



# *DM API Reference Guide*

**Hummingbird Enterprise™ 2004**

## **DM 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, and RM 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**RESTRICTED RIGHTS LEGEND.** Unpublished rights reserved under the copyright laws of the United States and any other appropriate countries.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](http://www.hummingbird.com)

# Contents

## Preface

About This Guide	<b>xv</b>
Who Should Read This Guide	<b>xv</b>
How This Guide Is Organized	<b>xv</b>
Documentation Conventions	<b>xvi</b>
Related Documentation	<b>xvi</b>
Professional Services	<b>xvii</b>
Where to Go for Information	<b>xviii</b>
On the Web	<b>xviii</b>
On the DM Suite CD	<b>xviii</b>
Technical Support and Training	<b>xix</b>
North America Technical Support	<b>xix</b>
Asia/Pacific Technical Support	<b>xix</b>
Europe Technical Support	<b>xix</b>
Training	<b>xx</b>

## Chapter 1     The DM Architecture

The DM Multi-Tier Architecture	<b>2</b>
DM Functionality	<b>2</b>
Document Security Tokens	<b>4</b>
Transactions	<b>4</b>
DM Object Model	<b>4</b>
DM Objects	<b>4</b>
DM Forms	<b>5</b>
Types of DM Objects	<b>5</b>
Defining a DM Object	<b>6</b>
DM Support Objects	<b>6</b>
DM Query Objects	<b>7</b>
Document Objects	<b>7</b>

Child Objects of PCDDocObject Objects	<b>7</b>
The Logon Process	<b>7</b>
Getting a List of Available Libraries	<b>8</b>
Providing Library Access	<b>9</b>
DM Search Transactions	<b>11</b>
General Steps to Performing a Search	<b>12</b>
Retrieving Recently Edited Documents	<b>13</b>
Performing a Simple Search	<b>16</b>
Document Objects	<b>19</b>
Fetching a DM Document Object	<b>19</b>
Getting and Updating Trustee Information	<b>23</b>

## **Chapter 2     An Overview of the DM API**

The PCDClient Object	<b>30</b>
List of DM API Objects	<b>30</b>
Early and Late Binding	<b>31</b>
Early Binding	<b>31</b>
Late Binding	<b>31</b>
Methods and Properties Supported by DM API Objects	<b>32</b>
Tokens Supported by DM API Methods and Properties	<b>41</b>

## **Chapter 3     DM API Objects**

PCDASPFileUpload	<b>44</b>
PCDDocObject	<b>47</b>
PCDEnumPropertyLists	<b>49</b>
PCDError	<b>50</b>
PCDGetDoc	<b>51</b>
PCDGetForm	<b>55</b>
PCDGetLoginLibs	<b>56</b>
PCDGetStream	<b>58</b>
PCDLogin	<b>61</b>
PCDLookup	<b>62</b>
PCDNetAliasList	<b>68</b>

**PCDNetworkInfo** **69**  
**PCDPropertyList** **83**  
**PCDPropertyLists** **96**  
**PCDPutDoc** **97**  
**PCDPutStream** **102**  
**PCDRecentDoc** **103**  
**PCDSearch** **105**  
**PCDSQL** **107**  
**PCDTrusteeList** **125**

## **Chapter 4 DM API Methods and Properties**

**AddLogin** **128**  
**AddLoginLicensed** **132**  
**AddOrderByProperty** **135**  
**AddProperty** **137**  
**AddReturnMetaProperty** **138**  
**AddReturnProperty** **140**  
**AddSearchCriteria** **142**  
**AddSearchLib** **145**  
**AddTrustee** **147**  
**AddUserFilterCriteria** **149**  
**BeginGetBlock** **150**  
**BeginIter** **151**  
**BytesRead** **153**  
**BytesWritten** **154**  
**ClearOrderByProperties** **155**  
**ClearUserFilterCriteria** **156**  
**Clone** **157**  
**ColumnCount** **158**  
**Create** **159**  
**Delete** **160**  
**DeleteProperty** **161**

DeleteTrustee **163**  
EndGetBlock **164**  
ErrDescription **165**  
ErrNumber **166**  
Execute **167**  
Fetch **169**  
FetchTrustees **170**  
GetAliasList **174**  
GetAt **175**  
GetColumnCount **176**  
GetColumnName **177**  
GetColumnValue **178**  
GetCurrentPropertyName **179**  
GetCurrentPropertyValue **180**  
GetCurrentTrusteeFlags **181**  
GetCurrentTrusteeName **182**  
GetCurrentTrusteeRights **183**  
GetDBVendor **184**  
GetDomainList **185**  
GetDOCSUserName **187**  
GetDST **188**  
GetFailedLoginList **189**  
GetGroupList **191**  
GetGroupMembers **193**  
GetLoginLibrary **195**  
GetMetaPropertyValue **196**  
GetMetaRowsFound **198**  
GetNextKey **200**  
GetPrimaryGroup **203**  
GetProperties **204**  
GetProperty **205**

---

GetPropertyIndex **206**  
GetPropertyValue **209**  
GetPropertyValueByIndex **211**  
GetReturnProperties **213**  
GetReturnProperty **215**  
GetRowCount **216**  
GetRowsAffected **217**  
GetRowsFound **218**  
GetSearchCriteria **220**  
GetSize **221**  
GetSQLErrorCode **223**  
GetTrustee **225**  
GetTrusteeIndex **227**  
GetTrusteeRights **229**  
GetTrustees **230**  
 GetUserFullName **231**  
 GetUserGroups **233**  
 GetUserList **235**  
 GetValue **237**  
 GrantRight **238**  
 HasRight **240**  
 IsEmpty **242**  
 IsMemberOf **243**  
 NewEnum **245**  
 Next **246**  
 NextMetaRow **247**  
 NextProperty **248**  
 NextRow **249**  
 NextTrustee **251**  
 OnEndPage **252**  
 OnStartPage **253**

---

Read **254**  
ReleaseResults **255**  
Reset **256**  
RevokeRight **257**  
Seek **259**  
SetChunkFactor **260**  
SetComplete **262**  
SetDST **263**  
SetLibrary **265**  
SetLookupId **266**  
SetMaxRows **268**  
SetMetaRow **270**  
SetObjectType **272**  
SetOptions **274**  
SetProperties **276**  
SetProperty **277**  
SetReturnProperties **279**  
SetRow **280**  
SetSearchCriteria **282**  
SetSearchObject **284**  
SetTargetProperty **286**  
SetTrustee **288**  
SetTrustees **290**  
SetTrusteeRights **292**  
Skip **293**  
UnitName **294**  
UnitType **295**  
Update **297**  
UpdateTrustees **298**  
UserName **299**  
Write **300**

---

%ADD\_ATTACHMENT **302**  
%ATTACHMENT\_ID **303**  
%CHECKIN\_DATE **304**  
%CHECKOUT\_COMMENT **305**  
%CONTENT **306**  
%CONTENTS\_AFTER\_ITEM **308**  
%CONTENTS\_COPY\_CONTENTS **310**  
%CONTENTS\_DIRECTIVE **312**  
%CONTENTS\_ITEM **314**  
%CONTENTS\_MOVE\_AFTER **316**  
%CONTENTS\_MOVE\_DOWN **318**  
%CONTENTS\_MOVE\_TO\_TOP **320**  
%CONTENTS\_MOVE\_UP **322**  
%CONTENTS\_REORDER\_CONTENTS **324**  
%CONTENTS\_SRC\_PARENT **326**  
%CONTENTS\_SRC\_PARENT\_LIBRARY **327**  
%CONTENTS\_SRC\_PARENT\_VERSION **328**  
%CONTENTS\_DST\_PARENT **329**  
%CONTENTS\_DST\_PARENT\_LIBRARY **330**  
%CONTENTS\_DST\_PARENT\_VERSION **331**  
%CONTENTS\_PARENT **332**  
%CONTENTS\_PARENT\_VERSION **333**  
%CONTENTS\_WHERE\_USED **334**  
%COPYDOC **336**  
%COPYDOC\_LIBRARY **337**  
%COPYDOC\_VERSION **338**  
%DATA **339**  
%DELETE\_ALL **341**  
%DELETE\_EXPUNGE **342**  
%DELETE\_OPTION **343**

