadPosition	Message position in a thread.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

IEIMessage::From

Returns a message originator.

C++ Syntax

```
HRESULT From(
  BSTR* pVal);
```

Visual Basic Syntax

```
[  pVal = ] EIMessage.From
```

Parameters

pVal	[out, retval] Returns a message originator.

Return Values

S_OK	The operation succeeded.
Other value	The operation failed.

Access

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIMessage" on page 1261

IEIMessage::To

Returns message recipients as specified in "To:" fields.

C++ Syntax

```
HRESULT To(
  BSTR* pVal);
```

Visual Basic Syntax

```
[  pVal = ] EIMessage.To
```

Parameters

pVal	[out, retval] Returns message "To:" recipients.

Return Values

S_OK	The operation succeeded.
Other value	The operation failed.

Access

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIMessage" on page 1261

IEIMessage::CC

Returns message recipients as specified in "CC:" fields.

C++ Syntax

```
HRESULT CC(
  BSTR* pVal);
```

Visual Basic Syntax

```
[  pVal = ] EIMessage.CC
```

Parameters

pVal	[out, retval] Returns message "CC:" recipients.

Return Values

S_OK	The operation succeeded.
Other value	The operation failed.

Access

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIMessage" on page 1261

IEIMessage::BCC

Returns message recipients as specified in "BCC:" fields.

C++ Syntax

```
HRESULT BCC(
  BSTR* pVal);
```

Visual Basic Syntax

```
[  pVal = ] EIMessage.BCC
```

Parameters

pVal	[out, retval] Returns message "BCC:" recipients.

Return Values

S_OK	The operation succeeded.
Other value	The operation failed.

Access

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIMessage" on page 1261

IEIMessage::Subject

Returns a message subject.

C++ Syntax

```
HRESULT Subject(
  BSTR* pVal);
```

Visual Basic Syntax

```
[  pVal = ] EIMessage.Subject
```

Parameters

pVal	[out, retval] Returns a message subject.

Return Values

S_OK	The operation succeeded.
Other value	The operation failed.

Access

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIMessage" on page 1261

IEIMessage::SendDate

Returns a message send date in the localized string.

C++ Syntax

```
HRESULT SendDate(
  BSTR* pVal);
```

Visual Basic Syntax

```
[  pVal = ] EIMessage.SendDate
```

Parameters

pVal	[out, retval] Returns a message send date.

Return Values

S_OK	The operation succeeded.
Other value	The operation failed.

Access

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIMessage" on page 1261

IEIMessage::Attachments

Returns a message attachment collection.

C++ Syntax

```
HRESULT Attachments(
  IEIAttachments** pVal);
```

Visual Basic Syntax

```
[  pVal = ] EIMessage.Attachments
```

Parameters

pVal	[out, retval] Returns an attachment collection.

Return Values

S_OK	The operation succeeded.
Other value	The operation failed.

Access

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIMessage" on page 1261

"IEIAttachments" on page 1282

IEIMessage::ID

Returns a message unique identifier.

C++ Syntax

```
HRESULT ID(
  BSTR* pVal);
```

Visual Basic Syntax

```
[  pVal = ] EIMessage.ID
```

Parameters

pVal	[out, retval] Returns a message unique identifier.

Return Values

S_OK	The operation succeeded.
Other value	The operation failed.

Access

Read Only

Remarks

The e-mail client module should return a unique identifier for the current message. This field is used for "Enhanced" e-mail integration to identify a message. The length of an identifier should be limited to 80 characters.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIMessage" on page 1261

IEIMessage::ParentID

Returns a parent message unique identifier.

C++ Syntax

```
HRESULT ParentID(
  BSTR* pVal);
```

Visual Basic Syntax

```
[  pVal = ] EIMessage.ParentID
```

Parameters

pVal	[out, retval] Returns a parent message unique identifier.

Return Values

S_OK	The operation succeeded.
Other value	The operation failed.

Access

Read Only

Remarks

The e-mail client module should return a unique identifier for a parent of the current message. This field is used for "Enhanced" e-mail integration to identify a parent message. The length of an identifier should be limited to 60 characters. If there is no information about a parent message the empty string should be returned.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIMessage" on page 1261

IEIMessage::ThreadPosition

Returns the position of a message in the messages thread.

C++ Syntax

```
HRESULT ThreadPosition(
  long* pVal);
```

Visual Basic Syntax

```
[  pVal = ] EIMessage.ThreadPosition
```

Parameters

pVal	[out, retval] Returns a message position in the messages thread.

Return Values

S_OK	The operation succeeded.
Other value	The operation failed.

Access

Read Only

Remarks

If there is no information about a parent message, 0 should be returned.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIMessage" on page 1261

IEIMessage::SaveToFile

Save a message to the specified file.

C++ Syntax

```
HRESULT SaveToFile(
  EISaveFileFormat eFileFormat,  BSTR bstrFilePath);
```

Visual Basic Syntax

```
EIMessage.SaveToFile  eFileFormat, bstrFilePath
```

Parameters

eFileFormat	[in] Specifies how a message should be saved. Can be one from EISaveFileFormat enumeration values.
bstrFilePath	[in] A full path to a file that should be used to save into.

Return Values

S_OK	The operation succeeded.
Other value	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIMessage" on page 1261

"EISaveFileFormat Enumeration" on page 1289

IEIMessage::Delete

Deletes a message from the message store.

C++ Syntax

```
HRESULT Delete();
```

Visual Basic Syntax

```
EIMessage.Delete
```

Parameters

None

Return Values

S_OK	The operation succeeded.
Other value	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIMessage" on page 1261

IEIAttachment

The interface is implemented by the e-mail client module and used to communicate with the Core module.

Methods

IEIAttachment Methods	Description
IEIAttachment::SaveToFile	Saves an attachment to a file.

Properties

IEIAttachment Properties	Description
IEIAttachment::Name	Attachment name or file name.
IEIAttachment::Extension	Attachment extension.
IEIAttachment::Type	Attachment type.
IEIAttachment::Number	Attachment number.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

IEIAttachment::Name

Returns an attachment name (e-mail subject for e-mail message attachment) or an attachment file name.

C++ Syntax

```
HRESULT Name(
  BSTR* pVal);
```

Visual Basic Syntax

```
[  pVal = ] EIAttachment.Name
```

Parameters

pVal	[out, retval] Returns an attachment name.

Return Values

S_OK	The operation succeeded.
Other value	The operation failed.

Access

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIAttachment" on page 1272

IEIAttachment::Extension

Returns the attachment extension (for e-mail messages the "E-mail" string can be used).

C++ Syntax

```
HRESULT Extension(
  BSTR* pVal);
```

Visual Basic Syntax

```
[  pVal = ] EIAttachment.Extension
```

Parameters

pVal	[out, retval] Returns the attachment extension.

Return Values

S_OK	The operation succeeded.
Other value	The operation failed.

Access

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIAttachment" on page 1272

IEIAttachment::Type

Returns an attachment type.

C++ Syntax

```
HRESULT Type(
  AttachmentType* pVal);
```

Visual Basic Syntax

```
[  pVal = ] EIAttachment.Type
```

Parameters

pVal	[out, retval] Returns an attachment type. For possible values see AttachmentType Enumeration.

Return Values

S_OK	The operation succeeded.
Other value	The operation failed.

Access

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIAttachment" on page 1272

"AttachmentType Enumeration" on page 1290

IEIAttachment::Number

Returns an attachment number.

C++ Syntax

```
HRESULT Number(
  long* pVal);
```

Visual Basic Syntax

```
[  pVal = ] EIAttachment.Number
```

Parameters

pVal	[out, retval] Returns an attachment number.

Return Values

S_OK	The operation succeeded.
Other value	The operation failed.

Access

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIAttachment" on page 1272

IEIAttachment::SaveToFile

Saves a message to the specified file.

C++ Syntax

```
HRESULT SaveToFile(
  BSTR bstrFilePath);
```

Visual Basic Syntax

```
EIAttachment.SaveToFile  bstrFilePath
```

Parameters

bstrFilePath	[in] A full path to the file that should be used to save into.

Return Values

S_OK	The operation succeeded.
Other value	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIAttachment" on page 1272

IEIMessages

The interface is implemented by the Core module and is used to store messages.

Methods

IEIMessages Methods	Description
IEIMessages::Add	Adds an item to a collection.
IEIMessages::Remove	Removes a specific item from a collection.
IEIMessages::Clear	Removes all items from a collection.

Properties

IEIMessages Properties	Description
IEIMessages::Item	Returns a specific item of a collection either by position or by key.
IEIMessages::Count	Returns the number of items in a collection.
IEIMessages::_NewEnum	Creates a new enumerator object.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

IEIMessages::Item

Returns the specified item from a collection.

C++ Syntax

```
HRESULT Item(
  VARIANT vIndex,  IEIMessage** ppMessage);
```

Visual Basic Syntax

```
[ ppMessage = ] EIMessages.Item(vIndex)
```

Parameters

vIndex	[in] A numeric expression that specifies the position of a collection member. vIndex must be a number from 1 to the value of the collection's IEIMessages::Count property.
ppMessage	[out, retval] Returns the IEIMessage object that specifies the requested item.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access

Read Only

Remarks

If the value provided as index does not match any existing member of the collection, an error occurs. Item is the default property for a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIMessages" on page 1277

"IEIMessage" on page 1261

"IEIMessages::Count" on page 1278

IEIMessages::Count

The property returns the number of items in a collection.

C++ Syntax

```
HRESULT Count(
  long* plCount);
```

Visual Basic Syntax

```
[  plCount = ] EIMessages.Count
```

Parameters

plCount	[out, retval] Returns a long that specifies the number of items in the collection.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIMessages" on page 1277

IEIMessages::_NewEnum

Creates a new enumerator object.

C++ Syntax

```
HRESULT _NewEnum(
  IUnknown** ppEnum);
```

Parameters

ppEnum	[out, retval] Specifies an address where a pointer to the IUnkown interface of a new enumerator should be returned.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIMessages" on page 1277

IEIMessages::Add

The method adds a new item to a collection.

C++ Syntax

```
HRESULT Add(
  IEIMessage* pMessage);
```

Visual Basic Syntax

```
EIMessages.Add pMessage
```

Parameters

pMessage	[in] The IEIMessage object that specifies the item to add to the collection.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIMessages" on page 1277

"IEIMessage" on page 1261

IEIMessages::Remove

The method removes a specific item from a collection.

C++ Syntax

```
HRESULT Remove(
  VARIANT vIndex);
```

Visual Basic Syntax

```
EIMessages.Remove vIndex
```

Parameters

vIndex	[in] A numeric expression specifying the index of the object in the collection to be removed.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIMessages" on page 1277

IEIMessages::Clear

The method removes all items from a collection.

C++ Syntax

```
HRESULT Clear();
```

Visual Basic Syntax

```
EIMessages.Clear
```

Parameters

None.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIMessages" on page 1277

IEIAttachments

The interface is implemented by the Core module and is used to store messages.

Methods

IEIAttachments Methods	Description
IEIAttachments::Add	Adds an item to a collection.
IEIAttachments::Remove	Removes a specific item from a collection.
IEIAttachments::Clear	Removes all items from a collection.

Properties

IEIAttachments Properties	Description
IEIAttachments::Item	Returns a specific item of a collection either by position or by key.
IEIAttachments::Count	Returns the number of items in a collection.
IEIAttachments::_NewEnum	Creates a new enumerator object.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

IEIAttachments::Item

Returns the specified item from a collection.

C++ Syntax

```
HRESULT Item(
  VARIANT vIndex,  IEIAttachment** ppAttachment);
```

Visual Basic Syntax

```
[ ppAttachment = ] EIAttachments.Item(vIndex)
```

Parameters

vIndex	[in] A numeric expression that specifies the position of a collection member. vIndex must be a number from 1 to the value of the collection's IEIAttachments::Count property.
ppAttachment	[out, retval] Returns an IEIAttachment object that specifies the requested item.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access

Read Only

Remarks

If the value provided as index does not match any existing member of the collection, an error occurs. Item is the default property for a collection.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIAttachments" on page 1282

"IEIAttachment" on page 1272

"IEIAttachments::Count" on page 1284

IEIAttachments::Count

The property returns the number of items in a collection.

C++ Syntax

```
HRESULT Count(
  long* plCount);
```

Visual Basic Syntax

```
[  plCount = ] EIAttachments.Count
```

Parameters

plCount	[out, retval] Returns a long that specifies the number of items in the collection.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIAttachments" on page 1282

IEIAttachments::_NewEnum

Creates a new enumerator object.

C++ Syntax

```
HRESULT _NewEnum(
  IUnknown** ppEnum);
```

Parameters

ppEnum	[out, retval] Specifies an address where a pointer to the IUnkown interface of a new enumerator should be returned.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access

Read Only

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIAttachments" on page 1282

IEIAttachments::Add

The method adds a new item to a collection.

C++ Syntax

```
HRESULT Add(
  IEIAttachment* pAttachment);
```

Visual Basic Syntax

```
EIAttachments.Add pAttachment
```

Parameters

pAttachment	[in] An IEIAttachment object that specifies the item to add to the collection.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIAttachments" on page 1282

"IEIAttachment" on page 1272

IEIAttachments::Remove

The method removes a specific item from a collection.

C++ Syntax

```
HRESULT Remove(
  VARIANT vIndex);
```

Visual Basic Syntax

```
EIAttachments.Remove vIndex
```

Parameters

vIndex	[in] A numeric expression specifying the index of the object in the collection to be removed.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIAttachments" on page 1282

IEIAttachments::Clear

The method removes all items from a collection.

C++ Syntax

```
HRESULT Clear();
```

Visual Basic Syntax

```
EIAttachments.Clear
```

Parameters

None.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIAttachments" on page 1282

Email Integration Enumerations

- "EISaveFlags Enumeration" on page 1288
- "EISaveFileFormat Enumeration" on page 1289
- "AttachmentType Enumeration" on page 1290

EISaveFlags Enumeration

This enumeration contains flags that are passed to IEICore::SaveMessages as the nFlags parameter.

Elements

EISaveFlags Elements	Description
EISF_CAN_SAVE_AS_MSG	Indicates that message(s) can be saved in the MSG format.
EISF_CAN_DELETE	Indicates that message(s) can be deleted after saving.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEICore" on page 1258

EISaveFileFormat Enumeration

This enumeration contains flags that are passed to IEIMessage::SaveToFile as the eFileFormat parameter.

Elements

EISaveFileFormat Elements	Description
EISFF_RTF_BODY	Indicates that the message should save to a file only it's body in the RTF format.
EISFF_RTF_ATTACHMENTS	Indicates that the message should save to a file it's body and all attachments in the RTF format.
EISFF_MSG_BODY	Indicates that the message should save to a file only it's body in the MSG format.
EISFF_MSG_ATTACHMENTS	Indicates that the message should save to a file it's body and all attachments in the MSG format.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIMessage" on page 1261

AttachmentType Enumeration

This enumeration contains flags that are returned by the IEIAttachment::Type property in the pVal parameter.

Elements

AttachmentType Elements	Description
AT_FILE	Indicates that an attachment is a file.
AT_MESSAGE	Indicates that an attachment is an e-mail message.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 4.0.1 (or later).

Header: #import "EMICore.idl"

See Also

"IEIAttachment" on page 1272

Chapter

13

Cache Manager

In This Chapter

- "About Cache Manager" on page 1292
- "Cache Manager Interfaces" on page 1292

About Cache Manager

The cache manager module downloads and uploads files from DM Server storage into a local computer.

Cache Manager Interfaces

- "IDownloadDataSink" on page 1292
- "IDownloadResourceProvider" on page 1293
- "IDownloadSession" on page 1299
- "IDownloadSink" on page 1302
- "ILocalFileCacheManager" on page 1306
- "IUploadResourceProvider" on page 1313

IDownloadDataSink

Downloads/uploads the progress sink.

When to Implement

Implemented by a client as an additional interface to IDownloadSink.

When to Use

Can be obtained via QueryInterface on IDownloadSink - in this case the manager will send data to notifications. Called by the download/upload manager module.

Methods

IDownloadDataSink Methods	Description
IDownloadDataSink::OnData	Called when a new portion of data arrives.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CacheMgr.dll"

See Also

"IDownloadSink" on page 1302

IDownloadDataSink::OnData

Called when a new portion of data arrives.

C++ Syntax

```
HRESULT OnData(
  DWORD dwLastPosition,  DWORD dwSize,  BYTE* pData);
```

Parameters

dwLastPosition	[in] A data position for the previous call.
dwSize	[in] The size of data currently read.
pData	[in, size_is(dwSize)] A buffer with data.

Return Values

S_OK	The operation succeeded.
E_POINTER	The parameter cannot be NULL.
E_INVALIDARG	An invalid parameter passed.
E_FAIL	An unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CacheMgr.dll"

See Also

"IDownloadDataSink" on page 1292

IDownloadResourceProvider

Downloads the resource provider.

When to Implement

Implemented by a resource provider object.

When to Use

Called by the download manager.

Methods

IDownloadResourceProvider Methods	Description
IDownloadResourceProvider::GetLocalFileName	Determines a local filename where a resource must be downloaded.
IDownloadResourceProvider::GetLastChangeDate	Determines the last change date of a resource.
IDownloadResourceProvider::GetSize	Determines a resource size.
IDownloadResourceProvider::OpenForRead	Opens a stream to download the resource to a local file.
IDownloadResourceProvider::GetLastError	Returns last error text from the provider.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CacheMgr.dll"

IDownloadResourceProvider::GetLocalFileName

The method determines local filename where resource must be downloaded.

C++ Syntax

```
HRESULT GetLocalFileName(
  BSTR lpszResourceID,  BSTR lpszParam,  BSTR* pszFileName);
```

Parameters

lpszResourceID	[in] Resource identification. Format of this string is resource provider specific.
lpszParam	[in] String with resource provider specific data.
pszFileName	[out, retval] Returns local filename (full pathname) where resource must be downloaded.

Return Values

S_OK	The operation succeeded.
E_POINTER	Parameter cannot be NULL.
E_INVALIDARG	Invalid parameter passed.
E_FAIL	Unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CacheMgr.dll"

See Also

"IDownloadResourceProvider" on page 1293

IDownloadResourceProvider::GetLastChangeDate

The method determines last change date of resource.

C++ Syntax

```
HRESULT GetLastChangeDate(
  BSTR lpszResourceID,  BSTR lpszParam,  DATE* dLastChangeDate);
```

Parameters

lpszResourceID	[in] Resource identification. Format of this string is resource provider specific.
lpszParam	[in] String with resource provider specific data.
dLastChangeDate	[out, retval] Returns last change date of the resource.

Return values

S_OK	The operation succeeded.
E_POINTER	Parameter cannot be NULL.
E_INVALIDARG	Invalid parameter passed.
E_FAIL	Unexpected error occurs.

Remarks

Used to check cache validity.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CacheMgr.dll"

See Also

"IDownloadResourceProvider" on page 1293

IDownloadResourceProvider::GetSize

Determines a resource size.

C++ Syntax

```
HRESULT GetSize(
  BSTR lpszResourceID,  BSTR lpszParam,  DWORD* dwSize);
```

Parameters

lpszResourceID	[in] Resource identification. The format of this string is resource provider's specific one.
lpszParam	[in] A string with resource provider's specific data.
dwSize	[out, retval] Returns a resource size.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_INVALIDARG	An invalid parameter passed.
E_FAIL	An unexpected error occurs.

Remarks

Cannot be implemented, because it leads to notification inaccuracy.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CacheMgr.dll"

See Also

"IDownloadResourceProvider" on page 1293

IDownloadResourceProvider::OpenForRead

Opens a stream to download the resource to a local file.

C++ Syntax

```
HRESULT OpenForRead(
  BSTR lpszResourceID,  BSTR lpszParam,  ISequentialStream**
pStream);
```

Parameters

lpszResourceID	[in] Resource identification. A format of this string is resource provider's specific one.
lpszParam	[in] A string with resource provider's specific data.
pStream	[out, retval] Returns a stream that will be used to download the resource to a local file.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_INVALIDARG	An invalid parameter passed.
E_FAIL	An unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CacheMgr.dll"

See Also

"IDownloadResourceProvider" on page 1293

IDownloadResourceProvider::GetLastError

The method returns a last error text from the provider.

C++ Syntax

```
HRESULT GetLastError(
  BSTR* lpszErrorText);
```

Parameters

lpszErrorText	[out, retval] Returns a last error text from the provider.

Return Values

S_OK	The operation succeeded.
E_POINTER	The parameter cannot be NULL.
E_INVALIDARG	An invalid parameter passed.
E_FAIL	An unexpected error occurs.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CacheMgr.dll"

See Also

"IDownloadResourceProvider" on page 1293

IDownloadSession

An active download session.

Methods

IDownloadSession Methods	Description
IDownloadSession::Abort	Aborts a download.
IDownloadSession::Suspend	Suspends a download.
IDownloadSession::Resume	Resumes the suspended download.
IDownloadSession::AddSink	Registers the new sink that will receive notifications.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CacheMgr.dll"

IDownloadSession::Abort

Aborts a download.

C++ Syntax

```
HRESULT Abort(
  HRESULT hrCause);
```

Parameters

hrCause	[in] A result of aborting a transfer.

Return Values

S_OK	The operation succeeded.
E_FAIL	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CacheMgr.dll"

See Also

"IDownloadSession" on page 1299

IDownloadSession::Suspend

Suspends a download.

C++ Syntax

```
HRESULT Suspend();
```

Parameters

None.

Return Values

S_OK	The operation succeeded.
E_FAIL	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CacheMgr.dll"

See Also

"IDownloadSession" on page 1299

IDownloadSession::Resume

Resumes a suspended download.

C++ Syntax

```
HRESULT Resume();
```

Parameters

None.

Return Values

S_OK	The operation succeeded.
E_FAIL	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CacheMgr.dll"

See Also

"IDownloadSession" on page 1299

IDownloadSession::AddSink

Registers the new sink that will receive notifications.

C++ Syntax

```
HRESULT AddSink(
  IDownloadSink* pSink);
```

Parameters

pSink	[in] The IDownloadSink object.

Return Values

S_OK	The operation succeeded.
E_FAIL	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CacheMgr.dll"

See Also

"IDownloadSession" on page 1299

"IDownloadSink" on page 1302

IDownloadSink

Downloads/uploads the progress sink.

When to Implement

Implemented by a client.

When to Use

Called by the download/upload manager module.

Methods

IDownloadSink Methods	Description
IDownloadSink::OnBegin	Called when downloading/uploading begins.
IDownloadSink::OnConnect	Called just after connection to the resource provider.
IDownloadSink::OnComplete	Called when downloading/uploading completes successfully.
IDownloadSink::OnAbort	Called when downloading/uploading is aborted.
IDownloadSink::OnProgress	Called when new data arrives.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CacheMgr.dll"

IDownloadSink::OnBegin

Called when downloading/uploading begins.

C++ Syntax

```
HRESULT OnBegin(
  IDownloadSession* pSession);
```

Parameters

pSession	[in] The IDownloadSession object.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_INVALIDARG	An invalid argument passed.
E_FAIL	The operation failed.

Remarks

Called when downloading/uploading begins. A client can store a session object pointer with AddRef(). In this case it will release it. A client can use a session object to control downloading/uploading.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CacheMgr.dll"

See Also

"IDownloadSink" on page 1302

"IDownloadSession" on page 1299

IDownloadSink::OnConnect

Called just after connection to the resource provider.

C++ Syntax

```
HRESULT OnConnect(
  DWORD dwTotalAmount);
```

Parameters

dwTotalAmount	[in] Resource's file size.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_INVALIDARG	An invalid argument passed.
E_FAIL	The operation failed.

Remarks

A resource size is passed if the provider supports it, else DWORD(-1) is passed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CacheMgr.dll"

See Also

"IDownloadSink" on page 1302

IDownloadSink::OnComplete

Called when downloading/uploading completes successfully.

C++ Syntax

```
HRESULT OnComplete(
  BSTR lpszLocalFileName);
```

Parameters

lpszLocalFileName	[in] A full path to the resource local file.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_INVALIDARG	An invalid argument passed.
E_FAIL	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CacheMgr.dll"

See Also

"IDownloadSink" on page 1302

IDownloadSink::OnAbort

Called when downloading/uploading is aborted.

C++ Syntax

```
HRESULT OnAbort(
  HRESULT hrCause,  BSTR lpszErrorText);
```

Parameters

hrCause	[in] A reason of aborting a transfer.
lpszErrorText	[in] Description of the abort reason.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_INVALIDARG	An invalid argument passed.
E_FAIL	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CacheMgr.dll"

See Also

"IDownloadSink" on page 1302

IDownloadSink::OnProgress

Called when new data arrives.

C++ Syntax

```
HRESULT OnProgress(
  DWORD dwTotalAmount,  DWORD dwCurrentAmount);
```

Parameters

dwTotalAmount	[in] Resource's total file size.
dwCurrentAmount	[in] The number of read bytes.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_INVALIDARG	An invalid argument passed.
E_FAIL	The operation failed.

Remarks

A total amount will be zero if size information is not available from the resource provider.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CacheMgr.dll"

See Also

"IDownloadSink" on page 1302

ILocalFileCacheManager

Local file cache manager.

When to Implement

Implemented by the download/upload manager module.

Methods

ILocalFileCacheManager Methods	Description
ILocalFileCacheManager::IsLocalFile	Is there a local file for the given resource ID?
ILocalFileCacheManager::GetLocalFile	Gets the valid local file name for a resource.
ILocalFileCacheManager::DownloadToFile	Downloads a resource to its permanent storage to the given filename.
ILocalFileCacheManager::Upload	Uploads a resource to its permanent storage.
ILocalFileCacheManager::UploadFromFile	Uploads a resource to its permanent storage from the given file.
ILocalFileCacheManager::Purge	Purges a cache.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CacheMgr.dll"

ILocalFileCacheManager::IsLocalFile

Is there a local file for the given resource ID?

C++ Syntax

```
HRESULT IsLocalFile(
  REFCLSID clsidResourceProvider,  BSTR lpszResourceID,  BSTR
lpszParam,  BSTR* pszLocalFileName);
```

Parameters

clsidResourceProvider	[in] CLSID of the resource provider.
lpszResourceID	[in] Resource identification. A format of this string is resource provider's specific one.
lpszParam	[in] A string with resource provider's specific data.
pszLocalFileName	[out, retval] Returns a full path to resource's local file.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_INVALIDARG	An invalid argument passed.
E_FAIL	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CacheMgr.dll"

See Also

"ILocalFileCacheManager" on page 1306

ILocalFileCacheManager::GetLocalFile

Gets a valid local file name for the resource.

C++ Syntax

```
HRESULT GetLocalFile(
  REFCLSID clsidResourceProvider,  BSTR lpszResourceID,  BSTR
lpszParam,  IDownloadSink* pSink,  BSTR* pszLocalFileName);
```

Parameters

clsidResourceProvider	[in] CLSID of the resource provider.
lpszResourceID	[in] Resource identification. A format of this string is resource provider's specific one.
lpszParam	[in] A string with resource provider's specific data.
pSink	[in] The IDownloadSink object that will receive notifications.
pszLocalFileName	[out, retval] A full path to resource's local file.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_INVALIDARG	A invalid argument passed.
E_FAIL	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CacheMgr.dll"

See Also

"ILocalFileCacheManager" on page 1306

"IDownloadSink" on page 1302

ILocalFileCacheManager::DownloadToFile

Downloads a resource to its permanent storage to the given filename.

C++ Syntax

```
HRESULT DownloadToFile(

  REFCLSID clsidResourceProvider,  BSTR lpszResourceID,  BSTR
lpszParam,  IDownloadSink* pSink,  BSTR pszLocalFileName);
```

Parameters

clsidResourceProvider	[in] CLSID of the resource provider.
lpszResourceID	[in] Resource identification. A format of this string is resource provider's specific one.
lpszParam	[in] A string with resource provider's specific data.
pSink	[in] The IDownloadSink object that will receive notifications.
pszLocalFileName	[in] A full path to resource's local file.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_INVALIDARG	An invalid argument passed.
E_FAIL	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CacheMgr.dll"

See Also

"ILocalFileCacheManager" on page 1306

"IDownloadSink" on page 1302

ILocalFileCacheManager::Upload

Uploads the resource to its permanent storage.

C++ Syntax

```
HRESULT Upload(

  REFCLSID clsidResourceProvider,  BSTR lpszResourceID,  BSTR
lpszParam,  IDownloadSink* pSink);
```

Parameters

clsidResourceProvider	[in] CLSID of the resource provider.
lpszResourceID	[in] Resource identification. A format of this string is resource provider's specific one.
lpszParam	[in] A string with resource provider's specific data.
pSink	[in] The IDownloadSink object that will receive notifications.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_INVALIDARG	An invalid argument passed.
E_FAIL	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CacheMgr.dll"

See Also

"ILocalFileCacheManager" on page 1306

"IDownloadSink" on page 1302

ILocalFileCacheManager::UploadFromFile

Uploads a resource to its permanent storage from the given file.

C++ Syntax

```
HRESULT UploadFromFile(
  REFCLSID clsidResourceProvider,  BSTR lpszResourceID,  BSTR
lpszParam,  IDownloadSink* pSink,  BSTR lpszFileName);
```

Parameters

clsidResourceProvider	[in] CLSID of the resource provider.
lpszResourceID	[in] Resource identification. A format of this string is resource provider's specific one.
lpszParam	[in] A string with resource provider's specific data.
pSink	[in] IDownloadSink object that will receive notifications.
lpszFileName	[in] A full path to file.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_INVALIDARG	An invalid argument passed.
E_FAIL	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CacheMgr.dll"

See Also

"ILocalFileCacheManager" on page 1306

"IDownloadSink" on page 1302

ILocalFileCacheManager::Purge

Purges a cache.

C++ Syntax

```
HRESULT Purge(
  DATE dOlderThan,  DWORD* lBytesFreed);
```

Parameters

dOlderThan	[in] A number of days for resources cache purging.
lBytesFreed	[out, retval] A number of bytes freed.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_INVALIDARG	An invalid argument passed.
E_FAIL	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CacheMgr.dll"

See Also

"ILocalFileCacheManager" on page 1306

IUploadResourceProvider

A resource provider with upload capabilities.

When to Implement

Implemented by a resource provider object as the additional interface to IDownloadResourceProvider.

When to Use

Can be obtain via QueryInterface on IDownloadResourceProvider. Called by the upload manager.

Methods

IUploadResourceManager Methods	Description
IUploadResourceProvider::OpenForWrite	Opens a stream to upload the resource from a local file.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CacheMgr.dll"

See Also

"IDownloadResourceProvider" on page 1293

IUploadResourceProvider::OpenForWrite

Opens a stream to upload the resource from a local file.

C++ Syntax

```
HRESULT OpenForWrite(
  BSTR lpszResourceID,  BSTR lpszParam,  ISequentialStream**
pStream);
```

Parameters

lpszResourceID	[in] Resource identification. A format of this string is resource provider's specific one.
lpszParam	[in] A string with resource provider's specific data.
pStream	[out, retval] Returns a stream object that will be used to write the resource from a local file.

Return Values

S_OK	The operation succeeded.
E_POINTER	A parameter cannot be NULL.
E_INVALIDARG	An invalid argument passed.
E_FAIL	The operation failed.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.5.1 (or later).

Header: #import "CacheMgr.dll"

See Also

"IUploadResourceProvider" on page 1313

Chapter

14

Attache Plug-in

In This Chapter

- "About Attache Plug-in" on page 1316
- "Attache Interfaces" on page 1317

About Attache Plug-in

Attache is a new feature of the DM product that helps the user to work with DM documents while being disconnected from the office network. In the usual mode (online mode) when the network is available the user can select the documents, which he/she will work later with in the offline mode. While startup DM Extensions will automatically detect whether or not the DM Server is available and prompt the user to work in the offline mode if the attempt to connect to the server failed.

The product also supports "server-down" functionality. If DM Extensions detects that the server is unavailable while the user works in the network mode, it automatically moves to the unplugged mode.

If the user logs into online DM successfully, he/she will be able to move the undocked documents back to the DM database. The user will be prompted to do it automatically if DM Extensions detects that there exist such documents.

Attache module is implemented as a plug-in. It also implemented as a notification service to track document changes in online mode.

To support the Attache functionality the following interfaces are implemented:

- IAttacheRoot,
- IUndockedProfile,
- IUndockedProfiles,
- IUndockedVersion,
- IUndockedVersions.

Attache Interfaces

- "IAttacheRoot" on page 1317
- "IUndockedProfile" on page 1323
- "IUndockedProfiles" on page 1335
- "IUndockedVersion" on page 1339
- "IUndockedVersions" on page 1344

IAttacheRoot

The interface exposes a root object's functionality.

Methods

IAttacheRoot Methods	Description
IAttacheRoot::AddDocuments	Adds documents to an Attache.
IAttacheRoot::RemoveDocuments	Removes documents from an Attache.
IAttacheRoot::AddVersions	Adds document's versions to an Attache.
IAttacheRoot::ShowImportForm	Shows an import form.

Properties

IAttacheRoot Properties	Description
IAttacheRoot::AttacheDocuments	The property returns a collection of Attache documents.
IAttacheRoot::ShadowDocuments	The property returns a collection of shadowed documents.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"Attache Plug-in" on page 1315

IAttacheRoot::AddDocuments

The method adds documents to an Attache.