%DELETE\_PHYSICAL\_FILES **345**  
%DOCS\_LIBRARY\_NAME **346**  
%DOCUMENT\_NUMBER **348**  
%EFFECTIVE\_RIGHTS **350**  
%ELAPSED\_TIME **352**  
%ENCAPSULATION\_TYPE **354**  
%FILTER\_DISABLED\_ROWS **356**  
%FOLDERITEM\_LIBRARY\_NAME **357**  
%FORM\_APPLICATION **359**  
%FORM\_DEFAULT\_PRIMARY **361**  
%FORM\_LIST\_TYPE **363**  
%FORM\_NAME **365**  
%FORM\_PROFILE\_DEFAULTS **367**  
%FORM\_TITLE **369**  
%FT\_CHARACTER\_SET **371**  
%FT\_CONFIDENCE **373**  
%FT\_FORMAT **375**  
%FT\_MARKER\_LIST **377**  
%FT\_SCORE **379**  
%FT\_SMART\_DOCUMENT **381**  
%FT\_TIMESTAMP **383**  
%FT\_VCC\_LIST **385**  
%FT\_VCC\_RULES **387**  
%GET\_ALL RELATED **389**  
%GET\_LOCAL RELATED **391**  
%GET RELATED ITEMS **393**  
%GET\_REMOTE RELATED **395**  
%HAS\_SUBFOLDERS **397**  
%HITLIST **399**  
%ISTREAM\_STATSTG\_CBSIZE\_LOWPART **401**  
%LOCK **402**

%LOCK\_FOR\_CHECKOUT **403**  
%LOOKUP\_ID **404**  
%MAKE\_READ\_ONLY **406**  
%MAXDAYS **408**  
%NUM\_COMPONENTS **409**  
%OBJECT\_IDENTIFIER **410**  
%OBJECT\_TYPE\_ID **412**  
%ORDER\_BY **413**  
%PCD\_DELETEVERSION **414**  
%PCD\_NEW\_VERSION **415**  
%PCD\_NEWSUBVERSION **416**  
%PCD\_PARM\_HTML\_RENDERING **417**  
%PCD\_PARM\_LIB\_SETTINGS **419**  
%PCD\_UPDATE\_VERSION **420**  
%PR\_ACCESS\_CONTROL **421**  
%PR\_CONTENT\_COPY **423**  
%PR\_CONTENT\_DELETE **425**  
%PR\_CONTENT\_EDIT **427**  
%PR\_CONTENT\_RETRIEVE **429**  
%PR\_CONTENT\_VIEW **431**  
%PR\_EDIT **433**  
%PR\_VIEW **435**  
%PRIMARY\_KEY **437**  
%PROFILE **438**  
%PROPERTYNAME **440**  
%PROPERTYTYPE **442**  
%PUBLISH\_VERSION **444**  
%QS\_DELETE **446**  
%QS\_EDIT **448**  
%QS\_VIEW **450**  
%RECENTACTIVITYDATE **452**

%RECENTACTIVITYTIME **454**  
%RELATED\_REMOTE\_LIBS **455**  
%REMOVE\_READ\_ONLY **456**  
%RENDITION\_TYPE **459**  
%RIGHT8 **461**  
%RIGHT9 **463**  
%SCORE\_GRAPHIC **465**  
%SCORE\_PERCENT **467**  
%SEARCH **469**  
%SECURITY **471**  
%STATUS **472**  
%TARGET\_LIBRARY **474**  
%TITLE **475**  
%TRUSTEE\_ID **477**  
%TRUSTEE\_RIGHTS **479**  
%TRUSTEE\_TYPE **481**  
%TRUSTEES\_ADD **483**  
%TRUSTEES\_REMOVE **485**  
%TRUSTEES\_SET **487**  
%TRUSTEES\_UPDATE **489**  
%UNLOCK **491**  
%UNPUBLISH\_VERSION **492**  
%USER\_ID **494**  
%VERIFY\_ONLY **495**  
%VERSION\_AUTHOR **497**  
%VERSION\_COMMENT **498**  
%VERSION\_DIRECTIVE **500**  
%VERSION\_ID **502**  
%VERSION\_LABEL **504**  
%VERSION\_TO\_INDEX **505**  
%VERSION\_TYPIST **506**

%VISIBLE **507**



## About This Guide

This guide describes the application programming interface (API) that is available as part of DM. It identifies each of the objects comprising the API and discusses the methods and properties that each object supports.

## Who Should Read This Guide

Users who extend DM functionality by creating their own custom enhancements will use the *DM API Reference Guide* to show them how their extensions can operate seamlessly with the base product. This will also be the case for software developers who create their own software products for specialized applications that address unique user needs that are not addressed by DM.

## How This Guide Is Organized

This book has four chapters.

- Chapter 1: The DM Architecture—This chapter describes how the API and other components of DM work together.
- Chapter 2: An Overview of the DM API—This chapter describes the overall structure of DM.
- Chapter 3: DM API Objects —This chapter describes the objects in the API and lists the methods and properties that each supports.
- Chapter 4: DM API Methods and Properties —This chapter describes each of the methods and properties that are supported by DM API objects.
- Chapter 5: DM API Tokens —This chapter describes each of the tokens that are supported by DM API methods.

# Documentation Conventions

This book uses the following fonts and styles to indicate different types of information.

Convention	Meaning
<i>Italic</i>	Indicates a new term or variable in a command line. For example, replace <i>filename</i> with the name of a file.
Monospaced font	Indicates a file, directory, drive or command name, program code, or other text that appears on the computer screen. For example, the default library for DM is usually c:\Program Files\ Hummingbird\DM Server.
<b>Bold</b>	In instruction steps, indicates information you must type. In text, indicates emphasis.
>	Separates items on more than one cascading menu or successive choices of icons or program groups.
Cross-reference	Click on these links to be taken to related information in the document.

## Related Documentation

In addition to this manual, you may find the following documents helpful.

- *DM/RM Data Dictionary* - Describes the tables and columns that comprise the DM/RM SQL database.
- *DM Designer Guide* - Provides detailed instructions on how you can use the Designer component of DM to create and maintain forms used by your DM installation. It also describes how you can use Designer to customize the tables and columns that comprise the SQL database that DM uses to manage your Hummingbird environment.

# Professional Services

Hummingbird's Professional Services' consultants and educators successfully design, create, and implement powerful integrated solutions based on Hummingbird technology. They are dedicated to forging sustainable, long-term relationships with our clients and partners by delivering superior consulting, training, and education solutions.