C++ Syntax

```
HRESULT AddDocuments(  IDocProfiles* pDocs,  long hWnd,
VARIANT_BOOL* pbSuccess);
```

Visual Basic Syntax

```
[ pbSuccess = ] AttacheRoot.AddDocuments (pDocs, hWnd)
```

Parameters

pDocs	[in] Specifies a document collection, which should be added to an Attache.
hWnd	[in] Specifies the parent window for the "Lock documents" dialog and "Choose versions" window. May be NULL.
pbSuccess	[out, retval] Specifies an address where a success flag should be returned.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

If returned value is S_FALSE methods of IOMErrInfo should be used for error description.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"IAttacheRoot" on page 1317

"IDocProfiles" on page 292

"IOMErrInfo" on page 412

IAttacheRoot::AddVersions

The method adds document's versions to an Attache.

C++ Syntax

```
HRESULT AddVersions(  VARIANT_BOOL bLock,  IVersions* pVersions,
VARIANT_BOOL* pbSuccess);
```

Visual Basic Syntax

```
[ pbSuccess = ] AttacheRoot.AddVersions (bLock, pVersions)
```

Parameters

bLock	[in] Specifies whether the document should be locked.
pVersions	[in] Specifies a version collection, which should be added to an Attache.
pbSuccess	[out, retval] Specifies an address where the success flag should be returned.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

If returned value is S_FALSE methods of IOMErrInfo should be used for error description.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"IAttacheRoot" on page 1317

"IVersions" on page 673

"IOMErrInfo" on page 412

IAttacheRoot::AttacheDocuments

The method returns a collection of Attache's documents.

C++ Syntax

```
HRESULT AttacheDocuments(  IUndockedProfiles** ppDocs);
```

Visual Basic Syntax

```
[ Set ppDocs = ] AttacheRoot.AttacheDocuments
```

Parameters

ppDocs	[out, retval] Specifies an address where a pointer to the IUndockedProfiles interface of an UndockedProfiles object should be returned.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access

Read Only

Remarks

If returned value is S_FALSE methods of IOMErrInfo should be used for error description.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"IAttacheRoot" on page 1317

"IUndockedProfiles" on page 1335

"IOMErrInfo" on page 412

IAttacheRoot::RemoveDocuments

The method removes documents from an Attache.

C++ Syntax

```
HRESULT RemoveDocuments(  short sFlag,  IDocProfiles* pDocs,  long
hWnd,  VARIANT_BOOL* pbSuccess);
```

Visual Basic Syntax

```
[ pbSuccess = ] AttacheRoot.RemoveDocuments (sFlag, pDocs, hWnd)
```

Parameters

sFlag	[in] Should be one of MethodsFlags enumerator values. Specifies operation's options. If OF_NOWARNING, confirmations dialogs will not be displayed. If OF_NOERRORS, errors messages will not be displayed.
pDocs	[in] Specifies a document collection, which should be removed from an Attache.
hWnd	[in] Specifies the parent window for confirmation dialogs. May be NULL.
pbSuccess	[out, retval] Specifies an address where the success flag should be returned.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

If returned value is S_FALSE methods of IOMErrInfo should be used for error description.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"IAttacheRoot" on page 1317

"IDocProfiles" on page 292

"IOMErrInfo" on page 412

"MethodsFlags Enumeration" on page 689

IAttacheRoot::ShadowDocuments

The method returns a collection of shadowed documents.

C++ Syntax

```
HRESULT ShadowDocuments(  IUndockedProfiles** ppDocs);
```

Visual Basic Syntax

```
[ Set ppDocs = ] AttacheRoot.ShadowDocuments
```

Parameters

ppDocs	[out, retval] Specifies an address where a pointer to the IUndockedProfiles interface of the UndockedProfiles object should be returned.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access

Read Only

Remarks

If returned value is S_FALSE methods of IOMErrInfo should be used for error description.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"IAttacheRoot" on page 1317

"IOMErrInfo" on page 412

"IUndockedProfiles" on page 1335

IAttacheRoot::ShowImportForm

The method shows an import form.

C++ Syntax

```
HRESULT ShowImportForm(  long hWnd,  short* psResult);
```

Visual Basic Syntax

```
[ psResult = ] AttacheRoot.ShowImportForm (hWnd)
```

Parameters

hWnd	[in] Specifies the parent window. May be NULL.
psResult	[out, retval] Specifies an address where a result should be returned. The result can contain one of the FormsResults enumerator values.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Remarks

If returned value is S_FALSE methods of IOMErrInfo should be used for error description.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"IAttacheRoot" on page 1317

"IOMErrInfo" on page 412

"FormsResults Enumeration" on page 686

IUndockedProfile

The interface exposed an undocked profile object's functionality.

Properties

IUndockedProfile Properties	Description
LibraryName	Returns the offline library name of a document.
DocNumber	Returns the internal offline number of a document.
Name	Returns the name of a document.
Icon	Returns the document icon.
AppID	Returns the native application id for a document.
Versions	Returns a collection of all undocked versions of the document.
Author	Returns the document author.
IsModified	Returns VARIANT_TRUE if the document is modified.
IsLocked	Returns VARIANT_TRUE if the document is locked.
IsAttacheDocument	Returns VARIANT_TRUE if the document is Attache's document or VARIANT_FALSE if the document is a shadowed document.
IsBounded	Returns VARIANT_TRUE, if the document is bounded to the network document.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"Attache Plug-in" on page 1315

IUndockedProfile::AppID

Returns the native application id for a document.

C++ Syntax

```
HRESULT AppID(  BSTR* pbstrAppID);
```

Visual Basic Syntax

```
[ pbstrAppID = ] UndockedProfile.AppID
```

Parameters

pbstrAppID	[out, retval] Specifies an address where an application's identifier should be returned.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access

Read Only

Remarks

If returned value is S_FALSE methods of IOMErrInfo should be used for error description.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"IUndockedProfile" on page 1323

"IOMErrInfo" on page 412

IUndockedProfile::Author

Returns the document author.

C++ Syntax

```
HRESULT Author(  BSTR* pbstrAuthor);
```

Visual Basic Syntax

```
[ pbstrAuthor = ] UndockedProfile.Author
```

Parameters

pbstrAuthor	[out, retval] Specifies an address where an author identifier should be returned.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access

Read Only

Remarks

If returned value is S_FALSE methods of IOMErrInfo should be used for error description.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"IUndockedProfile" on page 1323

"IOMErrInfo" on page 412

IUndockedProfile::DocNumber

Returns the internal offline number of a document.

C++ Syntax

```
HRESULT DocNumber(  long* plDocNumber);
```

Visual Basic Syntax

```
[ plDocNumber = ] UndockedProfile.DocNumber
```

Parameters

plDocNumber	[out, retval] Specifies an address where the offline number of document should be returned.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access

Read Only

Remarks

If the document is not bounded to the network document (IsBounded returns VARIANT_FALSE), the error will be returned. If returned value is S_FALSE methods of IOMErrInfo should be used for error description.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"IUndockedProfile" on page 1323

"IOMErrInfo" on page 412

IUndockedProfile::Icon

Returns the icon of a document.

C++ Syntax

```
HRESULT Icon(  long* plIcon);
```

Visual Basic Syntax

```
[ plIcon = ] UndockedProfile.Icon
```

Parameters

pIIcon	[out, retval] Specifies an address where icon's identifier should be returned

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access

Read Only

Remarks

If returned value is S_FALSE methods of IOMErrInfo should be used for error description.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"IUndockedProfile" on page 1323

"IOMErrInfo" on page 412

IUndockedProfile::IsAttacheDocument

Returns VARIANT_TRUE if the document is Attache's document or VARIANT_FALSE if the document is a shadowed document.

C++ Syntax

```
HRESULT IsAttacheDocument(  VARIANT_BOOL* pbResult);
```

Visual Basic Syntax

```
[ pbResult = ] UndockedProfile.IsAttacheDocument
```

Parameters

pbResult	[out, retval] Specifies an address where the result flag should be returned.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access

Read Only

Remarks

If returned value is S_FALSE methods of IOMErrInfo should be used for error description.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"IUndockedProfile" on page 1323

"IOMErrInfo" on page 412

IUndockedProfile::IsBounded

Returns VARIANT_TRUE, if the document is bounded to the network document.

C++ Syntax

```
HRESULT IsBounded(  VARIANT_BOOL* pbResult);
```

Visual Basic Syntax

```
[ pbResult = ] UndockedProfile.IsBounded
```

Parameters

pbResult	[out, retval] Specifies an address where the result flag should be returned.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access

Read Only

Remarks

If returned value is S_FALSE methods of IOMErrInfo should be used for error description.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"IUndockedProfile" on page 1323

"IOMErrInfo" on page 412

IUndockedProfile::IsLocked

Returns VARIANT_TRUE if the document is locked when adding to an Attache.

C++ Syntax

```
HRESULT IsLocked(  VARIANT_BOOL* pbResult);
```

Visual Basic Syntax

```
[ pbResult = ] UndockedProfile.IsLocked
```

Parameters

pbResult	[out, retval] Specifies an address where the lock flag should be returned.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access

Read Only

Remarks

If returned value is S_FALSE methods of IOMErrInfo should be used for error description.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"IUndockedProfile" on page 1323

"IOMErrInfo" on page 412

IUndockedProfile::IsModified

Returns VARIANT_TRUE if the document is modified in the offline mode.

C++ Syntax

```
HRESULT IsModified(  VARIANT_BOOL* pbResult);
```

Visual Basic Syntax

```
[ pbResult = ] UndockedProfile.IsModified
```

Parameters

pbResult	[out, retval] Specifies an address where the modification flag should be returned.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access

Read Only

Remarks

If returned value is S_FALSE methods of IOMErrInfo should be used for error description.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"IUndockedProfile" on page 1323

"IOMErrInfo" on page 412

IUndockedProfile::LibraryName

Returns the offline library name of a document.

C++ Syntax

```
HRESULT LibraryName(  BSTR* pbstrLibraryName);
```

Visual Basic Syntax

```
[ pbstrLibraryName = ] UndockedProfile.LibraryName
```

Parameters

pbstrLibraryName	[out, retval] Specifies an address where the offline library name should be returned.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access

Read Only

Remarks

If the document is not bounded to the network document (IsBounded returns VARIANT_FALSE), the error will be returned. If returned value is S_FALSE methods of IOMErrInfo should be used for error description.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"IUndockedProfile" on page 1323

"IOMErrInfo" on page 412

IUndockedProfile::Name

Returns the name of a document.

C++ Syntax

```
HRESULT Name(  BSTR* pbstrName);
```

Visual Basic Syntax

```
[ pbstrName = ] UndockedProfile.Name
```

Parameters

pbstrName	[out, retval] Specifies an address where the name of a document should be returned.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access

Read Only

Remarks

If returned value is S_FALSE methods of IOMErrInfo should be used for error description.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"IUndockedProfile" on page 1323

"IOMErrInfo" on page 412

IUndockedProfile::Versions

Returns a collection of all undocked versions of the document.

C++ Syntax

```
HRESULT Versions(  IUndockedVersions** ppVersions);
```

Visual Basic Syntax

```
[ Set ppVersions = ] UndockedProfile.Versions
```

Parameters

ppVersions	[out, retval] Specifies an address where a pointer to the IUndockedVersions interface of a version collection should be returned.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access

Read Only

Remarks

If returned value is S_FALSE methods of IOMErrInfo should be used for error description.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"IUndockedProfile" on page 1323

"IUndockedVersions" on page 1344

"IOMErrInfo" on page 412

IUndockedProfiles

The interface exposed undocked document's profiles collection functionality.

Properties

IUndockedProfiles Properties	Description
Item	Returns the specific item of a collection either by position or by key.
Count	Returns the number of items in a collection.
_NewEnum	Creates a new enumerator object.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"Attache Plug-in" on page 1315

IUndockedProfiles::Count

Returns the number of items in a collection.

C++ Syntax

```
HRESULT Count(  long* plCount);
```

Visual Basic Syntax

```
[ plCount = ] UndockedProfiles.Count
```

Parameters

plCount	[out, retval] Specifies an address where the number of items should be returned.

Return Values

This method supports the standard return values E_INVALIDARG, E_OUTOFMEMORY, and E_UNEXPECTED as well the following:

S_OK	The operation succeeded.

Access

Read Only

Remarks

If a returned value is S_OK the operation succeeded.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"IUndockedProfiles" on page 1335

IUndockedProfiles::Item

Returns the specific item of a collection either by position or by key.

C++ Syntax

```
HRESULT Item(  VARIANT vIndex,  IUndockedProfile** ppProfile);
```

Visual Basic Syntax

```
[ Set ppProfile = ] UndockedProfiles.Item (vIndex)
```

Parameters

vIndex	[in] Specifies a position or key of the item in a collection.
ppProfile	[out, retval] Specifies an address where a pointer to the IUndockedProfile interface of a document should be returned.

Return Values

This method supports the standard return values E_INVALIDARG, E_OUTOFMEMORY, and E_UNEXPECTED as well the following:

S_OK	The operation succeeded.

Access

Read Only

Remarks

If a returned value is S_OK the operation succeeded.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"IUndockedProfiles" on page 1335

"IUndockedProfile" on page 1323

IUndockedProfiles::_NewEnum

Creates a new enumerator object.

C++ Syntax

```
HRESULT _NewEnum(  LPUNKNOWN* ppEnum);
```

Parameters

ppEnum	[out, retval] Specifies an address where a pointer to the IUnkown interface of a new enumerator should be returned.

Return Values

This method supports the standard return values E_INVALIDARG, E_OUTOFMEMORY, and E_UNEXPECTED as well the following:

S_OK	The operation succeeded.

Access

Read Only

Remarks

If returned value is S_OK the operation succeeded.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"IUndockedProfiles" on page 1335

IUndockedVersion

The interface exposed undocked document version object's functionality.

Properties

IUndockedVersion Properties	Description
Author	Returns the version author.
Typist	Returns the version typist.
VersionLabel	Returns the version label.
VersionID	Returns the version identifier.
UndockedProfile	Returns the parent document of a version.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"Attache Plug-in" on page 1315

IUndockedVersion::Author

Returns the document author.

C++ Syntax

```
HRESULT Author(  BSTR* pbstrAuthor);
```

Visual Basic Syntax

```
[ pbstrAuthor = ] UndockedVersion.Author
```

Parameters

pbstrAuthor	[out, retval] Specifies an address where an author identifier should be returned.

Return Values

S_OK

The operation succeeded.

S_FALSE

The operation failed.

Access

Read Only

Remarks

If returned value is S_FALSE methods of IOMErrInfo should be used for error description.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"IUndockedVersion" on page 1339

"IOMErrInfo" on page 412

IUndockedVersion::Typist

Returns the version typist.

C++ Syntax

```
HRESULT Typist(  BSTR* pbstrTypist);
```

Visual Basic Syntax

```
[ pbstrTypist = ] UndockedVersion.Typist
```

Parameters

pbstrTypist	[out, retval] Specifies an address where a typist identifier should be returned.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access

Read Only

Remarks

If returned value is S_FALSE methods of IOMErrInfo should be used for error description.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"IUndockedVersion" on page 1339

"IOMErrInfo" on page 412

IUndockedVersion::UndockedProfile

Returns the parent document of a version.

C++ Syntax

```
HRESULT UndockedProfile(  IUndockedProfile** ppProfile);
```

Visual Basic Syntax

```
[ Set ppProfile = ] UndockedVersion.UndockedProfile
```

Parameters

ppProfile	[out, retval] Specifies an address where a pointer to the IUndockedProfile interface of a document should be returned.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access

Read Only

Remarks

If returned value is S_FALSE methods of IOMErrInfo should be used for error description.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"IUndockedVersion" on page 1339

"IUndockedProfile" on page 1323

"IOMErrInfo" on page 412

IUndockedVersion::VersionID

Returns the version's identifier.

C++ Syntax

```
HRESULT VersionID(  BSTR* pbstrID);
```

Visual Basic Syntax

```
[ pbstrID = ] UndockedVersion.VersionID
```

Parameters

pbstrID	[out, retval] Specifies an address where version's identifier should be returned.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access

Read Only

Remarks

If the document is not bound to the network document (IUndockedProfile::IsBounded returns VARIANT_FALSE), the error will be returned.

If returned value is S_FALSE methods of IOMErrInfo should be used for error description.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"IUndockedVersion" on page 1339

"IUndockedProfile" on page 1323

"IOMErrInfo" on page 412

IUndockedVersion::VersionLabel

Returns the version label.

C++ Syntax

```
HRESULT VersionLabel(  BSTR* pbstrVersionLabel);
```

Visual Basic Syntax

```
[ pbstrVersionLabel = ] UndockedVersion.VersionLabel
```

Parameters

pbstrVersionLabel	[out, retval] Specifies an address where version's label should be returned.

Return Values

S_OK	The operation succeeded.
S_FALSE	The operation failed.

Access

Read Only

Remarks

If returned value is S_FALSE methods of IOMErrInfo should be used for error description.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"IUndockedVersion" on page 1339

"IOMErrInfo" on page 412

IUndockedVersions

Interface exposed undocked document's versions collection functionality.

Properties

IUndockedVersions Properties	Description
Item	Returns the specific item of a collection either by position or by key.
Count	Returns the number of items in a collection.
_NewEnum	Creates a new enumerator object.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"Attache Plug-in" on page 1315

IUndockedVersions::Count

Returns the number of items in a collection.

C++ Syntax

```
HRESULT Count(  long* plCount);
```

Visual Basic Syntax

```
[ plCount = ] UndockedVersions.Count
```

Parameters

plCount	[out, retval] Specifies an address where the number of items should be returned.

Return Values

This method supports the standard return values E_INVALIDARG, E_OUTOFMEMORY, and E_UNEXPECTED as well the following:

S_OK	The operation succeeded.

Access

Read Only

Remarks

If returned value is S_OK the operation succeeded.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"IUndockedVersions" on page 1344

IUndockedVersions::Item

Returns the specific item of a collection either by position or by key.

C++ Syntax

```
HRESULT Item(  VARIANT vIndex,  IUndockedVersion** ppVersion);
```

Visual Basic Syntax

```
[ Set ppVersion = ] UndockedVersions.Item (vIndex)
```

Parameters

vIndex	[in] Specifies a position or key of an item in collection.
ppVersion	[out, retval] Specifies an address where a pointer to the IUndockedVersion interface of document's version should be returned.

Return Values

This method supports the standard return values E_INVALIDARG, E_OUTOFMEMORY, and E_UNEXPECTED as well the following:

S_OK	The operation succeeded.

Access

Read Only

Remarks

If returned value is S_OK the operation succeeded.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"IUndockedVersions" on page 1344

"IUndockedVersion" on page 1339

IUndockedVersions::_NewEnum

Creates a new enumerator object.

C++ Syntax

```
HRESULT _NewEnum(  LPUNKNOWN* ppEnum);
```

Parameters

ppEnum	[out, retval] Specifies an address where a pointer to the IUnkown interface of a new enumerator should be returned.

Return Values

This method supports the standard return values E_INVALIDARG, E_OUTOFMEMORY, and E_UNEXPECTED as well the following:

S_OK	The operation succeeded.

Access

Read Only

Remarks

If returned value is S_OK the operation succeeded.

Requirements

DM: Requires DM Extensions 5.0 or PowerDOCS 3.9 (or later).

Header: #import "UnPlugin.dll"

See Also

"IUndockedVersions" on page 1344

Appendix

A

DM 5.0 Registry Settings

In This Appendix

- "Application Integration Settings" on page 1350
- "Attachè related registry settings" on page 1364
- "General registry settings" on page 1366
- "Custom Forms Registry Settings" on page 1372
- "DM plug-in Settings" on page 1373
- "E-Mail Integration related registry settings" on page 1390
- "MS Outlook DM Extension Registry Settings" on page 1394
- "Installation related registry settings" on page 1395

- “User Interface related registry settings” on page 1396

Application Integration Settings

- “Common Settings” on page 1350
- “Active (ODMA) Integration related registry settings” on page 1353
- “Interceptor related registry settings” on page 1355
- “Passive Integration related registry settings” on page 1361
- “Linking Registry Settings” on page 1363

Common Settings

HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ AppIntegration			
	AdvancedPathSearch	DWORD	1 - advanced path search is enabled; 0 – search application only in folders specified by PATH environment variable (default).
	StartPrintJobTimeout	DWORD	The timeout before the printed document is skipped, if the printing job is not started (msec). By default – 15000 msec.
	LaunchProcessTimeout	DWORD	Idle process timeout (msec). By default – 20000 msec.
	DDEOpenRetry	DWORD	Maximum number of DDE transaction retry when document is opened. By default – 10.
	DDEOpenTimeout	DWORD	Timeout for each DDE transaction when document is opened in milliseconds. By default – 30000 (30 sec).
	DDEPrintRetry	DWORD	Maximum number of DDE transaction retry when document is printed. By default – 1.

	DDEPrintTimeout	DWORD	Timeout for each DDE transaction when document is printed in milliseconds. By default – 3600000 (1 hour).
	DisableNativeDialogs	DWORD	0 – enable native dialogs in ODMA and Interceptor monitored applications; 1 – disable native dialogs.
	TrackChars	DWORD	1 – Only characters will be accounted in Cost Recovery; 0 – all keys will be accounted in Cost Recovery (default).
	EnableFullTypingTime	DWORD	Nn - seconds, the time before the Typing time did not accounted in Cost Recovery until new key will be pressed. 0 (default) - will be accounted for full Typing Time.
<<Integration Scheme Name>>	(default)		Integration scheme CLSID
	Disable	DWORD	0 – integration scheme is enabled; 1 – integration scheme is disabled.
	InsertAt	String	Character. Sets the order of integration schemes polling.
	Description	String	Scheme description. Used by Administration Tool
DefExtensions	Application ID	String	Default extension for the application. This value is used if no default extension for given application is specified in the database.

Compatible	<<Application process name>>	DWORD	Application flags: 0x01 – always return success code when printing a document; 0x02 - close application after printing; 0x04 - disable message filter for application in ODMA; 0x08 - show title instead name in ODMA; 0x10 - return empty type in ODMA; 0x20 - application always run as single instance; 0x40 - Use short path instead of long for printing; 0x80 - GetActiveWindow() is used to receive the parent window; 0x100 - Ignore command line settings for the running application; 0x200 - check unsaved profile when closing document; 0x400 - ignore application-defined callback; 0x800 - wait finish of printing of the previous document, before starting new printing; 0x1000 - do not change APP_ID when open; 0x2000 - file compare method: detect changing file size and last write date\time; 0x4000 - file compare method: detect changing file archive attribute; 0x8000 - file compare method: detect changing file CRC.
FrontEnd	<<Application ID>>	DWORD	1 – front end profiling will not be used when application is launched from “Applications” node. 0 – front end profiling will not be used for this application.

	Enable Native Run	DWORD	0 (default) - user cannot continue running application, if application does not support native template and front end profiling is enabled for application. 1 – User can launch application without front end profiling.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ AppIntegration			
	DisableNativeDialogs	DWORD	1 – disable native dialogs; 0 – enable native dialogs (default).
HKEY_CURRENT_USER\ Software\ Microsoft\ Office\ 8.0\			
Word\Options	EnableMacroVirusProtection	String	"0" This value disables macro virus protection for Word.
Excel\Microsoft Excel	Options6	DWORD	"0" This value disables macro virus protection for Excel.

Active (ODMA) Integration related registry settings

HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ AppIntegration\ ODMA			
	DisableAutoUpdate	DWORD	Indicates whether or not the ODMA ISM should register the DM ODMA DMS module as a default DMS in the IIntegrationScheme::Init method as follows: 0 – (default) module should be registered; 1 – module should not be registered.
	OpenTimeOut	DWORD	The timeout in seconds that specifies in what time a document will be unlocked if the application does not confirm that the document is opened. By default, this value is 60.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ AppIntegration\ ODMA			

	CheckOutPath	String	Check-out path used in ODMA applications.
HKEY_CLASSES_ROOT\ ODMA32			
	(default)	String	Contains the string: Open Document Management API (32 bit)
PCDOCS	(default)	String	Path to file ODMA DMS (DOCS_DMS.DLL prefixed by the DM install directory).
PCDOCS\DEFAULT			If presented ODMA is enabled for the all applications supporting it. If not, the key HKEY_CURRENT_USER\ SOFTWARE\ Classes\ <AppID>\ ODMA32 or HKEY_CLASSES_ROOT\ <AppID>\ ODMA32 with default value "PCDOCS" should be created for each application.
PCDOCS\RESET	(default)	String	If AutoUpdate value is set 1 then key PCDOCS\DEFAULT is created by default. The key PCDOCS\RESET prevents from automatical creation of the key PCDOCS\DEFAULT.
PCDOCS\FullName	(default)	String	Contains ODMA DMS name: "PCDOCS DMS".
PCDOCS\Log	EnableLog	String	Path to LOG-file.
HKEY_CLASSES_ROOT			
MS Word\ODMA32	(default)	String	"" This value disables ODMA native support for Word.
MS Excel\ODMA32	(default)	String	"PCDOCS" This value enables ODMA support for macros in Excel.
PRJ40\ODMA32	(default)	String	"PCDOCS" This value enables ODMA support for macros in Project.
MS VB\ODMA32	(Default)	String	
HKEY_LOCAL_MACHINE\ SOFTWARE\ Microsoft\ ODMA32\ ANSIOnNT			
SubKey	**Value Name**	**Type**	**Value**

	(default)		This key enables API calls with ANSI string parameters.

Interceptor related registry settings

HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ AppIntegration\ Interceptor			
	EnableLog	String	The full path to the Interceptor log file. If not specified (default), the log file isn't used.
	OpenTimeOut	DWORD	The timeout in seconds for the Confirm Release Document dialog. By default, this value is 30. Used if value in HKEY_CURRENT_USER is not defined.
	ShowConfirmRelease	DWORD	Indicates whether or not the Confirm Release Document dialog should be shown as follows: 0 - dialog will not be shown; 1 - dialog will be shown.. Used if value in HKEY_CURRENT_USER is not defined.
	ChangeDefaults	DWORD	1 – If user has "Edit" access to HKEY_LOCAL_MACHINE registry key he/she will be asked to save changes under HKEY_LOCAL_MACHINE. Settings will be available to all users on the computer. 0 (default) – User will not be asked to save settings as default. Settings will be always saved under HKEY_CURRENT_USER and will be available for current user only.
Components	InterceptorRes	String	Specifies the full path to the module that contains resources used by the Interceptor.

	InterceptorTuner	String	Specifies the full path to the Interceptor tuner module.
	InterceptorHooks	String	Specifies the full path to the Interceptor hook module.
Settings	DocFilters	String	A template that the Interceptor uses for parsing document identifiers. By default, this value is " %1-#%2-v%3-;%1-U%2-;%4".
	DisableCheckOLEDialogs	DWORD	Indicates whether or not monitoring of OLE dialogs is disabled as follows: 0 - (default) monitoring is enabled; 1 - monitoring is disabled.
	EditControlClasses	String	The window classes of the File Name fields in application dialogs. The class names should be separated by a semicolon. By default, this value is "Edit;RichEdit20W".
	Hook	String	The full path to the Interceptor hook module.
	ShowNativeDialog	DWORD	Indicates whether or not application native dialogs will be hidden as follows: 0 - (default) dialogs will be hidden; 1 - dialogs will be shown.
	PrintUpdateTimeout	DWORD	The timeout of checking the printer spooler in milliseconds (needs only for printers that do not support printer monitoring functions). By default, this value is 300.
	FileUpdateTimeout	DWORD	The timeout (in milliseconds) of checking whether files were modified in milliseconds. By default, this value is 0.
	DisablePrinterMonitoring	DWORD	Disabling printer monitoring in Interceptor: 1 – disable monitoring; 0 – enable monitoring (default).

	EnableLog	String	Full path to the Interceptor Core log file.
	FileCompareMethod	DWORD	Specifies how files are monitored to know when they were modified as a combination of the following values: 0x01 - monitors the file size and last modified date; 0x02 - monitors the file archive attribute; 0x04 - calculates and compares the file CRC.
	FileNotificationWaitTimeout	DWORD	Delay of change file notification processing (in milliseconds). By default – 1000 msec.
HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ AppIntegration\ Interceptor\ Settings\ LaunchMethods\ <<Library Name>>			
			The key's name is the name of the library whose launch methods are monitored. All monitored launch methods on this library should be added as subkeys in this key. There is one predefined subkey, (default), in this key. This subkey lists default settings for some launch methods.
<<Application ID;Launch method description>>	(default)		The key's name is the identifier of the launch method. Its default value is one of the Interceptor's application identifiers listed in the AppDB.
	APP_ID	String	The DM application identifier (e.g., "MS WORD") used in the database.
	Description	String	The DM application description as specified in the database.

	Enable	DWORD	Indicates whether or not monitoring the launch method is enabled as follows: 0 - monitoring is disabled; 1 - (Default) monitoring is enabled.
	Path	String	The path to the launch method's executable file.
	Partial	DWORD	0 - "Path" field contains full path to the application; 1 – "Path" field contains partial of the path.
HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ AppIntegration\ Interceptor\ Settings\ AppDB			
<<Interceptor's application key>>	(default)		The name of this key is the identifier of a monitored application. Its default value will be used as a keyword to attach a launch method to these application settings.
	Description	String	The application's description.
	IsMDI	DWORD	Specifies the application type as follows: 0 - SDI application; 1 - MDI application; 2 - "Advanced" SDI application (like MS Word 2000). If not specified, the Interceptor will automatically detect this value.
	MDIClass	String	The window class of the application's MDI client windows. If not specified, the Interceptor will automatically detect this value.
	DocWinClass	String	The window class of the windows that displays the document contents in the application. If not specified, the Interceptor will automatically detect this value.

	ConfirmMsgBox	String	The hotkey to confirm replacing a file in the message box that is displayed when the user selects an exiting file in a Save As operation. If not specified, the Interceptor will automatically detect this value.
HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ AppIntegration\ Interceptor\ Settings\ AppDB\ <<Interceptor's application key>>			
<<Dialog caption>>	(default)		The name of this key is the caption of an application's dialog in the following format: [MenuItem;]DialogCaption. The Interceptor will monitor dialogs that have the caption DialogCaption and pop up by selecting the menu item MenuItem in the application menu. The MenuItem part should specify the full menu path (including submenus) separated by '|'. For instance, File|Open. If it's missed, the dialog will be monitored independently of the selected menu.
	CancelCID	DWORD	The control ID of the dialog Cancel button. If not specified, the Interceptor will automatically detect this value.
	EditCID	DWORD	The control ID of the dialog File Name field. If not specified, the Interceptor will automatically detect this value.
	OkCID	DWORD	The control ID of the dialog OK button. If not specified, the Interceptor will automatically detect this value.

	Partial	DWORD	Indicates whether or not the DialogCaption key specifies the full dialog caption as follows: 0 - (default) key specifies the full dialog caption; 1 - key specifies only a part of the caption.
	InitScript	String	The script that sets the appropriate file format. This script will be played when the dialog appears. The value should have the following format: ID;[Tabs];[Index];[Chars]. The Interceptor plays the script as follows: - activates the field whose ID is CID; - presses the TAB key Tabs times; - if Index is specified, selects the item with index Index in the active combo box; - if Chars is specified, types the characters of Chars in the active combo box.
	AfterSetScript	String	The script that sets the appropriate file format. This script will be played after specifying a file in the dialog File Name field. The value should have the same format as InitScript.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ AppIntegration\ Interceptor			
	OpenTimeOut	DWORD	The timeout (in seconds) for opening document.
	ShowConfirmRelease	DWORD	1 - show confirm release dialog.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ AppIntegration\ Interceptor\ Settings\ LaunchMethods\ <<Library Name>>			
<<Launch method ID>>	(default)		Interceptor's application key.
	APP_ID	String	Application ID.
	Description	String	Launch method description.

	Enable	DWORD	0 - launch method is disabled by default; 1 – enabled.
	Partial	DWORD	0 - "Path" field contains full path to the application; 1 – part of the path.
	Path	String	Path to the executable module of the application.

Passive Integration related registry settings

HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ AppIntegration\ PassiveODMA			
	OpenTimeOut	DWORD	Timeout of document opening.
	Disable	DWORD	1 – Passive integration scheme will not be used when document is opened from hitlist. 0 (default) – Passive integration will be enabled.
<<Application ID>>		String	"DDE" or "OLE".
	ReleaseBeforeOpen	DWORD	0 – do not release document before opening; 1 - release document.
	CLSID	DWORD	(Used if the application is integrated via OLE Automation). The string representation of the CLSID of the application. The string must begin with an open curly brace '{' and end with a close curly brace '}'. This CLSID is used to create an IDispatch interface that has a method to run an application macro.
	RunCommand	String	(Used if the application is integrated via OLE Automation). The name of the macro that should be run to open a document.