Professional Services' engagements include all aspects of the system implementation life cycle, specifically:

- project management
- planning and process analysis
- requirements definition
- software installation and prototyping
- custom code development
- code reviews and walkthroughs
- integration with non-Hummingbird technology
- benchmarking
- data conversion and migration
- documentation knowledge transfer

Our consultants are available to help you build or enhance application components, high-level APIs, user interfaces and system administration utilities for Windows clients, Web clients, and server-side applications using various development environments.

If you are implementing enterprise portals, document and knowledge management, data integration and reporting, or ETL, Hummingbird Professional Services can help guide you to success. To find out more about Professional Services visit our Web site at <http://www.hummingbird.com/solutions/services> or send an e-mail to [getinfo@hummingbird.com](mailto:getinfo@hummingbird.com).

# Where to Go for Information

## On the Web

Our Web site at [www.hummingbird.com/support/dkm/](http://www.hummingbird.com/support/dkm/) carries the most up-to-date information about Hummingbird's document and knowledge management products. This information is presented in individual technical bulletins and in Knowledge Base Solutions, each dealing with a specific topic that is not covered in our manuals or that updates information in the manuals. Before installing or upgrading DM, we suggest you browse through the bulletins or search the Knowledge Base for items that may be pertinent to your installation. For WebSupport, you will be asked to enter your user name and password for authentication. If you have not registered for a WebSupport account, you can do so at [www.hummingbird.com/support/dkm/registration.html](http://www.hummingbird.com/support/dkm/registration.html).

## On the DM Suite CD

In addition to software, the DM Suite CD houses important documentation and ancillary files to assist you install and use DM.

### Release Notes

The DM Release Notes for the current version reside in the ...\\Documentation\\Readme folder on the CD. The release notes contain information that came to light after the documentation was printed. Hyperlinks to known bugs and workarounds are also listed in the Release Notes.

### Installing Acrobat Reader

The DM product line's online manuals, which are published in PDF format, are provided on the DM Suite CD. To read these manuals, you will need a copy of Adobe Acrobat Reader installed on your machine. A setup program for Acrobat Reader 5.0 is provided on the DM CD. If you do not already have a copy, follow the instructions below.

- 1** Insert the DM Suite CD into your computer's CD-ROM drive.
- 2** The DM product line installation menu will be displayed. Click Documentation.
- 3** Click Install Acrobat Reader. The Acrobat Reader version 5.0 install program will launch.

- 4** Follow the on-screen instructions and install the program. When complete, exit the DM Product Line Installation menu.

To access the entire DM documentation set:

- 1** Insert the DM Suite CD into your computer's CD-ROM drive.
- 2** The DM Product Line Installation menu will be displayed. Click Documentation.
- 3** Select Browse Documentation. Double-click the Online Manuals directory.
- 4** Double-click HummingbirdDMBooks.html. Your Internet browser will launch and a menu listing the DM documentation set will appear.
- 5** To open a document, click the document name.

## Technical Support and Training

### North America Technical Support

124 Marriott Drive  
Tallahassee, FL 32301 USA  
Telephone: 800 486 0095 from 8:00 A.M. to 8:00 P.M. ET  
Fax: 850 942 8085  
E-Mail: [supportTLH@hummingbird.com](mailto:supportTLH@hummingbird.com)

### Asia/Pacific Technical Support

Level 12  
80 Mount Street  
North Sydney, NSW 2060, Australia  
Telephone: 61 2 9923 2011  
Fax: 61 2 9922 3097  
E-Mail: [supportsyd@hummingbird.com](mailto:supportsyd@hummingbird.com)

### Europe Technical Support

20 rue Thérèse  
75001 Paris, France  
Telephone: 33 1 55 35 96 80  
Fax: 33 1 42 61 31 87  
E-Mail: [support.eu.kmdm@hummingbird.com](mailto:support.eu.kmdm@hummingbird.com)

## **Training**

Hummingbird's Education Services offers courses at authorized training centers worldwide. For more information or to register for classes, contact Hummingbird Education Services:

Telephone: 613 238 1761

E-Mail: education@hummingbird.com

Internet Address: [www.hummingbird.com/education](http://www.hummingbird.com/education)

# 1

# The DM Architecture

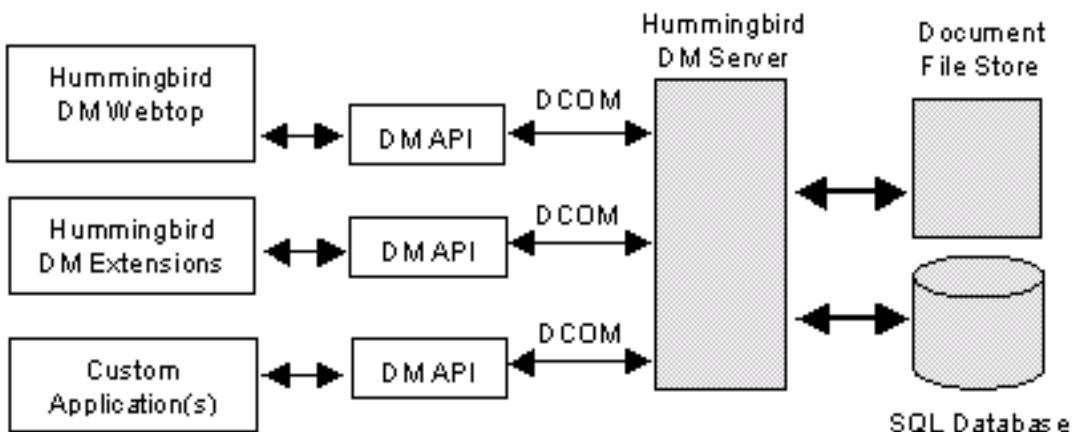
## In This Chapter

This chapter provides an overview of the DM functionality, illustrating how the major components of DM are designed to work with one another. Code samples illustrate how custom applications can perform many basic operations, such as allowing user access and performing searches.

# The DM Multi-Tier Architecture

## The DM Multi-Tier Architecture

DM is an application that runs on Windows NT/2000/XP Pro systems. As shown in the illustration below, the DM API provides a basic set of operations that interface with the DM Server. Client applications—whether the DM Webtop, any of the DM Extensions, or custom software you might develop—extend the DM API by providing a user interface and any other functionality appropriate to the new application.

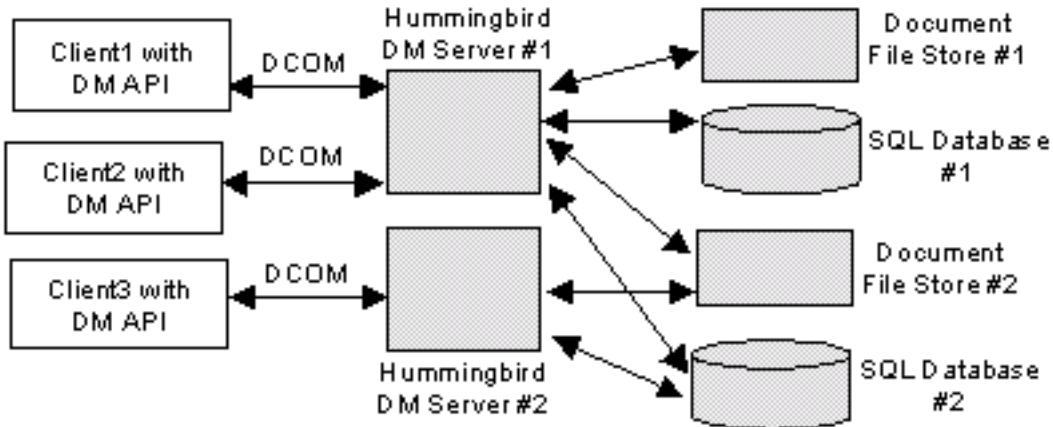


## DM Functionality

The diagram below illustrates how multiple clients can connect to a single DM Server and how a DM Server can connect to multiple DM repositories. A DM repository is comprised of the Document File Store, the SQL database, and the full-text indexer.

# DM Functionality

The DM Server is a transaction server, similar to Microsoft SQL Server.



The DM Server manages database connections via a connection pool. A client application connects to the DM Server via DCOM (Distributed Component Object Model) interfaces that are typically free threaded. The DM executable file (DOCSFusion.EXE) loads COM objects configured in the registry. Once connected, users can connect to any repositories to which they have access rights.

DM is stateless, so the client does not remain connected to the DM Server. A client application submits a transaction using a document security token (DST), and DM then determines the validity of the DST. If the DST is valid, the DM Server processes the transaction and returns the results to the client software. No state information is kept by the DM Server about the client.

*Note:* DM enforces document security rights. Users cannot access objects (for example, documents or folders) to which they do not have rights.

In DM, library maintenance tasks such as archiving and full-text indexing occur in the background, and there is no native user interface (UI). (To simplify these diagrams, DM Indexer Servers are not shown.) Developers must create user interfaces to connect to the DM Server.

## **Document Security Tokens**

A document security token (DST) is initially constructed by DM when a user logs on via a client application to the server and is required for DM transactions. The DST can be used across multiple DM Servers—when the user logs on to other DM Servers to access other Document libraries, the server information is appended to the DST—or the client application can use multiple DSTs.

## **Transactions**

With DM, multiple libraries can be configured for each server to manage, and transactions can occur on these specified libraries. The libraries available to a DM Server are configured via the DM Server Manager program and the PCDOCS.INI file. The client application must handle multiple library access and take into account issues such as that of incompatible forms.

*Caution:* You have direct access to the SQL database, allowing you to implement many operations directly by use of the computer language you use for your application. For example, specialized terms you implement can be managed by accessing the TERMINOLOGY table in the SQL database.

It is recommended that you do not use this capability. Using the appropriate DM API objects and methods will insure that the correct security controls are enforced and that other internal database relationships are maintained.

## **DM Object Model**

DM provides the following document management services:

- Search for documents stored in DM file stores.
- Create, delete, and update document profiles.
- Modify security on profiles and folders.
- Upload files to and download files from DM file stores.

## **DM Objects**

DM objects are low-level and general objects that presuppose a knowledge of how the DM environment operates (for example, how security rights are enforced). It is also important for DM API users to have a thorough understanding of both client and server software, as well as how they interact.

# Types of DM Objects

The DM Object Model consists of Distributed Component Object Model (DCOM) objects that are exposed via DM. Operations on DM objects map to operations on tables and typically require several properties to be set to define the objects. The object properties after instantiation may differ depending on how the object was created.

When building a DM client application, you will use the PCDClient type library (pcdclient.dll) and development tools such as C++, Visual Basic, Java, or Active Server Pages.

*Note:* Examples in this document show sample code written in Visual Basic. Similarly, this document refers to API items using Visual Basic nomenclature in preference to terminology more common to C++, Java, or other programming language environments. For example, the term "object" is generally used in preference to "class" or "interface." Also, "method" is used in place of "function," and "property" is used in place of "attribute."

**DM Forms** DM forms include those used as profile forms, query-by-example and other search forms, results lists, and lookups. These forms are fully customizable using DM Designer. Using DM Designer, you can also create your own forms.

With DM, forms are an abstraction layer for the SQL database that allows your applications to access the library database. DM supports Search, Query, Lookup, and Profile forms. DM uses a form definition to mediate client requests for data and read/write access to the database. The form definition maps the name of a field in the form to a SQL column in the database. Most client application calls are done using the name of the field as shown on the form, rather than the name of the column in the SQL database. DM uses the SQL\_INFO call, which is embedded in the form definition that resides in the SQL database. Each form must specify a primary table in the database as part of its form definition.

# Types of DM Objects

The DM API defines several types of objects:

- Support (read only), which facilitates the log on process.

# Types of DM Objects

- Query (read only), which is used for search, Quick Retrieve, and lookups.
- PCDDocObject (read/write), which implements the DM objects and modifies tables in the library database.
- Child objects of PCDDocObject (read/write), which get or set trustees, and perform file uploads and downloads.

## Defining a DM Object

In general, to define a DM object, you need to do the following:

- 1 Create the object in one of the following ways:

— Early bind:

```
Dim search as PCDSearch
```

— Late bind:

```
Dim search as Object
```

```
Set search = _  
CreateObject("PCDCIent.PCDSearch")
```

- 2 Set the DST using the [SetDST](#) method.

- 3 Specify a library for the object.

- 4 Specify the object type (the name of a form).

- 5 Set properties to identify the specific object.

- 6 Perform a method.

## DM Support Objects

The DM support objects provide a number of functions for the DM Server:

- Provide information required for the DM Server log on process.
- Log on to the DM Server.
- Return a document security token (DST) for later transactions.
- Provide additional general objects that do not require a DST.

The following are DM support objects:

# The Logon Process

- [PCDError](#)
- [PCDGetLoginLibs](#)
- [PCDLogin](#)
- [PCDNetAliasList](#)

## DM Query Objects

Query objects are used to search the library database. The query is constructed in memory, executed, and the results returned to the client application. The following are DM query objects:

- [PCDLookup](#)
- [PCDRecentDoc](#)
- [PCDSearch](#)

## Document Objects

Document objects are document management system (DMS) objects that represent database entities that can be modified. These entities include profiles, projects, and Quick Searches. The object identifier is:

[PCDDocObject](#)

## Child Objects of PCDDocObject Objects

The following are child objects of PCDDocObject objects:

- [PCDGetDoc](#)
- [PCDGetStream](#)
- [PCDPutDoc](#)
- [PCDPutStream](#)
- [PCDPropertyList](#)
- [PCDTrusteeList](#)

# The Logon Process

The following subsections describe the client logon process for DM.

# The Logon Process

## Getting a List of Available Libraries

The following example displays a list of libraries managed by the DM Server that the user can log on to.

```
Private Declare Function GetUserName Lib _
    "advapi32.dll" Alias "GetUserNameA" _
    (ByVal l pBuffer As String, nSize As Long) _
    As Long
Public OK As Boolean
Private Sub Form_Load()
    Dim sBuffer As String
    Dim lSize As Long
    Dim libname As String

    Dim i As Long
    Dim lim As Long
    Dim opOK As Boolean

    Dim Libs As New PCDGetLogonLibs

    ' Initialize public variables.
    OK = False
    dst = ""
    user = ""
    library = ""
    group = ""
    ' Get the user we are currently logged on as.
    sBuffer = Space$(255)
    lSize = Len(sBuffer)
    Call GetUserName(sBuffer, lSize)
    If lSize > 0 Then
        txtUserName.Text = Left$(sBuffer, lSize)
    Else
        txtUserName.Text = vbNullString
    End If

    ' Get the list of available libraries
    ' and fill in combo box.
    Libs.Execute
    lim = Libs.GetSize() - 1
    opOK = Libs.ErrNumber = 0
    LibCombo.Text = ""

    For i = 0 To lim
```

# The Logon Process

```
If (opOK) Then
    Libname = Libs.GetAt(i)
    opOK = opOK And Libs.ErrNumber = 0
    If i = 0 Then LibCombo.Text = Libname
    LibCombo.AddItem Libname
End If
Next
' List of Network logon types.
NetworkType.AddItem "Network Bindery"
NetworkType.AddItem "Network NDS"
NetworkType.AddItem "Microsoft Network"
NetworkType.Text = ""

End Sub

Private Sub cmdCancel_Click()
    OK = False
    Me.Hide
End Sub
```

## Providing Library Access

The next subroutine creates the [PCDLogin](#) object. It appends a network alias to a document security token (DST). The following algorithm is used:

- 1** Create the [PCDLogin](#) object.
- 2** Call the [AddLogin](#) method, which provides the logon mode, logon location, user name, and password.
- 3** Call the [Execute](#) method.
- 4** Call the [GetDST](#) method.