	DISPID	DWORD	(Used if the application is integrated via OLE Automation). The DISPID of the method to run an application macro.
	Service	String	(Used if the application is integrated via DDE). The DDE service name of the application which, in most cases, is the name of the executable file without its extension.
	Topic	String	(Used if the application is integrated via DDE). The topic name to establish a DDE conversation. The topic is application-specific and must be understood by the application. For example, one topic supplied by most DDE servers is the System topic. See the application documentation for the service and topic names supported by the application.
	Open	String	(Used if the application is integrated via DDE). The DDE command to open a document inside the application.
	OpenNotRunning	String	(Used if the application is integrated via DDE). The DDE command to open a document inside the application when the application is not already running.
	Print	String	(Used if the application is integrated via DDE). The DDE command to print a document inside the application.
	PrintNotRunning		(Used if the application is integrated via DDE). The DDE command to print a document inside the application when the application is not already running.

Linking Registry Settings

HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ Settings\ Linking\ AllowLastVer			
<<AppName>>			“Link to Last version” option will be enabled for Application “AppName” if this registry key exist.
HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ Settings\ Linking\ DisableLinking			
	(Default)		“0” or empty – linking will be enabled; “1” – linking will be disabled. “smart” – smart linking will be used. Linking will be enabled when key specified by SmartKeyCode is pressed.
	SmartKeyCode	String	This value defines key for enabling/disabling smart linking. The following codes are supported: 17 – Ctrl, 16 – Shift, 18 – Alt, 144 – Num Lock, 145 – Scroll Lock, 20 – Caps Lock. Multiple keys should be delimiter by comma. Examples: “16, 17” – links will be created only if Ctrl and Shift keys are pressed during Copy operation. “16, 145, 20” – links will be created only when Caps Lock and Scroll Lock is turned on and Shift key is pressed.
<<Application ID>>			“0” or empty – linking will be enabled for the application. “smart” - smart linking will be used for the application “1” – linking will be disabled for application.

	SmartKeyCode	String	Key code for smart linking for the application.

Attachè related registry settings

HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins\ Attache			
	PingInterval	DWORD	Network ping interval in milliseconds. By default - 10 seconds.
	LogFile	String	Full path to the log file
	LogMask	DWORD	Attaché plug-in events log mask (0xFFFFFFFF – log all events).
	APILogFile	String	Full path to Attaché Plug-in API log file.
	APILogFlags	DWORD	API log file flags: 0x01 – log file enabled; 0x02 – append log to the existing file
Components	HelpFile	String	Full path to the DM Extensions help file. One path for Document Management and Attaché plug-ins.
	UnpluggedRes	String	Resource file path used by Attaché plug-in.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS \Core\ Plugins\ Attache			
	UnpluggedInfo	Binary	Contains the encoded list of <Library><User><Password > triplets for all unplugged libraries
Settings	DocumentPath	String	Path to undocked file storage.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins\ Attache\ <<Library Name>>\<<User ID>>			
	AttacheLogins	DWORD	0 - last login in network mode; More than 0 – a number of logins in Attachè mode.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins\ Attache\ <<Library Name>>\ <<User ID>>\ <<Document identifier>>			

	(Default)	String	Document name.
	AppID	String	Application identifier.
	AuthorID	String	Author identifier.
	DocNumber	String	It is equal to the source profile number for profiled documents and has zero value for new documents.
	Flags	DWORD	0 – normal document 1 – document was modified in Attachè mode. 2 – original (network) document was checked-out.
	Library	String	The source library name, from which the document was copied.
	Storage	String	Storage type of original (network) document.
<<The unique identifier of the undocked version>>	FilePath	String	Full path to the document file.
	Flags	DWORD	0 – normal version; 1 – version was modified in Attachè mode.
	OriginalDate	Binary	Creation of version date and time.
	ModifedDate	Binary	Date and time of version was modified in Attachè mode.
	VersionAuthor	String	Author identifier for version.
	VersionID	DWORD	Version identifier.
	VersionLabel	String	Version name.
	VersionTypist	String	Typist identifier for version.
	ElapsedTime	DWORD	Time, when document is opened (seconds) in offline mode.
	TypingTime	DWORD	Typing time in offline mode.
	KeyBoard	DWORD	Keystrokes entered in offline.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins\ Attache\ Forms			
<<Form's name>>	ColumnsInfo	String	Information about columns. <Column number>-<Width>; …

	Width	DWORD	Form's Width.
	Height	DWORD	Form's Height.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins\ Attache\ Settings\ DetailColumns\ <<Library Name>>			
	<<Object type>>	String	Columns definition for this object type. Includes column order, column width, sort order.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins\ Attache\ Settings\ ViewMode			
<<Library Name>>	<<Object type>>	DWORD	View mode of the object 1 – list; 2 – details; 4 – large icons; 8 – small icons.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins\ Attache\ Settings\ CustomNames			
	Root	String	Customized name for Attachè node.
	Shadow	String	Customized name for Shadow node.

General registry settings

HKEY_LOCAL_MACHINE\ Software\ Hummingbird			
PowerDOCS	CurVer	String	DM Version (“5.0” for example).
PowerDOCS 5.0	Path	String	Full path to DM directory.
HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS			
5.0	Path	String	Full path to DM directory.
HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ Core			
	EnableLogger	DWORD	1 - logger is turned on; 0 – logger is turned off.
	LoggerFile	String	Full path to the logger file.
	LoggerMask	DWORD	Mask for logging: 01 - write into NT log; 10 - write into file, 11 – write into both.

	LoggerLevels	DWORD	Logger level: 0 – minimum; 1 – middle; 2 – maximum (default).
	LogFile	String	Full path to core log file.
	LogMask	DWORD	Core events log mask (0xFFFFFFFF – log all events).
	LogFlush	DWORD	1 – flush log file buffers to disk after each string; 0 – do not flush log file buffers (default).
	LogMinSysRes	DWORD	Minimum system resources to cache log file strings; Windows 95, 98, Me only; by default – 50%.
	UE LogFile	String	Full path to the file where unprocessed exceptions of PowerDOCS.exe will be stored.
Components	DERes	String	Path to the directory where core resource dll is located.
	ATRes	String	Path to the directory where Client Deployment Utility resource dll is located.
	EIRes	String	Path to the directory where E-mail integration resource dll is located.
HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ Forms			
	<<Form name>>	String	Form object Program ID or CLSID. “Use internal form” value is used for internal form.
HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ Forms\ <<Form name>>			
<<Custom Form Identifier		String	Form object Program ID or CLSID
<<Custom Form Identifier	<< Form specific value name>>	String or DWORD	<<Form specific value>>
HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ CacheMgr			
	LogFile	String	Full path to Cache Manager log file.

	LogMask	DWORD	Cache Manager events log mask (0xFFFFFFFF – log all events).
HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ FulView			
FulView	LogFile	String	Full path to Viewer log file.
	LogMask	DWORD	Viewer events log mask (0xFFFFFFFF – log all events).
HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ Core\ ImplementedObjects			
<<Original CLSID for implemented object**>>**	(Default)	String	CLSID for substitution of DM plug-in object.
	Class Name	String	Class name for DM plug-in object.
	Flags	DWORD	0 - Indicates that an object is created by default; 1 - Indicates whether an object supports aggregation; 2 - Indicates whether an object is a singleton.
HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ Core\ Registrars			
<<Custom Registrar ID**>>**	(Default)	String	CLSID or ProgID for custom registrar object.
	Disable	DWORD	1 – registrar is disabled, 0 – registrar is enabled.
HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ Core			
	Root Objects Filter	String	Filtered root objects class name. Multiple objects should be delimiter by ';'.
	Root Objects Filter Type	DWORD	Root objects filter type. 0 (default) - exclude objects specified in "Root Objects Filter". 1 – include only objects specified in "Root Objects Filter"
HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins			
<<Plugin name>>	CLSID	String	Plug-in's module Class ID.
	InsertAt	String	Character. Sets the order of plugins.

	Description	String	Plugin description; used by Administration Tool.
HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ OM\ EventHandlers			
<<Event Handler's name>>	(Default)		EventHandler's CLSID or ProgID.
	InsertAt	String	Character. Sets the order of EventHandlers polling.
	Disable	DWORD	1 – event handler is disabled; 0 – event handler is enabled.
HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ Settings			
	ReplaceInFileName	String	Chars that should be replaced in document filename. By default - " []=|,^<>\t\r\n\\.*?:\"/"
	UseToReplaceInFileName	String	Char that should be used instead of replaced characters. By default – "_"
HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ Settings\ Linking\ AllowLastVer			
<<AppName>>			"Link to Last version" option will be enabled for Application "AppName" if this registry key exist.
HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ Settings\ Linking\ DisableLinking			
	(Default)		"0" or empty – linking will be enabled; "1" – linking will be disabled. "smart" – smart linking will be used. Linking will be enabled when key specified by SmartKeyCode is pressed.

	SmartKeyCode	String	This value defines key for enabling/disabling smart linking. The following codes are supported: 17 – Ctrl, 16 – Shift, 18 – Alt, 144 – Num Lock, 145 – Scroll Lock, 20 – Caps Lock. Multiple keys should be delimiter by comma. Examples: “16, 17” – links will be created only if Ctrl and Shift keys are pressed during Copy operation. “16, 145, 20” – links will be created only when Caps Lock and Scroll Lock is turned on and Shift key is pressed.
<<Application ID>>			“0” or empty – linking will be enabled for the application. “smart” - smart linking will be used for the application “1” – linking will be disabled for application.
	SmartKeyCode	String	Key code for smart linking for the application.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins			
Fusion	CacheLibs	DWORD	1 – cache libraries information in local storage (default); 0 – not cache.
	MaxLibs	DWORD	Maximum number of remote libraries used in DM plug-in.

	NoLoadThread	DWORD	1 - Do not start background thread for reading libraries information. Reads all information in main thread 0 – reads libraries information in a background thread This value is used only in Debug version.
	QuickStart	DWORD	1 - do not load VB forms and COM servers at startup, do not read libraries information.
	CacheDir	String	Path to directory where cache files are located.
	WaitTimeout	DWORD	Wait for the first DM Server call timeout (seconds). By default 7 seconds.
	WTRange	String	Minimum and maximum “WaitTimeout” value (seconds). “Intelligent” timeout feature will adjust timeout value in this range. By default – “2;300” (minimum – 2 sec, maximum – 300 sec)

	WTAttemptsBeforeChange	String	The value specifies how many sequential connects to DM Server should be made before timeout will be changed. The first value specifies how many connects should be performed before timeout will be increased. The second value specifies how many successful connects to the DM Server should be made before timeout will be decreased. By default – “1;10” (Timeout will be increased after the first connect that took more than “WaitTimeout” value. Timeout will be decreased after the 10 sequential connects to the DM Server that took less than “WaitTimeout” value.)
	WTIncreaseStatus	String	Contains information about last connects to the DM Server that took more than “WaitTimeout” time.
	WTDecreaseStatus	String	Contains information about last connects to the DM Server that took less than “WaiTimeout” time.
HKEY_CLASSES_ROOT			
.pdr	(Default)	String	PDRFile (PDR key name in HKCR).
PDRFile	(Default)	String	DM Object Reference.
PDRFile \DefaultIcon	(Default)	String	<DM Path>,-20000.
PDRFile \Shell\open\ command	(Default)	String	<DM Path> %1 (Full path to PDR file).

Custom Forms Registry Settings

HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ Forms

	<<Form name>>	String	Form object Program ID or CLSID. “Use internal form” value is used for internal form.
HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ Forms\ <<Form name>>			
<<Custom Form Identifier		String	Form object Program ID or CLSID
<<Custom Form Identifier	<< Form specific value name>>	String or DWORD	<<Form specific value>>

DM plug-in Settings

- “DM Server General Registry Settings” on page 1373
- “DM Server Login Registry Settings” on page 1385
- “DM Server Forms Registry Settings” on page 1386
- “DM Server Shadow Documents Registry Settings” on page 1389

DM Server General Registry Settings

HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins\ Fusion			
	DisableGPF	DWORD	0 or Not exists - GPF handler is enabled; 1 - GPF handler is disabled.
	EnableLogger	DWORD	1 - logger is turned on for DM plugin, 0 – logger is turned off.
	LogFile	String	Full path to DM plug-in log file
	LogMask	DWORD	DM plugin events log mask 0x0FFFFFFF – logs normal events; 0xFFFFFFFF – logs all events – the extended log.
	APILogFile	String	Full path to document management Plug-in API log file.

	APILogFlags	DWORD	API log file flags: 0x01 – the log file enabled; 0x02 – append log to the existing file.
	AutoReInitTimeout	DWORD	Auto re-initialize timeout for a DM Server call (seconds). By default 4 seconds.
	EnableIntelligentTimeout	DWORD	Enable “intelligent” wait timeout for the first DM Server call: 1 – enable; 0 – disable (default).
	TrackFusionLog	String	Log file name for tracking all DM Server CSI calls.
Components	HelpFile	String	Full path to the DM help file.
	WCFRes	String	Resource file path used by the DM's plug-in.
HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins\ Fusion\ Settings			
	FileName Template	String	Template string for file names. By default – “%LibraryName%-#%DocumentNumber%-v%VersionLabel%-%DocumentName%”
	Reference FileName Template	String	Template string for reference file name (.drf). By default – “%LibraryName%-%DocumentNumber%”.
	ShortFileName Template	String	Template string for short file name. By default – “%DocumentName%”.

	Fusion CSI Threads	DWORD	Maximum number of threads for DM Server CSI calls: 0 - do not create separate threads for DM Server CSI call - making call from the same thread (default); 0xFFFFFFFE - call DM Server CSI from a separate thread for calls from main FsPlugin thread; other value - always call DM Server from a separate thread. Create a new thread if all other running DM Server threads are occupied – up to the maximum number of running threads.
	MonitorFiles	DWORD	Monitors downloaded files to detect possible missing edits issues: 1 – monitor files; 0 – do not monitor files (default).
	Slow Fusion	DWORD	Emulates slow DM Server: 0x01 – when creating new object (profile, search, project); 0x02 – when searching 0x04 – when downloading file; 0x08 – when uploading file.
	Slow Fusion Timeout	DWORD	Slow Fusion timeout - DM will sleep during the specified interval in milliseconds before DM Server call.
DefaultDetailColumns	<<Object Type>>	String	Default columns set definition. The settings allow to specify default column set for the specific object type.
Lookups	Prefill	DWORD	1 – pre-fill lookup with values (default); 0 - not pre-fill lookup.

QuickSearches	FullText Language	DWORD	Fulltext search language. If presented this value will override the current value from the resource module (by default – 0x409 – english).
	FullText Linguistic	String	Fulltext search linguistic. If presented this value will be added to DM Server fulltext searches.
DetailColumnsDefinition	<<Column's SQL info>>	String	DisplayName; Description; Flags: DisplayName – column name in a list, Description - column description in Customize Columns dialog, Flags - flags (1 – do not display this column in a list).
	ExcludeUnknown	DWORD	1 - do not show columns for which SQLInfo is not found (default - 0).
HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins\ Fusion\ Settings\ DetailColumnsDefinition			
<<LibraryName>>	<<Column's SQL info>>	String	Columns definition overwritten for the specific library
	ExcludeUnknown	DWORD	Do not show columns (overwritten for specific library).
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins\ Fusion			
PreCache	Forms	DWORD	Pre-cache forms: 0 – do not pre-cache; 0x01 – pre-cache for primary library; 0x02 – pre-cache for remote libraries included in search; 0x04 – pre-cache for other remote libraries.

	Applications	DWORD	Pre-cache applications: 0 – do not pre-cache; 0x01 – pre-cache for primary library; 0x02 – pre-cache for remote libraries included in search; 0x04 – pre-cache for other remote libraries.
	UserSysID	DWORD	Pre-cache user & primary group SystemID: 0 – do not pre-cache; 0x01 – pre-cache for primary library; 0x02 – pre-cache for remote libraries included in search; 0x04 – pre-cache for other remote libraries.
	FormsDefs	DWORD	Pre-cache forms defaults: 0 – do not pre-cache; 0x01 – pre-cache for primary library; 0x02 – pre-cache for remote libraries included in search; 0x04 – pre-cache for other remote libraries.
	Params	DWORD	Pre-cache Library parameters: 0 – do not pre-cache; 0x01 – pre-cache for primary library; 0x02 – pre-cache for remote libraries included in search; 0x04 – pre-cache for other remote libraries.
	PGParams	DWORD	Pre-cache parameters for user's primary group: 0 – do not pre-cache; 0x01 – pre-cache for primary library; 0x02 – pre-cache for remote libraries included in search; 0x04 – pre-cache for other remote libraries.