*Note:* If there is no network alias, the user name and Attaché password are used. Also, a second logon may be needed when file access occurs, such as when the user logs on to the DM Server or when a valid user account on a particular DM Server is required. A second logon may also be required if a network alias to a DM Server is not set in Library maintenance.

The following example illustrates how you can provide Library logon support for your users:

```
Private Sub cmdOK_Click()
    Dim Login As New PCDLogin
    Dim f1 As Object
    Dim i As Long
```

# The Logon Process

```
Dim l im As Long
Dim LogonType As Integer
Dim emsg As String

If (NetworkType = "Network Bindery") Then
    LogonType = 1
Else If (NetworkType = "Network NDS") Then
    LogonType = 2
Else If (NetworkType = "Microsoft Network") Then
    LogonType = 8
Else
    LogonType = 0
End If

If (Domain.Text <> "" And LogonType <> 0) Then
    'The Network and Domain are specified, so
    'log on to the network.
    Logon.AddLogin LogonType, Domain.Text, _
        txtUserName.Text, txtPassword.Text
Else
    'Just log on as a Library user.
    Logon.AddLogin 0, Library, _
        txtUserName.Text, txtPassword.Text
End If
OK = (Logon.ErrNumber = 0)

If (OK) Then
    Logon.Execute
    OK = (Logon.ErrNumber = 0)
End If

If OK Then
    'Set the public variables others will
    'need and then hide.
    Library = Logon.GetLoginLibrary()
    user = Logon.GetDOCSUserName()
    group = Logon.GetPrimaryGroup()
    dst = Logon.GetDST
    Me.Hide
Else
    'Unable to log on.
    emsg = "Error logging on: " & _
        Logon.ErrNumber & String(1, 13) & _
        String(1, 10) & Logon.ErrDescription
    MsgBox emsg, , "Logon"
    txtPassword.SetFocus
```

# DM Search Transactions

```
txtPassword.Sel Start = 0
txtPassword.Sel Length = Len(txtPassword.Text)
End If

'Dump any available information for
'debugging purposes.
Set fl = _
CreateObject("PCDCList.PCDNetAliasList")
Set fl = Login.GetFailedLoginList

lim = fl.GetSize() - 1
For i = 0 To lim
Debug.Print "FAILED List: "; _
fl.UnitName(i) & ";" & fl.UnitType(i) _
& ";" & fl.UserName(i)
Next

Set fl = Nothing
Set fl = Login.GetAliasList

lim = fl.GetSize() - 1

For i = 0 To lim
Debug.Print "Alias List: "; _
fl.UnitName(i) & ";" & fl.UnitType(i) _
& ";" & fl.UserName(i)
Next

Set fl = Nothing
Debug.Print "User: "; Login.GetDOCSUserName() _
& " Primary Group: " & _
Login.GetPrimaryGroup() & " LoginList: " _
& Login.GetLoginList()

End Sub
```

# DM Search Transactions

A typical search session involves the following:

- 1** A user enters search criteria in a user interface from which search data is extracted. Most commonly, the user interface is a form.
- 2** A search request is sent from the client.

# DM Search Transactions

**3** The DM Server does the following:

- Captures search criteria, and immediately returns a “no results” message to the client.
- Performs the search on requested databases and full-text repositories.
- Collects the results set and sorts it as user specified.
- Caches the results of the search and returns them to the client when requested.

The cached results can be deleted by the client using the `ReleaseResults` method. Also, results may time out based on administration settings.

The following sections describe various ways to execute search transactions in a DM environment.

## General Steps to Performing a Search

The following are the general steps in performing a search in the DM environment:

- 1** Create the search object.
- 2** Set the DST.
- 3** Add a search library (or multiple libraries).
- 4** Set the search object to the name of the search form.
- 5** Add return properties. Properties refer to the field names in the form. You need to specify the fields to return in the results set. Alternately, you can set an entire collection of properties using [SetReturnProperties](#).
- 6** Add search criteria, specifying a field name in the form and its value.
- 7** Add a property by which the return properties are to be ordered.
- 8** Set the maximum number of rows to return in the results set.
- 9** Optionally, set a chunk factor, which is the number of rows to return from the server results set into the local cache in one

# DM Search Transactions

operation. The default is 10. You do not need to repeat the execution of the search to get the next chunk. DM automatically does that for you.

## 10 Execute the search.

For the search results returned, you can:

- Obtain the number of rows found.
- Get all the returned properties.
- Iterate through the rows.
- Get values for properties returned by the search.
- Release the results set to free server memory.

## Retrieving Recently Edited Documents

The following example shows how to retrieve a list of documents that the user has recently edited.

```
Private Sub cmdCancel_Click()
    Unload Me
End Sub

Private Sub Form_Load()
    Dim rec As New PCDRecentDoc
    Dim i As Long
    Dim lIm As Long
    Dim row As String

    Screen.MousePointer = vbHourglass

    rec.SetDST DST
    rec.AddSearchLibrary
    ' This example uses the CYD_DEFPROF search form.
    rec.SetSearchObject "CYD_DEFPROF"
    rec.AddSearchCriteria "TYPE_ID", Chr(34) _
        & user & Chr(34)

    rec.AddReturnProperty "DOCNUM"
    rec.AddReturnProperty "SYSTEM_ID"
    rec.AddReturnProperty "APP_ID"
```

# DM Search Transactions

```
rec.AddReturnProperty "LASTEDI TDATE"
rec.AddReturnProperty "DOCNAME"
rec.AddReturnProperty "TYPIST_ID"
rec.AddReturnProperty "STATUS"

rec.AddOrderByProperty "LAST_EDI T_DATE", 0
rec.Execute

If (rec.ErrNumber <> 0) Then
    MsgBox "RecentEdit Failure on Execute: " & _
        rec.ErrNumber & " " & _
        rec.ErrDescription, , "Recent Edit"
Else
    lim = rec.GetRowsFound
    If lim > 50 Then lim = 50

    row = "DOCNAME" & Chr(9) & _
        "DOCNUM" & Chr(9) & "SYSTEM_ID" & _
        Chr(9) & "APP_ID" & Chr(9) & _
        "LAST_EDI T_DATE" & Chr(9) & _
        "TYPIST_ID" & Chr(9) & "STATUS"

    reGrid.Cols = 8
    reGrid.AddItem row
    i = 1
    While i <= lim
        rec.NextRow
        If (rec.ErrNumber <> 0) Then
            MsgBox "RecentEdit Failure on " & _
                "NextRow: " & rec.ErrNumber & " " & rec.ErrDescription, , "Recent Edit"
        Else
            row = rec.GetPropertyVal ue("DOCNAME") & Chr(9)
            row = row & _ rec.GetPropertyVal ue("DOCNUM") & Chr(9)
            row = row & _ rec.GetPropertyVal ue("SYSTEM_ID") & Chr(9)
            row = row & _ rec.GetPropertyVal ue("APP_ID") & Chr(9)
            row = row & _ rec.GetPropertyVal ue("LASTEDI TDATE") & Chr(9)
```

# DM Search Transactions

```
    row = row & _
        rec.GetPropertyValue("TYPE_ID") _
        & Chr(9)
    row = row & _
        rec.GetPropertyValue("STATUS")
    reGrid.AddItem row
End If
i = i + 1
Wend
rec.ReleaseResults
End If
Screen.MousePointer = vbDefault
End Sub

Private Sub Form_Resize()
Dim i As Long
Dim cw As Long
reGrid.Width = Width - 100
cw = reGrid.Width / reGrid.Columns
For i = 0 To reGrid.Columns - 1
    reGrid.ColumnWidth(i) = cw
Next
End Sub

Private Sub reGrid_Click()
If reGrid.Row <= 0 Then
    MsgBox "Select valid row"
    Exit Sub
End If

reGrid.Row = reGrid.RowSel
reGrid.Col = 1
txtSelDocNumber = reGrid.Text
docnumber = reGrid.Text

End Sub

Private Sub reGrid_DblClick()
Dim c
Dim dn As Variant
c = reGrid.Col
reGrid.Col = 1
dn = reGrid.Text

If IsNumeric(dn) Then
    ' Extract routine in frmVersions.Form_Load
End If
End Sub
```

# DM Search Transactions

```
' Versions (dn)
End If
End Sub
```

## Performing a Simple Search

The following example demonstrates how to do a simple search.

```

.
.

Private Sub cmdCancel_Click()
    Me.Hide
End Sub

Private Sub cmdDoLookup_Click()

    cboDocType.CLEAR
    Call LookupDocType(cboDocType)
End Sub

Private Sub cmdSearch_Click()
    Dim rec As New PCDSearch
    Dim i As Long
    Dim count As Long
    Dim row As String

    Screen.MousePointer = vbHourglass

    rec.SetDST DST
    rec.AddSearchLibrary
    ' This example uses the CYD_DEFPROF form.
    rec.SetSearchObject "CYD_DEFPROF"
    If txtAuthor <> "" Then
        rec.AddSearchCriteria "AUTHOR_ID", _
            Chr(34) & txtAuthor & Chr(34)
    End If
    If txtApp <> "" Then
        rec.AddSearchCriteria "APP_ID", Chr(34) _
            & txtApp & Chr(34)
    End If
    If cboDocType <> "" Then
        rec.AddSearchCriteria "TYPE_ID", Chr(34) _
            & cboDocType & Chr(34)
    End If

    rec.AddReturnProperty "DOCNUM"
    rec.AddReturnProperty "APP_ID"
```

# DM Search Transactions

```
rec.AddReturnProperty "SYSTEM_ID"
rec.AddReturnProperty "TYPE_ID"
rec.AddReturnProperty "LASTEDITDATE"
rec.AddReturnProperty "DOCNAME"
rec.AddReturnProperty "AUTHOR_ID"
rec.AddReturnProperty "STATUS"
' NOTE: This search should return data sorted
' by last edit then by document number.
rec.AddOrderByProperty "LAST_EDIT_DATE", 0
rec.AddOrderByProperty "DOCNUM", 1

rec.Execute

If (rec.ErrNumber <> 0) Then
    MsgBox "Search Failure on Execute: " &
        rec.ErrNumber & " " & rec.ErrDescription, _
        , "Search"
Else
    count = rec.GetRowsFound
    txtCount = Str(count)
    If count > 50 Then count = 50
    row = "DOCNAME" & Chr(9) & "DOCNUM" _
        & Chr(9) & "SYSTEM_ID" & Chr(9) & _
        "APP_ID" & Chr(9) & "TYPE_ID" & Chr(9) _ 
        & _ "LAST_EDIT_DATE" & Chr(9) _ 
        & "AUTHOR_ID" & Chr(9) & "STATUS"

    reGrid.Clear
    reGrid.Cols = 8
    reGrid.Rows = 0
    reGrid.AddItem row

    i = 1
    While i <= count
        rec.NextRow
        If (rec.ErrNumber <> 0) Then
            MsgBox "Search Failure on NextRow: " &
                rec.ErrNumber & " " & rec.ErrDescription, , "Search"
        Else
            row = rec.GetPropertyVal ue("DOCNAME") _
                & Chr(9)
            row = row & _
                rec.GetPropertyVal ue("DOCNUM") _
                & Chr(9)
            row = row & _
```

# DM Search Transactions

```
    rec.GetPropertyValue("SYSTEM_ID") _  
    & Chr(9)  
    row = row & _  
    rec.GetPropertyValue("APP_ID") _  
    & Chr(9)  
    row = row & _  
    rec.GetPropertyValue("TYPE_ID") _  
    & Chr(9)  
    row = row & _  
    rec.GetPropertyValue("LASTEDITDATE") _  
    & Chr(9)  
    row = row & _  
    rec.GetPropertyValue("AUTHOR_ID") _  
    & Chr(9)  
    row = row & _  
    rec.GetPropertyValue("STATUS")  
    reGrid.AddItem row  
End If  
i = i + 1  
Wend  
rec.ReleaseResults  
End If  
  
Screen.MousePointer = vbDefault  
  
End Sub  
  
Private Sub cmdShowLookup_Click()  
    frmLookup.Show vbModal  
End Sub  
  
Private Sub Form_Resize()  
    Dim i As Long  
    Dim cw As Long  
    reGrid.Columns = 8  
    cw = reGrid.Width / reGrid.Columns  
    For i = 0 To reGrid.Columns - 1  
        reGrid.ColumnWidth(i) = cw  
    Next  
End Sub  
  
Private Sub reGrid_Click()  
    If reGrid.Row <= 0 Then  
        MsgBox "Select valid row"  
        Exit Sub  
    End If
```

# Document Objects

```
reGrid.Row = reGrid.RowSel  
reGrid.Col = 1  
txtSel.DocNumber = reGrid.Text  
docnumber = reGrid.Text  
  
End Sub
```

# Document Objects

The steps for working with the [PCDDocObject](#) object and other DM API objects are as follows:

- 1** Set the DST.
- 2** Set the object type property to the name of the form.
- 3** Set the library as a property of the object.
- 4** Set the object properties, each of which requires a name/value pair.
- 5** Once all the relevant properties have been set, create the object.

There are also methods for fetching information on the properties of an object as well as deletion and update methods.