	DocTypes	DWORD	Pre-cache document types: 0 – do not pre-cache; 0x01 – pre-cache for primary library; 0x02 – pre-cache for remote libraries included in search; 0x04 – pre-cache for other remote libraries.
	LookupDefs	DWORD	Pre-cache lookups definition: 0 – do not pre-cache; 0x01 – pre-cache for primary library; 0x02 – pre-cache for remote libraries included in search; 0x04 – pre-cache for other remote libraries.
	Lookups	DWORD	Pre-cache lookups: 0 – do not pre-cache; 0x01 – pre-cache for primary library; 0x02 – pre-cache for remote libraries included in search; 0x04 – pre-cache for other remote libraries.
	RED DocNumbers	DWORD	Pre-cache Recently edited documents numbers: 0 – do not pre-cache; 0x01 – pre-cache for primary library; 0x02 – pre-cache for remote libraries included in search; 0x04 – pre-cache for other remote libraries.
	RED Docs	DWORD	Pre-cache Recently edited documents properties: 0 – do not pre-cache; 0x01 – pre-cache for primary library; 0x02 – pre-cache for remote libraries included in search; 0x04 – pre-cache for other remote libraries.

	CleanShadow	DWORD	1 - auto cleanup shadow at startup (if the related value is set in library maintenance utility; 0 – do not cleanup shadow at startup.
	UsersAndGroups	DWORD	Pre-cache users and groups: 0 – do not pre-cache; 0x01 – pre-cache for primary library; 0x02 – pre-cache for remote libraries included in search; 0x04 – pre-cache for other remote libraries.
	LaunchMethods	DWORD	Pre-cache launch methods: 0 – do not pre-cache; 0x01 – pre-cache for primary library; 0x02 – pre-cache for remote libraries included in search; 0x04 – pre-cache for other remote libraries.
	CompatibleApps	DWORD	Pre-cache compatible applications: 0 – do not pre-cache; 0x01 – pre-cache for primary library; 0x02 – pre-cache for remote libraries included in search; 0x04 – pre-cache for other remote libraries.
	Terminology	DWORD	Pre-cache Terminology: 0 – do not pre-cache; 0x01 – pre-cache for primary library; 0x02 – pre-cache for remote libraries included in search; 0x04 – pre-cache for other remote libraries.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins\ Fusion\ Settings			
	LocalPath	String	Local path to downloaded files.

	Hide Transfer Window	DWORD	1 – do not show upload/ download dialog; 0 – show dialog (default).
	ChunkSize	DWORD	Maximum objects to be cached during objects enumeration. By default – 100 objects.
	Export Path	String	Default path for documents export.
	Always Reread Versions	DWORD	1 – refresh versions when it is requested by external module (default); 0 – do not refresh versions.
	SelectLastVersion	DWORD	Auto select last version: 0 – do not auto select last version (default); 1 – auto select last version.
	Create Subversion	DWORD	1 – user can create subversion for any subversion; 0 – user can create subversion only for the last subversion in version (default).
Projects	Column Set	DWORD	0 – a single column set used for all projects (default), 1 – particular column set used for each project.
	Check Children	DWORD	0 - do not detect if a project has subprojects (default); 1 – always detect if a project has subprojects.
QuickSearches	Column Set	DWORD	0 – a single column set used for all QuickSearches; 1 – particular column set used for each QuickSearch (default).
	Max Size	DWORD	Maximum amount of documents shown in QuickSearch (default - 500).

	RED Max Days	DWORD	Maximum amount of days from the current day that defines when a document is included in Recently Edited Documents. By default – 90 days. Enabled when “Use DOCSFusion RED” value is set to 0.
	RED Max Size	DWORD	Maximum amount of documents shown in Recently Edited Documents Search (default - 30).
	Use DOCSFusion RED	DWORD	1 - use Fusion IPCDRecentDoc object (default); 0 - use PCDSearch with “RED Max Days” limitation.
	RED Columns	DWORD	RED Columns behavior: 0 - all columns are available and cached; 1 - only visible columns are cached (default); 2 – a number of available columns is limited to "Available RED Columns" value.
	Available RED Columns	String	List of all available columns for RED search. By default – “$DOCNUMBER; $DOCNAME; $APPLICATION.APPLICATION; DOCSADM.APPS.SYSTEM_ID; $STATUS; $STORAGETYPE”.
	<<Library Name>> Available RED Columns	String	List of all available columns for RED search for the specific library
	AutoInclude In Search	DWORD	1 - automatically allows to add to the current quick search the library that contains this search; 0 – display dialog.
	ChunkFactor	DWORD	Default chunk factor in searches.

Cache	Cache Lookups	DWORD	1 - cache lookups (default); 0 - do not cache lookups (get information from DB at any time).
	Cache Users And Groups	DWORD	1 - cache Users and Groups lists (default); 0 - do not cache (get information from DB any time)
	Pre-cache lookups	DWORD	Which of the lookups need to be pre-cached (read information in background process). The value is provided by the following flags: 0 - do not use pre-cache; 0x01 - pre-cache for DOCUMENTTYPES lookup; 0x02 - pre-cache PEOPLE, 0x04 - pre-cache APPS; 0x08 - pre-cache specific industry lookups (MATTER/ CLIENT/ ORG/ DEPARTMENT), 0x10 - pre-cache %KEYWORDS, 0x1000 - pre-cache lookups for all libraries, otherwise for primary library only); 0x2000 - look through all Profile forms, otherwise through GROUP_DEF form only.
	Max lookup cache	DWORD	Maximum number items in lookup to be cached. By default –200.
	MaxDocTypes	DWORD	Maximum number of documents to be cached. By default –100.
	ChunkFactor	DWORD	Default chunk factor for lookups.
Favorite Folder	<<Library Name>>	String	Projects IDs divided delimiters by comma are added to “Favorites Project".
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins\ Fusion\ Settings\ CustomizedStrings			

<<Library Name>>	PROJ_NAME	String	Customized “Project” name.
	PROJS_NAME	String	Customized “Projects” name.
	PROJSUB_NAME	String	Customized “Sub-project” name.
	PROJMEMB_NAME	String	Customized “Project Membership” name.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins\ Fusion\ Settings\ REDList\ <<UserName>>			
<<Library Name>>	(Default)		Last logged date/time.
	<<Document Number>>	String	Last edit date/time for document.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins\ Fusion\ Settings\ DefaultLaunchMethods			
<<Library Name>>	<<Application ID>>	DWORD	ID of default launch method.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins\ Fusion\ Settings\ ViewMode			
<<Library Name>>	<<Object type>>	DWORD	View mode for the object’s type 1 – list; 2 – details; 4 – large icons; 8 – small icons.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins\ Fusion\ Settings\ DetailColumns\ <<LibraryName>>			
	<<Object type>>	String	Columns definition for the object’s type. Includes column order, column width, sort order.
<<Object type>>	<<Custom properties>>	String, DWORD, BINARY	Properties saved by DM browsers (e.g Outlook Client).
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins\ Fusion\ OM\ CheckedOutDocs\ <<Library Name>>			
<<Document Number>>	<<Version ID>>	String	Full path to document file locked for editing.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins			
Fusion	CacheLibs	DWORD	1 – cache libraries information in local storage (default); 0 – not cache.
	MaxLibs	DWORD	Maximum number of remote libraries used in DM plug-in.

	NoLoadThread	DWORD	1 - Do not start background thread for reading libraries information. Reads all information in main thread 0 – reads libraries information in a background thread This value is used only in Debug version.
	QuickStart	DWORD	1 - do not load VB forms and COM servers at startup, do not read libraries information.
	CacheDir	String	Path to directory where cache files are located.
	WaitTimeout	DWORD	Wait for the first DM Server call timeout (seconds). By default 7 seconds.
	WTRange	String	Minimum and maximum “WaitTimeout” value (seconds). “Intelligent” timeout feature will adjust timeout value in this range. By default – “2;300” (minimum – 2 sec, maximum – 300 sec)

	WTAttemptsBeforeChange	String	The value specifies how many sequential connects to DM Server should be made before timeout will be changed. The first value specifies how many connects should be performed before timeout will be increased. The second value specifies how many successful connects to the DM Server should be made before timeout will be decreased. By default – “1;10” (Timeout will be increased after the first connect that took more than “WaitTimeout” value. Timeout will be decreased after the 10 sequential connects to the DM Server that took less than “WaitTimeout” value.)
	WTIncreaseStatus	String	Contains information about last connects to the DM Server that took more than “WaitTimeout” time.
	WTDecreaseStatus	String	Contains information about last connects to the DM Server that took less than “WaiTimeout” time.

DM Server Login Registry Settings

HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins\ Fusion			
Login	Login Library	String	Library name that will be used for sign-on login. If not specified the DM Server default library will be used.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins\ Fusion			
Login	AutoLogin	DWORD	1 - use auto login; 0 – not use auto login.
	Primary Library	String	Primary library name for auto login.

	Password	String	Password for Auto login (crypted).
	Network Name	String	Network name for auto login.
	LastLoggedLibrary	String	Library name of last login.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins\ Fusion\ Login\ AutoLogin\ <<Library Name>>			
	<<User Name>>	DWORD	1 – user can use auto login; 0 – user cannot use auto-login
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins\ Fusion\ Login			
UserIdent	<<Library Name>>	DWORD	1 - user-supplied identification is allowed in this library; 0 – user supplied identification is not allowed in this library.

DM Server Forms Registry Settings

HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins\ Fusion			
Forms	project_components	String	Form name for reading Project content. The form is located on DM Server. By default – “v_proj_cont”.
	projects	String	Form name for reading Projects. By default – “v_projects”.
	def_profile	String	Form name for reading profiles. By default - “v_defprof”.
	forms	String	Form name for reading library forms. By default – “v_forms”.
	quicksearches	String	Form name for reading quick searches. By default – “v_quicksearch”.
	versions	String	Form name for reading document versions. By default – “v_versions”.
	checkout	String	Form name for reading check-out information. By default – “v_checkout”.

	reqdoc	String	Form name for reading “Restored Documents” quick search content. By default – “v_reqdoc”.
	docsparms	String	Form name for reading library parameters. By default – “v_docsparms”.
	groups	String	Form name for reading library groups. By default – “v_groups”.
	peoplegroups	String	Form name for reading library groups content (people included in a group). By default – “v_peoplegroups”.
	peoples	String	Form name for reading users information. By default – “v_peoples”.
	appinfo	String	Form name for reading applications. By default – “v_appinfo”.
	launchmethod	String	Form name for reading launch methods. By default – “v_launchmethod”.
	compapps	String	Form name for reading compatible applications. By default – “v_compapps”.
	remlibs	String	Form name for reading remote libraries. By default – “v_remlibs”.
	activitylog	String	Form name for reading a document’s history. By default – “v_activitylog”.
	FormUser	String	Form name for reading default forms settings. By default – “v_FormUser”.
	typesdoc	String	Form name for reading document types. By default – “v_typesdoc”.
	Lookups	String	Form name for reading lookups information. By default – “v_lookups”.
	Keywordix	String	Form name for reading document keywords.

	OrgHierarchy	String	Form name for reading NDS organizations. By default – “v_orghierarchy”.
	NetworkAliases	String	Form name for reading network aliases. By default – “v_naliases”.
	profile_defaults	String	Form name for profile defaults. By default – “v_profile_defaults”.
	Terminology	String	Form name for Terminology. By default – “v_Terminology”.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins\ Fusion\ Settings\ DefaultForms			
<<Library name>>	<<Application ID>>	String	Default form for this application.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins\ Fusion\ Forms			
<<Form name>>	Width	DWORD	Form's width.
	Height	DWORD	Form's height.
	ColumnsInfo	String	Information about columns order and width in list view.
	ColumnsInfoEx	String	Alternative form columns info (order and column width) -"Col1Order-Col1Width;Col2Order-Col2Width;..."
Access	LastView	DWORD	Last view mode in Access form 1 – view NDS hierarchy (default); 0 – view normal hierarchy.
Export	LastAction	DWORD	Last user’s operation 0 – Check-out (copy); 1 – Check-out and Lock; 2 – Create Reference.
Mail	LastAction	DWORD	Last user's operation 0 – send copy; 1 – send copy and checkout; 2 – send reference.
SelectVersions	ProfileColumnsInfo	String	Information about Profile's columns

	VersionColumnsInfo	String	Information about Version's columns.
Lookup	MaxRecentItems	DWORD	Maximum number of items in “Recently used items” tab. By default – 10.
	StartInRecent	DWORD	1 - start lookup form on “recently used items” tab; 0 – start lookup form on “Full list” tab.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins\ Fusion\ Forms\ Lookup\ <<Library Name>>\ <<Lookup ID>>			
	MaxRecentItems	DWORD	Allows to redefine value for the given lookup.
	StartInRecent	DWORD	1 - start lookup form on “recently used items” tab for this lookup; 0 – start lookup form on “Full list” tab for this lookup.
Recent Items	<<Number>>	String	Recently used item (properties of the item are divided by semicolon).
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins\ Fusion\ Forms\ QR			
<<Library Name>>			QuickRetrieve form properties for the specific library

DM Server Shadow Documents Registry Settings

HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins\ Fusion			
DOCSShadowPath	Drive	String	Full path where shadow documents are stored.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins\ Fusion\ Shadow			
	Delete Days	DWORD	Number of the days defined by user for which shadow documents are to be deleted.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Core\ Plugins\ Fusion\ Shadow\ <<Library Name>>\ <<Document identifier>>			
	AppID	String	Application identifier.
	AuthorID	String	Author identifier.
	DocNumber	DWORD	Document number.

	LibraryName	String	The source library name, from which the document was copied.
	Storage	String	Storage type of document.
<<Version identifier>>	(Default)	String	Version label.
	FilePath	String	Full path to shadow file.
	VersionAuthor	String	Author identifier of version.
	VersionID	DWORD	Version number.
	VersionTypist	String	Typist identifier of version.
	UTCOffset	DWORD	Offset from UTC when file was last accessed (minutes)
	Edited	BINARY	Last edited date/time

E-Mail Integration related registry settings

HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ EMail Integration\ 1.5			
DMS	Current:CLSID	String	Current DMS module Class ID.
	Current:Description	String	Current DMS module description.
	List of CLSIDs	String	List of Class IDs for all DMS modules.
	List of Descriptions	String	Descriptions of all DMS modules.
Help	HelpPath	String	This entry names full path to DM Extensions E-mail Integration help files.
HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ EMail Integration\ 1.5 \ Settings			
	Format	String	0 – Allows to save messages in RTF format; 1 – Allows to save messages in MSG format..
	Type	String	"registry" - obtains the current settings from registry.

	DefaultInsertType	String	Default insert type Possible values: (Empty), Copy, CopyAndCheckout, Reference. The entry defines a default value for Insert From DM documents to the messages
	IntegrationType	String	None – disables Integration; Weak – this integration type does not store message parameters. The messages and attachments are stored in DM as simple separate documents; Strong – this integration supports storing the message parameters. (for example: MAIL_ID, ATTACH_NUM, MSG_ITEM, DELIVER_REC, PARENTMAIL_ID and THREAD_NUM).
	Unload	DWORD	1 – unloads DM Extensions after each call; 0 – keeps DM Extensions in memory until e-mail application is closed (default).
HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ EMail Integration\ 1.5\ Settings\ <<LibraryName>>			
	EmailGroupName	String	Contains a DM Extensions e-mail user group name to be used for security.
	EmailMessageAppID	String	This entry is used for messages that are to be saved with this Application ID value into DM Extensions.
	EmailMessageDocType	String	This entry is used for messages that are to be saved with this Document Type value in DM Extensions.