## Fetching a DM Document Object

The following example fetches a DM document object.

```
Attribute VB_Name = "frmFetchDoc"  
Attribute VB_GlobalNameSpace = False  
Attribute VB_Creatable = False  
Attribute VB_PredeclaredId = True  
Attribute VB_Exposed = False  
Option Explicit  
Dim ret As Boolean  
Dim i As Long  
Dim FileName As String  
Dim FetchFlag As Boolean  
  
Private Sub cmdCheckIn_Click()
```

# Document Objects

```
Dim doc As New PCDDocObject

If docnumber = "" Or versionid = "" Then
    MsgBox "Check In requires that you set " _
        & "the document number and " _
        & "the version ID."
    Exit Sub
End If

doc.SetDST DST
doc.SetObjectType "cyd_defprof"
doc SetProperty "%TARGET_LIBRARY", Library
doc SetProperty "%OBJECT_IDENTIFIER", docnumber
doc SetProperty "%VERSION_ID", versionid
doc SetProperty "%STATUS", "%UNLOCK"
doc.Update

' Check for error.
Dim LongEnum As Long
LongEnum = doc.ErrNumber
If LongEnum <> 0 Then
    Dim strEDesc As String, strEnum As String
    strEDesc = doc.ErrDescription
    strEnum = CStr( LongEnum )
    MsgBox "Error " & strEnum & ": " & strEDesc
    'Handle the error...
End If

Set doc = Nothing
Set doc = New PCDDocObject
doc.SetDST DST
doc.SetObjectType "cyd_defprof"
doc SetProperty "%TARGET_LIBRARY", Library
doc SetProperty "%OBJECT_IDENTIFIER", docnumber
doc.Fetch

' Check for error.
Dim LongEnum As Long
LongEnum = doc.ErrNumber
If LongEnum <> 0 Then
    Dim strEDesc As String, strEnum As String
    strEDesc = doc.ErrDescription
    strEnum = CStr( LongEnum )
    MsgBox "Error " & strEnum & ": " & strEDesc
    'Handle the error...
End If
```

# Document Objects

```
txtStatus = doc.GetReturnProperty("STATUS")
Set doc = Nothing

If txtStatus = 0 Then
    cmdCheckOut.Enabled = True
    cmdCheckIn.Enabled = False

Else
    cmdCheckOut.Enabled = False
    cmdCheckIn.Enabled = True
End If

MsgBox "Status field in profile updated " _
& "to 0 (checked in)."
End Sub
.

.

.

Private Sub FetchButton_Click()
Dim rec As New PCDGetDoc
Dim i As Long
Dim lim As Long
Dim opOK As Boolean
Dim row As String

Dim gd As Object
Dim indata As Variant
Dim bdata() As Byte

If docnumber = "" Or versionid = "" Then
    MsgBox "Must specify the document number " _
& "and the version ID."
    Exit Sub
End If

FetchFlag = False
FileName = ""
rec.SetDST_DST
rec.AddSearchCriteria "%TARGET_LIBRARY", _
Library
rec.AddSearchCriteria "%DOCUMENT_NUMBER", _
docnumber
rec.AddSearchCriteria "%VERSION_ID", versionid

rec.Execute
```

# Document Objects

```
If (rec.ErrNumber <> 0) Then
    MsgBox "Fetch Doc Failure on Execute: " & _
        rec.ErrNumber & " " & rec.ErrDescription, _
        , "Fetch Doc"
Else
    lim = rec.GetRowsFound
    i = 1
    While i <= lim
        rec.NextRow
        If (rec.ErrNumber <> 0) Then
            i = lim + 1
            MsgBox "Fetch Doc Failure on Execute: " & _
                & rec.ErrNumber & " " & _
                rec.ErrDescription, , "Fetch Doc"
        End If
        If (i = 1) Then
            Set gd = rec.GetPropertyVal ue("%CONTENT")
            On Error GoTo saveFailed
            fileName = "c:\temp\" & _
                Trim(docnumber) & "_" & _
                Trim(versionid) & ".doc"
            Open fileName For Binary Access Write _
                As #1
            bdata() = ""
            bdata() = gd.Read(1024)
            While (gd.BytesRead > 0)
                Put #1, , bdata()
                bdata() = ""
                bdata() = gd.Read(1024)
            Wend
            Close #1
            GoTo saveDone
        saveFailed:
            MsgBox "Fetch Doc Failure File Open: " & _
                , "Fetch Doc"
        savecancelled:
            On Error Resume Next
            saveDone:
        End If
        i = i + 1
    Wend
```

# Document Objects

```
End If  
FetchFlag = True  
MsgBox "Exported document content to " _  
    & "designated file: " & FileName  
  
End Sub
```

## Getting and Updating Trustee Information

The following example updates the trustees for a profiled document object.

```
Attribute VB_Name = "frmTrustees"  
Attribute VB_GlobalNameSpace = False  
Attribute VB_Creatable = False  
Attribute VB_PredeclaredId = True  
Attribute VB_Exposed = False  
Option Explicit  
  
Dim bRet As Boolean  
Dim Status As String  
Dim DocName As String  
Dim DefaultRights As Long  
Dim EffectiveRights As Long  
Dim AccessControl As Boolean  
Dim i As Integer  
Dim UserOrGroup As String  
Dim PDoc As Object  
Dim TrusteeList As PCDTrusteeList  
  
Private Function _  
    GetTrusteesforProfile(docnumber As String) _  
        As Boolean  
  
    Dim docsfound As Long  
  
    On Error GoTo ErrorHandler  
    GetTrusteesforProfile = False  
  
    If Val(docnumber) <= 0 Then Exit Function  
  
    Dim pclient As PCDSearch
```

# Document Objects

```
Set pclient = _
    CreateObject("PCDClient.PCDSearch")

pclient.SetDST DST
pclient.AddSearchLibrary
'This example uses the "def_qbe" form.
pclient.SetSearchObject "def_qbe"

'Set the properties to be returned. These
'are the properties to be displayed in the
'search results page.
pclient.AddReturnProperty("DOCNAME")
pclient.AddReturnProperty("DOCNUM")
pclient.AddReturnProperty("AUTHOR_ID")
pclient.AddReturnProperty("STATUS")
pclient.AddReturnProperty("VERSION")
pclient.AddReturnProperty("SECURITY")
pclient.AddReturnProperty("VERSION_ID")
pclient.AddOrderByProperty "VERSION_ID", True
pclient.AddSearchCriteria "DOCNUM", docnumber

pclient.Execute

'Check for error.
Dim lngEnum As Long
lngEnum = pclient.ErrNumber
If lngEnum <> 0 Then
    Dim strEDesc As String, strEnum As String
    strEDesc = pclient.ErrDescription
    strEnum = CStr(lngEnum)
    MsgBox "Error " & strEnum & ": " & strEDesc
    'Handle the error...
End If

docsfound = pclient.GetRowsFound
txtDocsFound = Str(docsfound)
If docsfound = 0 Then
    Set pclient = Nothing
    Exit Function
End If

pclient.NextRow

DefaultRights = _
    pclient.GetPropertyVal("SECURITY")
DocName = pclient.GetPropertyVal("DOCNAME")
```

# Document Objects

```
txtDefaultRights = Str(DefaultRights)
pclient.ReleaseResults

Set PDoc = _
    CreateObject("PCDClient.PCDDocObject")
PDoc.SetDST DST
PDoc SetProperty "%TARGET_LIBRARY", Library
PDoc SetPropertyType("DEF_PROF")
PDoc SetProperty "%OBJECT_IDENTIFIER", _
    docnumber

PDoc.Fetch

' Check for error.
Dim IngEnum As Long
IngEnum = PDoc.ErrNumber
If IngEnum <> 0 Then
    Dim strEDesc As String, strEnum As String
    strEDesc = PDoc.ErrDescription
    strEnum = CStr(IngEnum)
    MsgBox "Error " & strEnum & ": " & strEDesc
    ' Handle the error...
End If

EffectiveRights = _
    PDoc.GetPropertyVal("EFFECTIVE_RIGHTS")
AccessControl = _
    PDoc.HasRight("%PR_ACCESS_CONTROL", _
        EffectiveRights)
txtEffectiveRights = Str(EffectiveRights)
txtAccessControl = Str(AccessControl)

' Only display trustees if security is set.
If DefaultRights Then
    Call DisplayTrustees(PDoc, _
        TreeTrustees, AccessControl)
    If AccessControl Then
        cmdAddTrustee.Enabled = True
        cmdRemoveTrustee.Enabled = True
    End If
Else
    MsgBox "No security set for this profile."
End If

Set PDoc = Nothing
Set pclient = Nothing
```