	EmailMessageMSGAppID	String	This entry is used for messages that are to be saved in MSG-format. Message will be saved with this Application ID value in DM Extensions.
	EmailMessageMSGDocType	String	This entry is used for messages that are to be saved in MSG-format. Message will be saved with this Document Type value in DM Extensions.
	UnknownAppID	String	This entry is used for attachments when E-mail integration cannot find primary or any compatible application of the extension of attachment.
	UnknownDocType	String	This entry is used for attachments when Email Integration cannot find primary or any compatible Document Type of the extension of attachment.
	ProfileForm	String	The name of the default profile form used in document saving for the enhanced integration.
	AttachmentDocName	String	FileName – use filename for document name of attachments (default); EMailSubject – use message subject for document name of attachments.
	Security	String	None – none security (default); CurrentUser – grants access only to the current user; AllRecipients – grants access to all message recipients; EmailGroup – grants access to e-mail group

	VersionCommentsMessage	String	EMailSubject – use message subject for version comments (default); Custom – use custom comments.
	VersionCommentsAttachment	String	FileName – use attachment file name for version comments (default); EMailSubject – use message subject for version comments; Custom – use custom comments.
	SaveEmailMessage	String	Initial state of the “Save message(s) body” checkbox: Yes (default); No.
	SaveAttachments	String	Initial state of the “Delete original message(s) after save” checkbox: Yes; No (default).
	DeleteEmailAfterSave	String	Initial state of the “Save attachments” checkbox: Yes (default); No.
FieldMappings	BCC	String	DM column name containing BCC value.
	CC	String	DM column name containing CC value.
	Date	String	DM column name containing DATE value.
	From	String	DM column name containing FROM value.
	Subject	String	DM column name containing SUBJECT value.
	TO	String	DM column name containing TO value.
MapEmailNameToPCDOCS	<<Email User Name>>	String	DM user name. Sets one-to-one relation between e-mail user name and DM user name.
HKEY_CURRENT_USER\ SOFTWARE\ Hummingbird\ Email Integration\ 1.5			

SaveForm	Height	DWORD	Save form height.
	Width	DWORD	Save form width.
	ColumnsInfo	String	Save form columns information.
	DefaultFormat	DWORD	1 – save in MSG format (default); 0 – save in RTF format.
HKEY_LOCAL_MACHINE\ SOFTWARE\ Microsoft\ Exchange\ Client\ Extensions			
	PowerDOCS E-mail integration	String	4.0;<DM Extensions Install Folder>\ EmailIntegration\ EMIExch.dll;1

MS Outlook DM Extension Registry Settings

HKEY_LOCAL_MACHINE\ SOFTWARE\ Hummingbird\ PowerDOCS\ Outlook Client			
	InstallPath	String	Install Path.
Components	ORes	String	Full path to Outlook Client resource module.
HKEY_LOCAL_MACHINE\ SOFTWARE\ Hummingbird\ PowerDOCS\ Outlook Client\ GoodApps			
	App1	String	RichEdit20A.
	Comments	String	
	DefScenario	Number	0
HKEY_CURRENT_USER\ SOFTWARE\ Hummingbird\ PowerDOCS\ Outlook Client			
	Disable5414Workaround	String	Outlook 2000 SR-1 E-mail Security Update (this update changes Outlook version to 9.00.5414) for MS Outlook causes crash Outlook when Outlook Client for DM Extensions is installed. DM Extensions implements workaround to bypass this issue. If any not empty string (except for “0”) specified in this registry value workaround will be disabled.

Installation related registry settings

HKEY_LOCAL_MACHINE\ Software\ Hummingbird \ PowerDOCS \ Installation			
	Path	String	Full path to DM directory.
	Main	DWORD	Bitwise number that specifies the following installed components: 0x01 – Desktop; 0x02 – Client Deployment Utility; 0x04 – Word AutoExec macro; 0x08 – Core components; 0x10 – Windows Explorer DM Extension; 0x100 – Microsoft Outlook DM Extension.
	EMail	DWORD	Bitwise number that specifies installed e-mail integration components: 0x01 – Outlook; 0x10 – Lotus Notes; 0x100 – Groupwise.
	Passive	DWORD	Bitwise number that specifies installed passive integration components: 0x01 – MS Word; 0x010 – MS Excel; 0x100 – MS Project; 0x02 – Acrobat Exchange.
InstalledComponents	WORD	String	“1” if MS Word passive integration is installed.
	EXCEL	String	“1” if MS Excel passive integration is installed.
	PROJECT	String	“1” if MS Project passive integration is installed.
	ACROBAT	String	“1” if Acrobat Exchange passive integration is installed.
	OUTLOOK	String	“1” if MS Outlook e-mail integration is installed.

	LOTUS	String	“1” if Lotus Notes e-mail integration is installed.
	GROUPWISE	String	“1” if GroupWise e-mail integration is installed.
	EXPLORER	String	“1” if Windows Explorer DM Extension is installed.
	OUTLOOKCLIENT	String	“1” if Microsoft Outlook DM Extension is installed.
	ADMINTOOL	String	“1” if Client Deployment Utility is installed.

User Interface related registry settings

HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Viewer			
	Toolbar	DWORD	Specifies whether or not the viewer’s toolbar is visible. 0 - toolbar is hidden; 1 - toolbar is visible (default).
	StatusBar	DWORD	Specifies whether or not the viewer’s status bar is visible. 0 - status bar is hidden; 1 - status bar is visible (default).
	WindowPlacement	Binary	Specifies the viewer’s WINDOWPLACEMENT structure as returned by the GetWindowPlacement function. If the entry does not exist, the default window position is used.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Settings			
	AutoSelect	DWORD	1 – auto select first item in list (default); 0 – do not select first item automatically.
	WindowPlacement	BINARY	Specifies the viewer’s WINDOWPLACEMENT structure as returned by the GetWindowPlacement function for DM Extensions.

	Open in own window	DWORD	1 – open each folder in its own window (default); 0 – open each folder in the same window.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Settings\ Linking			
	LinkToVersion	DWORD	When a link to Hummingbrid DM Excel (or optionally Word) document is created, user can define which version of the document the link is pointed to: -1 – not specified, dialog should be displayed (default); 0 – use current version; 1 – use last version.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Settings\ Admin			
Defaults	Favorites Bar	DWORD	1 – show Favorites bar in DM Extensions (default); 0 – do not show Favorites bar.
	Status Bar	DWORD	1 – show Status bar in DM Extensions (default); 0 – do not show Status bar.
	Toolbar	DWORD	1 – show Toolbar in DM Extensions (default); 0 – do not show .
	Templates	DWORD	1 – show Templates in DM Extensions (default); 0 – do not show Templates.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Settings\ FavoritBar			
	Width	DWORD	Favorites bar width in pixels. By default – 100.
<<Library Name>>	ActiveFolder	DWORD	Active group number.
	FolderCount	DWORD	Number of groups.
	View	DWORD	View mode: 0 – large icons (default); 1 – small icons.
	Folder#<<Folder Number>>Caption	String	Group description.
	Folder#<<Folder Number>>ItemCount	DWORD	Number of items in group.

	Folder#<<Item Number>>Item#<<Folder Number>>	String	Object ID for item in group.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Settings\ Toolbar			
	Inscription	DWORD	0 – view toolbar buttons with text labels (default); 2 – view toolbar buttons without text labels.
<<Loaded plug-ins list>>	Settings	String	Toolbar buttons definition;
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Settings\ HierarchyList			
Printing	Font	BINARY	Fonts for printing list content.
	Header	String	Header for printing list content.
	Footer	String	Footer for printing list content.
	Margins	BINARY	Printing margins.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Settings\ PD_Pages			
	ExplorerTabName	String	Name of Explorer tab. By default – “Explorer”.
	NewTabName	String	Name of New tab. By default – “New”.
	NewSplitPercent	DWORD	Position of splitter for New tab in percentage. By default – 50%.
	TreeSplitPercent	DWORD	Explorer tab vertical splitter position in percentage. By default – 30%.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Settings\ PD_Pages\ <<Library Name>>			
	StartPage	String	Initial page name when DM Extensions is starting.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Settings\ PD_Pages\ <<Library Name>>\ <<Page Name>>			
	HorPercent	DWORD	Horizontal splitter position in percentage. By default – 30%.
	ListAppearance	DWORD	View mode for list 1 – list; 2 – details (default); 4 – large icons; 8 – small icons.

	ShowAddons	DWORD	1 – add-ons pane is visible (default); 0 – add-ons pane hidden.
	ShowTree	DWORD	1 – tree pane is visible (default); 0 – tree pane is hidden.
	TreePath	String	Object IDs for selection in Tree pane.
	TreeRoot	String	Root object for Tree pane hierarchy.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Settings\ PD_Pages\ <<Library Name>>\ <<Page Name>>\Addons			
	Current	String	Active add-on name.
<<Add-on name>>	Hide	DWORD	0 – add-on is visible 1 – add-in is hidden
HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ Explorer			
	Use Cache List	DWORD	1 - Cache data in list (default) 0 – do not cache data in list
	LogFile	String	Full path to DM Extensions log file.
	LogMask		DM Extensions events log mask (0xFFFFFFFF – log all events).
	Edit Labels	DWORD	1 – allow to edit labels in Desktop tree pane; 0 – disable to edit labels (default).
Shell	LogFile	String	Full path to the Windows Explorer DM Extension log file.
	LogMask	DWORD	Windows Explorer DM Extension events log mask (0xFFFFFFFF – log all events).
VisualStyles	<<Object Visual Style Name>>	String	Font and colors used to display object of this type.
HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ Explorer\ AddOns			
<<AddOn name>>	(Default)		Add-on CLSID or ProgID.

	Description	String	Add-on title shown in tab. If absent, add-on name is shown; also used in Administration Tool.
	Disable	DWORD	1 – add-on is disabled; 0 – add-on is enabled.
	InsertAt	String	Character. Sets the order of add-on polling.
	WorksWith	String	Plug-in list add-on can work with.
	ObjectsSupported	String	Objects class name supported by this add-on.
HKEY_LOCAL_MACHINE\ Software\ Hummingbird\ PowerDOCS\ Explorer\ Tabs			
<<Custom Tab Name>>	(Default)		Tab CLSID or ProgID.
	Disable	DWORD	1 – tab is disabled; 0 – tab is enabled.
	WorksWith	String	Plug-in list tab can work with.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Explorer			
	Tree Refresh Time	DWORD	
	Hide Splash Screen	DWORD	1 – hide splash screen for DM Extensions; 0 – do not hide splash screen (default).
	Shell Addon View	DWORD	1 – view add-ons pane in Windows Explorer DM Extension (default); 0 - hide add-ons pane in Windows Explorer DM Extension.
HKEY_CURRENT_USER\ Software\ Hummingbird\ PowerDOCS\ Explorer\ Addons			
	Current	String	Active add-on name in Windows Explorer DM Extension and Outlook Client.
<<Add-on name>>	Hide	DWORD	1 – add-on is hidden in Windows Explorer DM Extension and Outlook Client; 0 – add-on is visible.
	Description	String	Customized add-on description.

VisualStyles	Plain/_Plain	String	Used to define the default style for an object if the object doesn't support any visual style.

Index

Symbols

_Item **819**
_NewEnum **291**

A

Abort **1299**
AccelKey **162**
AccessControlAccess **487**
AccessCopy **489**
AccessDelete **490**
AccessDeleteSearch **601**
AccessEditDocument **491**
AccessEditProfile **492**
AccessEditSearch **602**
AccessMode **679**
AccessRetrieve **493**
AccessType **680**
AccessViewDocument **494**
AccessViewProfile **495**
AccessViewPublishedOnly **496**
AccessViewSearch **603**
Activate **1003**
Active **799**
ActiveAddonChanged **1209**
ActiveFolder **775**
ActivityInfo **193**
Add **674**
AddDocument **499**
AddDocumentActivity **422**
AddDocuments **1317**
AddGroup **610**
AddItem **1072**
AddMenuItem **30**
AddObject **817**
AddOnCnt **1207**
AddOnCount **1209**
AddonDescription **1210**
AddonEnabled **1210**
AddonName **1211**
AddonVisible **1212**
AddPerson **612**
AddSink **1301**
AddSubProject **500**
AddToRelated **303**
AddVerb **830**
AddVersions **1318**
AfterVerb **739**
Alignment **1186**
AllowHideCommand **776**
AnimationSpeed **776**
App **203**
Appearance **739**
AppForms **222**
AppID **1324**
AppInformation **363**
Application **407**
AttacheDocuments **1319**
Attachments **1267**
AttachmentType **1290**
Author **1339**

B

BackColor **740**
BCC **1264**
BCObject **806**
BeforeVerb **740**
BeginEdit **614**
Billable **205**
BitmapAt **1155**
Border **1037**
BorderStyle **716**
BrowseObject **717**
BrowseObjectFlags **174**
BrowserInfoFlags **174**
BuildCmdLineString **961**
ButtonPressed **1119**

C

CanAdd **897**
Cancel **58**
CancelMonitor **963**
CancelTransfer **628**
CanChangeStatus **670**
CanChangeVersionStatus **680**
CanControlAccess **475**
CanCopy **477**
CanDelete **478**
CanDeleteProject **541**
CanDeleteSearch **595**
CanDrag **53**
CanDropBetweenObjects **54**
CanEditDocument **479**
CanEditProfile **480**
CanEditProject **542**
CanEditSearch **597**
Canned **571**
CanRemove **936**
CanRename **800**
CanRestoreColumnDefaults **742**
CanRetrieve **481**
CanSaveAsNewSubversion **629**
CanSaveAsNewVersion **630**
CanSaveColumnDefaults **743**
CanViewDocument **483**
CanViewProfile **484**
CanViewProject **543**
CanViewSearch **598**
Caption **558**
CC **1264**
ChangeDisplayNameForm **527**
ChangeMenuItem **26**
ChangeState **831**
ChangeStatus **1244**
ChangeVersionTypeForm **528**
CheckedTrigger **1054**
CheckInActions **681**
CheckOut **631**
CheckoutPath **423**
ChildControl **802**
Clear **295**
ClearValue **270**
ClientHeight **1120**
ClientWidth **1120**
Clone **59**
CloseDoc **1004**
CloseDocEx **1015**
CloseOpenDocuments **964**
CMSBox **1034**
CMSCheckBox **1049**

CMSEdit **1068**
CMSForm **1116**
CMSGotFocus **1121**
CMSList **1136**
CMSLostFocus **1122**
CMSPush **1153**
CMSRadioGroup **1169**
CMSText **1184**
ColumnFlags **682**
Columns **572**
ColumnTypes **682**
ComboBox **1073**
CommandLine **334**
Comment **633**
CompatibleWith **223**
Completed **1225**
ComponentDisplayName **529**
Contents **471**
ContextMenuEnabled **719**
ContextMenuMode **779**
ContextParamSupported **102**
Controls **1123**
ControlType **1189**
Copy **426**
CopyAll **1226**
CopySelection **1226**
CoreEvents **175**
CoreFlags **177**
Count **358**
CreateBrowser **37**
CreateDocFromTemplate **364**
CreateProfile **366**
CreateProject **367**
CreateSearch **368**
CreateVersionFromFile **426**
CreateVersionFromFileFlags **683**
CurrentLibrary **242**
CurrentVersion **429**
CustomizeStructuredView **744**

D

DataChanged **1075**
DataTypes **683**
DblClick **1140**
DDEOpenAppName **335**
DDEOpenCmd **336**
DDEOpenCmd2 **337**
DDEOpenTopic **338**
DDEPrintAppName **339**
DDEPrintCmd **340**
DDEPrintCmd2 **341**
DDEPrintTopic **342**
Default **343**
DefaultDirectory **344**
DefaultExtension **224**
DefaultForm **400**
DefParentWindow **243**
Delete **503**
DeleteAllVersions **433**
DeleteVerb **832**
Description **120**
DEvents **862**
DEventTypes **865**
Disabled **226**
DisplayMessage **263**
DisplayMessageTypes **684**
DisplayModal **559**
DisplayName **164**

DocNumber **1326**
DocProfiles **195**
Document Attributes **1025**
Document Management Objects Structure (Application tree)
ree) **187**
DocumentClosed **976**
DocumentCount **561**
DocumentOrder **522**
Documents **575**
DocumentsCount **576**
DocumentVersionType **530**
DoLookup **1075**
DoubleClick **1125**
DoVerb **93**
DownBitmap **1158**
Download **663**
DownloadFile **635**
DownloadToFile **1309**
Drop **55**
DropEffect **55**
DST **244**

E

EditAlignment **1076**
EditBackColor **1077**
EditBorder **1078**
EditColumns **745**
EditFont **1079**
EditForeColor **1080**
EditStyles **1081**
EISaveFileFormat **1289**
EISaveFlags **1288**
ElapsedTime **206**
Enabled **1213**
EnabledRights **937**
EnumAllVerbs **41**
EnumChildObjects **155**
EnumContentProperties **85**
EnumVerbs **42**
ErrCode **412**
ErrorText **1083**
ErrText **413**
EventFired **1245**
ExecuteMenuItem **857**
ExecuteMenuItems **296**
Explorer Core Object Model **11**
Export **636**
ExportInFlags **954**
ExportOutFlags **955**
Extension **1273**

F

Favorite **531**
FavoritesBar **773**
FavoritesBarContextMenu **781**
FavoritesFolders **404**
Fetch **435**
FileNameMask **408**
FilePath **637**
FileTypes **685**
FilingScheme **227**
Filter **720**
FilterMode **746**
Find **60**
FinishEdit **616**
FireLastRowRetrieved **1142**
FireRowRetrieved **1143**

FirstHit **1227**
Flags **122**
Folder **797**
FolderBackColor **782**
FolderChange **783**
FolderContextMenu **783**
FolderFont **784**
FolderForeColor **785**
FolderItem **805**
FolderItems **816**
Folders **811**
Font **786**
ForeColor **1228**
Format **1084**
FormatAllWindowControls **605**
FormatMenu **606**
FormatTerms **607**
FormatWindowText **608**
FormDefinition **307**
FormName **436**
Forms **371**
FormsResults **686**
FormStyles **1126**
From **1262**
FTypes **686**
FullPath **723**
FulViewOCX **1221**

G

get_Size **61**
GetAlternateContent **1016**
GetAt **131**
GetBrowser **35**
GetBrowserFlags **177**
GetBrowserInfo **27**
GetBrowserInfoEx **34**
GetColumnProperty **132**
GetControl **1128**
GetDefault **123**
GetDetailColumns **103**
GetDMSInfo **1005**
GetDocInfo **1005**
GetDocRelation **1016**
GetEffectiveRights **618**
GetLastChangeDate **1295**
GetLastError **1298**
GetLeadMoniker **1007**
GetLocalFile **1308**
GetLocalFileName **1294**
GetMenu **833**
GetMenuItems **859**
GetNameFromSQL **308**
GetObjectParam **46**
GetParentWindow **28**
GetPluginClass **112**
GetPluginInfo **117**
GetProfile **372**
GetProject **373**
GetProperty **86**
GetSession **36**
GetSessionInfo **156**
GetSize **1296**
GetStructuredViewDefinitions **104**
GetTerminology **265**
GetValue **1129**
GetVariantParam **47**
GetViewMode **108**

H

HasLookup **273**
HasOpenDocuments **979**
HasValidValue **1086**
hasValue **1058**
HeaderMenuEnabled **750**
HelpID **1160**
HiddenTrustees **913**
Hide **787**
HideColumnHearders **750**
HideSelection **751**
Hierarchy **713**
HierList **735**
HitsEnabled **1229**
HitTest **788**
HResultCodes **865**
HResultCodesEx **866**
Hscroll **1087**
hWnd **789**

I

IActivityInfo **202**
IAppForm **213**
IAppForms **217**
IAppInfo **220**
IAppInfo2 **240**
IAppIntegration **960**
IApplication **241**
IApplication2 **256**
IApplication3 **261**
IApplication4 **262**
IApplication5 **264**
IApplications **266**
IAttacheRoot **1317**
IBCBrowser **25**
IBCBrowser2 **30**
IBCBrowser3 **33**
IBCBrowserCore **34**
IBCBrowserFactory **37**
IBCContextMenuSupport **38**
IBCCoreCtx **46**
IBCDragDropSupport **52**
IBCEnumObjects **57**
IBCEnumProperties **68**
IBCEnumVerbs **74**
IBCObject **80**
IBCObject2 **90**
IBCObjectArray **91**
IBCObjectCaps **102**
IBCObjectCaps2 **107**
IBCObjectChanged **110**
IBCPlugin **111**
IBCProperty **119**
IBCPropertyArray **127**
IBCRegistar **141**
IBCRegistrarExt **145**
IBCRetrievalSink **151**
IBCSession **153**
IBCVerb **160**
IChangeDisplayNameForm **874**
ICloseNotify **976**
IColumn **269**
IColumn2 **281**
IColumns **285**
ICompatibleAppsForm **875**
Icon **165**
IconActive **166**
IconDown **167**

Iconized **578**
IconSizeBitmap **1161**
IControlMenu **829**
ID **346**
IDOComponent **315**
IDOComponents **321**
IDocProfiles **292**
IDocProfiles2 **302**
IDOCSForm **306**
IDOCSForms **312**
IDocTransferNotify **327**
IDownloadDataSink **1292**
IDownloadResourceProvider **1293**
IDownloadSession **1299**
IDownloadSink **1302**
IEIAttachment **1272**
IEIAttachments **1282**
IEICore **1258**
IEIMessage **1261**
IEIMessages **1277**
IExportForm **877**
IGroupCheckinForm **880**
IGroupCheckoutForm **881**
IHistoryForm **883**
IIntegrationScheme **978**
ILaunchMethod **331**
ILaunchMethodChooserForm **885**
ILaunchMethods **354**
ILibraries **357**
ILibrariesForm **886**
ILibrary **360**
ILibrary2 **400**
ILibrary3 **403**
ILibraryChooserForm **887**
ILocalFileCacheManager **1306**
ILocationForm **889**
ILookupForm **890**
Image **1041**
ImageType **1042**
IMailForm **893**
IMonitorFile **406**
IMonitorFiles **409**
Implementing Custom Form **871**
Indentation **725**
Init **1060**
InitTransfer **328**
InsertAt **97**
InsertBefore **169**
InsertComponent **532**
InsertDocument **523**
InsertDocuments **1260**
InsertSubProject **524**
IntegrationScheme **347**
InvalidInput **1090**
IODMDocMan **1001**
IODMDocMan2 **1013**
IODMQuery **1023**
IOMDocumentEventSink **838**
IOMDocumentMenuSink **848**
IOMEnumSink **854**
IOMErrInfo **412**
IOMEventHandler **852**
IOMMenuSink **856**
IOMObjects **414**
IProfile2 **470**
IProfileAccessForm **894**
IProfileAccessForm2 **896**
IProfileForm **904**

IProfileFormChooser **906**
IProfileTrustee **474**
IProfileTrustee2 **486**
IProject **497**
IProject2 **521**
IProject3 **526**
IProjectAccessForm **908**
IProjectAccessForm2 **909**
IProjectChooserForm **917**
IProjectChooserForm2 **919**
IProjectForm **921**
IProjectMembershipForm **922**
IProjectMembershipForm2 **924**
IProjects **534**
IProjectTrustee **540**
IProperties **546**
IPropertiesForm **926**
IProperty **551**
IQuickRetrieve **556**
IQuickRetrieveCallback **566**
IQuickRetrieveForm **928**
IRelatedItemsForm **929**
IsAttacheDocument **1328**
ISaveAsOptionsForm **931**
IsBounded **1329**
IsCurrent **374**
ISearch **567**
ISearchAccessForm **933**
ISearchAccessForm2 **934**
ISearchCriteriaForm **942**
ISearches **591**
ISearchFormChooser **944**
ISearchOptionsForm **945**
ISearchTrustee **595**
ISearchTrustee2 **601**
ISelectVersionsForm **947**
ISelectVersionTypeForm **949**
IsEqual **135**
IsGroup **545**
IsLocalFile **1307**
IsLocked **1330**
IsModified **1331**
IsPrimary **376**
Item **1277**
ItemClick **790**
ItemContextMenu **791**
ITemplateChooserForm **951**
Items **1179**
ITerminology **604**
ITrusteeCollection **609**
IUndockedProfile **1323**
IUndockedProfiles **1335**
IUndockedVersion **1339**
IUndockedVersions **1344**
IUploadResourceManager **1313**
IVersion **625**
IVersion2 **663**
IVersion3 **669**
IVersionChooserForm **952**
IVersions **673**

K

KeyFlags **177**
Keystrokes **207**

L

LastEditDate **638**
LastHit **1230**

LastObjectRetrieved **752**
LastRowReason **178**
LastRowRetrieved **152**
LastVersion **437**
LaunchApplication **966**
LaunchMethodChooserFlags **687**
LaunchMethods **230**
LeftText **1061**
Libraries **246**
Library **438**
LibraryChooserForm **888**
LibraryGroup **377**
LibraryName **1332**
LineStyle **725**
List **1090**
ListIndex **1091**
LM_INTEGRATIONSCHEME **688**
LocalFile **667**
Location **348**
LockProfile **439**
LockProfileFlags **688**
LockVersionForEdit **639**
LoggerOCX **1243**
LoggerOCXNotificationSink **1244**
Login **257**
LoginCAP **263**
Logoff **113**
Logon **114**
LookupName **1092**
LowerCase **1093**

M

Mail **441**
MakeCurrent **378**
MaxChars **1093**
MaxFolderNumber **792**
MaxSelectionSize **753**
MethodsFlags **689**
Mode **261**
Modes **689**
MonitorClose **967**
MonitorDir **349**
MonitorFiles **350**
MonitorSubdirs **351**
MouseMove **792**
MultiLine **1094**
MVFValue **275**

N

Name **1333**
NewDoc **1008**
Next **62**
NextHit **1230**
NoChildrenText **754**
Number **1275**

O

ObjectByID **118**
ObjectCount **754**
ObjectDblClicked **726**
ObjectRetrieved **755**
ODMA Document ID **1000**
ODMAIntegration **959**
ODMGetODMInterface **1001**
OLEDragMode **727**
OLEDropMode **793**
OMActivityConsts **690**
OMErrorCodes **692**

OMFlagsLookUp **693**
OMFlagsQR **694**
OnAbort **1305**
OnBegin **1302**
OnComplete **1304**
OnConnect **1303**
OnData **1293**
OnDataChange **110**
OnEnumChildObjects **855**
OnError **329**
OnOk **566**
OnProgress **1306**
OnProgressNotify **330**
Opaque **1044**
Open **443**
OpenDoc **1008**
OpenDocument **971**
OpenForRead **1297**
OpenForWrite **1314**
OpenInCompatibleApp **445**
Order **317**
OrderBy **63**

P

Pages **208**
PageSetup **756**
Parent **170**
ParentID **1268**
ParentProjects **447**
ParentVersion **318**
Password **1096**
PopupMenuEnabled **757**
Position **804**
Prev **64**
Preview **645**
PreviewCommand **1146**
PreviewCommands **758**
PreviewDoubleClick **759**
PreviousHit **1231**
PrimaryLibrary **248**
Print **986**
PrintApp **231**
PrintCurrentDoc **1232**
PrintDoc **450**
PrintOut **760**
PrintSetup **1233**
PrintVersion **647**
Profile **319**
ProfiledDocument **533**
ProfileDeleteFlags **694**
ProfileDirectory **404**
ProfileEvent **838**
ProfileFile **379**
ProfileRights **695**
ProfileStatus **696**
ProfileUserDefaults **401**
ProfView **1218**
Project **197**
ProjectChooserFlags **697**
ProjectID **510**
ProjectRights **697**
Projects **380**
Prompt **1097**
PromptAlignment **1098**
PromptBackColor **1099**
PromptBorder **1100**
PromptFont **1101**
PromptForeColor **1102**

PromptLocation **1103**
PromptWidth **1104**
Properties **381**
PropertyArrayFlags **179**
PropertyArraySaveFlags **179**
PropertyAttributes **180**
PublicFolders **405**
Publish **472**
Purge **1312**

Q

QRCReturnFlags **698**
QueryCapability **1017**
QueryClose **1023**
QueryExecute **1024**
QueryGetResults **1025**
QuickRetrieve **199**
QuickSearchesID **698**
QuickSearchRights **699**

R

ReadOnly **1105**
RefDoc **209**
Refresh **31**
RegisterClass.htm **142**
RegisterExtObject **146**
RegisterFlags **181**
Registering Custom Form **872**
RegisterObjectParam **48**
RegisterVariantParam **49**
RegistrarCommand **181**
registry settings **1389**
RelatedItems **472**
ReleaseDoc **450**
Rem ove **538**
Remove **300**
RemoveAll **98**
RemoveAt **137**
RemoveDocument **511**
RemoveDocuments **1320**
RemoveFromRelated **304**
RemoveGroup **621**
RemoveItem **1106**
RemoveMenuItem **32**
RemovePerson **622**
RemoveSubProject **512**
RemoveTempFiles **975**
Rename **804**
ReorderEnabled **761**
Required **1107**
Reset **65**
RestoreColumnDefaults **761**
RestoreDefaults **137**
RestrictedTrustees **901**
Resume **1301**
Retrieve **66**
RetrieveColumns **762**
RetrievingText **763**
RevokeObjectParam **50**
RevokeVariantParam **51**
RightsTemplates **940**
Root **728**
RowRetrieved **1133**
RowsRetrieved **152**

S

SAOFFlags **700**
Save **514**

SaveAs **1010**
SaveAsEx **1018**
SaveAsNewSubversion **648**
SaveAsNewVersion **650**
SaveColumnDefaults **764**
SaveDoc **1011**
SaveDocEx **1019**
SaveMessages **1258**
SaveSettings **794**
SaveToFile **1276**
ScrollToEnd **67**
Search **563**
SearchByDefault **389**
SearchDialog **1234**
Searches **391**
SearchID **583**
SearchNext **1235**
SelChanged **1236**
SelectAll **765**
SelectDoc **1012**
SelectDocEx **1020**
SelectedAddon **1216**
SelectedDocuments **564**
SelectedSubProjects **516**
SelectForEdit **453**
SelectForEditFlags **700**
SelectForEditReturnFlags **701**
Selection **729**
SelectionChanged **730**
SelectProfileForm **392**
SelectProjectForCriteria **919**
SelectSearchForm **394**
SelectTemplate **951**
SelectVersionsFormFlags **701**
SelSelection **1242**
SendDate **1266**
Session **795**
SetAlternateContent **1021**
SetAsDefault **44**
SetAt **139**
SetColumnDefaults **290**
SetColumnProperty **140**
SetColumns **584**
SetCriteriaForm **584**
SetDocEvent **1021**
SetDocInfo **1012**
SetDocRelation **1022**
SetDocument **455**
SetFocus **1063**
SetParent **1217**
SetProject **516**
SetProperty **90**
SetSearch **1149**
SetSelection **1218**
SetStatusBarText **1237**
SetValue **1134**
SetVerbIcons **834**
SetVerbProperties **835**
SetViewMode **109**
Shadow **395**
ShadowDocuments **1322**
ShadowPathTemplate **397**
ShortName **171**
Show **926**
ShowAccessForm **456**
ShowChangeDisplayNameForm **874**
ShowCompatibleAppsForm **876**
ShowDefaultSettings **249**

ShowEditCriteriaForm **587**
ShowExportForm **878**
ShowExportFormAndAction **879**
ShowGroupCheckinForm **880**
ShowGroupCheckoutForm **882**
ShowHistoryForm **458**
ShowImportForm **1322**
ShowLaunchMethodChooserForm **885**
ShowLibrariesForm **250**
ShowLibraryChooserForm **252**
ShowLocationForm **889**
ShowMailForm **894**
ShowProfile **458**
ShowProfileAccessForm **895**
ShowProfileFlags **702**
ShowProfileForm **905**
ShowProgressWindow **766**
ShowProjectAccessForm **908**
ShowProjectChooserForm **918**
ShowProjectForm **921**
ShowProjectMembershipForm **925**
ShowProjectsMembershipForm **460**
ShowProperties **652**
ShowQuickRetrieveForm **928**
ShowRelatedItemsForm **930**
ShowSaveAsOptionsForm **931**
ShowSearchAccessForm **933**
ShowSearchCriteriaForm **943**
ShowSearchOptionsForm **946**
ShowSelectVersionsForm **301**
ShowSelectVersionTypeForm **950**
ShowsOnDesktop **234**
ShowStatusBar **1238**
ShowTemplateChooserForm **255**
ShowVersionChooserForm **463**
SID **767**
Size **101**
Skip **67**
SlowRetrieveText **768**
SmallImage **810**
SortingEnabled **768**
SPFFlags **702**
SQLInfo **1181**
SQLName **278**
SSFFlags **703**
Stretch **1047**
Style **731**
SubDocuments **519**
Subject **1265**
SubProjectOrder **525**
SubProjects **519**
SupervisorApp **235**
SupportedViewModes **105**
Suspend **1300**

T

TabStop **1151**
Terminate **854**
Text **1196**
TextAt **1166**
ThreadPosition **1269**
TimeOut **236**
Title **310**
To **1263**
ToolTip **172**
ToolTipText **810**
TransferNotifyCompletedReason **703**
Trustees **465**

Type **1110**
TypeTime **210**
Typist **1340**

U

UIContext **796**
UIContextSupported **106**
UncheckedTrigger **1066**
UndockedProfile **1341**
UnlockProfile **466**
UnlockVersion **657**
UnregisterClass.htm **144**
UpBitmap **1167**
Update **658**
UpdateMenuItem **32**
Upload **1310**
UploadFile **659**
UploadFileEx **671**
UploadFileExFlags **704**
UploadFromFile **1311**
UpperCase **1111**
UserName **398**
UserPrimaryGroup **398**
UseUNC **237**
Using E-Mail Integration **1250**
Using E-Mail Integration Step 1 **1250**
Using E-Mail Integration Step 2 **1252**
Using E-Mail Integration Step 3 **1255**

V

Validation **1112**
ValidForOpen **352**
ValidForPrint **353**
ValidOnProfile **238**
Value **1066**
VerbFlags **182**
Version **212**
VersionChooserFormFlags **704**
VersionEvent **844**
VersionID **1342**
VersionLabel **1343**
VersionPropertiesFlags **705**
Versions **1334**
VersionsAddon **1240**
VersionStatus **672**
VersionType **320**
View **769**
ViewApp **239**
ViewAvailable **770**
ViewChanged **771**
ViewFile **1239**
ViewMode **183**
ViewModeChanged **1135**
ViewVersion **1239**
Visible **1048**
VisibleRights **903**
Vscroll **1114**

W

WantReturn **1115**
WhereUsed **473**
WorkingVersion **469**