# Document Objects

```
GetTrusteesforProfile = True
Exit Function

End If

ErrorHandler:
MsgBox "Unhandled Error: " & _
      Str(Err.Number) & " was generated by " & _
      & Err.Source & Chr(13) & Err.Description

End Function

.

.

Private Sub cmdAddTrustee_Click()

    ' Two ways to do this:
    ' 1) Set trustee on profile object and
    '     update trustees.
    Set PDoc = _
        CreateObject("PCDCClient.PCDDocObject")
    PDoc.SetDST DST
    PDoc SetProperty "%TARGET_LIBRARY", Library
    PDoc SetProperty("DEF_PROF")
    PDoc SetProperty "%OBJECT_IDENTIFIER", docnumber
    PDoc.FetchTrustees
    ' Set flags to 1 for group and 2 for user, but
    ' 0 may work.
    PDoc.SetTrustee cboTrustees.Text, 2, _
        Val(txtRights)

    ' Check for error.
    Dim IngEnum As Long
    IngEnum = PDoc.ErrNumber
    If IngEnum <> 0 Then
        Dim strEDesc As String, strEnum As String
        strEDesc = PDoc.ErrDescription
        strEnum = CStr(IngEnum)
        MsgBox "Error " & strEnum & ": " & strEDesc
        ' Handle the error...
    End If

    PDoc.UpdateTrustees

    ' Check for error.
    Dim IngEnum As Long
```

# Document Objects

```
lNgEnum = PDoc.ErrNumber
If lNgEnum <> 0 Then
    Dim strEDesc As String, strEnum As String
    strEDesc = PDoc.ErrDescription
    strEnum = CStr(lNgEnum)
    MsgBox "Error " & strEnum & ": " & strEDesc
    'Handle the error...
End If

PDoc.Update

'Check for error.
Dim lNgEnum As Long
lNgEnum = PDoc.ErrNumber
If lNgEnum <> 0 Then
    Dim strEDesc As String, strEnum As String
    strEDesc = PDoc.ErrDescription
    strEnum = CStr(lNgEnum)
    MsgBox "Error " & strEnum & ": " & strEDesc
    'Handle the error...
End If

Set PDoc = Nothing
MsgBox "Trustee " & cboTrustees.Text & _
       " has been added."
Exit Sub

'2) Add trustee to the trustees collection
'   and update trustees.
PDoc.FetchTrustees
TrusteeList.AddTrustee cboTrustees.Text, _
                     2, Val(txtRights)
PDoc.SetTrustees TrusteeList
PDoc.UpdateTrustees

'Check for error.
Dim lNgEnum As Long
lNgEnum = PDoc.ErrNumber
If lNgEnum <> 0 Then
    Dim strEDesc As String, strEnum As String
    strEDesc = PDoc.ErrDescription
    strEnum = CStr(lNgEnum)
    MsgBox "Error " & strEnum & ": " & strEDesc
    'Handle the error...
End If
```

# Document Objects

```
PDoc.Update  
  
' Check for error.  
Dim IngEnum As Long  
IngEnum = PDoc.ErrNumber  
If IngEnum <> 0 Then  
    Dim strEDesc As String, strEnum As String  
    strEDesc = PDoc.ErrDescription  
    strEnum = CStr( IngEnum )  
    MsgBox "Error " & strEnum & ": " & strEDesc  
    ' Handle the error...  
End If  
  
End Sub  
  
.
```

# 2

## An Overview of the DM API

### In This Chapter

This chapter describes the overall structure of the DM API, including an itemized list that shows the methods and properties that each object supports. It also presents information that applies across the entire API.

# The PCDClient Object

## The PCDClient Object

Your custom applications interact with the DM Server through a number of objects that are collectively referred to as the PCDClient objects. To use the PCDClient objects, you should have some understanding of the Distributed Component Object Model (DCOM).

## List of DM API Objects

The following is a list of the DM client objects that comprise the DM API.

[PCDDocObject](#)  
[PCDEnumPropertyLists](#)  
[PCDError](#)  
[PCDGetDoc](#)  
[PCDGetForm](#)  
[PCDGetLoginLibs](#)  
[PCDGetStream](#)  
[PCDLogin](#)  
[PCDLookup](#)  
[PCDNetAliasList](#)  
[PCDNetworkInfo](#)  
[PCDPropertyList](#)  
[PCDPropertyLists](#)  
[PCDPutDoc](#)  
[PCDPutStream](#)  
[PCDRecentDoc](#)  
[PCDSearch](#)

# Early and Late Binding

[PCDSQL](#)

[PCDTrusteeList](#)

## Early and Late Binding

You can create objects in the DM API using either early binding or late binding. Examples shown in this document will sometimes use early binding, and other times will use late binding. In custom applications you implement, you can usually use either, but there are a few instances where the text will indicate that one should be used in preference to the other.

### Early Binding

With early binding you create an object in a single step when you first dimension it. This may be the most appropriate way to create an object if you are going to use it immediately.

Early binding uses the New keyword within the Dim statement when the object is first dimensioned. For example:

```
Dim pDoc As New PCDCIent.PCDPutDoc
```

The New operator must be used on Visual Basic objects that have Private or Public `NotCreatable` instancing properties.

### Late Binding

With late binding you first dimension the item you want to create, often dimensioning it as a generic object. Later, when you are ready to use the object, you use the Set statement to instantiate the object in memory. For example:

```
Dim pObj As Object  
...  
Set pObj = CreateObject("PCDCIent.PCDLookup")
```

If your application uses the Microsoft Transaction Server, you must create objects using late binding with the `CreateObject` method. The Transaction Server cannot see instances of externally created objects that you generate with the New operator.

# Methods and Properties Supported by DM API Objects

Each of the DM API objects supports two or more methods or properties that perform the various tasks related to that object. The methods and properties that each DM API object supports are listed below.

Many PCDCLient objects support methods and properties with the same name. For example, all PCDCLient objects access the ErrNumber and ErrDescription properties.

To avoid repeating the information by discussing methods and properties with the objects they support, methods and properties are discussed separately. Chapter 3 presents DM API objects. Chapter 4 discusses the rich array of methods and properties available in the DM API.

## **PCDASPFileUpload**

Execute  
IsEmpty  
OnEndPage  
OnStartPage

## **PCDDocObject**

Create  
Delete  
Fetch  
FetchTrustees  
GetProperties  
GetProperty  
GetReturnProperties  
GetReturnProperty  
GetTrustee  
GetTrustees  
GrantRight

# Methods and Properties Supported by DM API Objects

[HasRight](#)  
[RevokeRight](#)  
[SetDST](#)  
[SetObjectType](#)  
[SetProperties](#)  
[SetProperty](#)  
[SetTrustee](#)  
[SetTrustees](#)  
[Update](#)  
[UpdateTrustees](#)

## **PCDEnumPropertyLists**

[Clone](#)  
[Next](#)  
[Reset](#)  
[Skip](#)

## **PCDError**

[ErrDescription](#)  
[ErrNumber](#)

## **PCDGetDoc**

[AddSearchCriteria](#)  
[Execute](#)  
[GetPropertyValue](#)  
[GetReturnProperties](#)  
[GetRowsFound](#)  
[GetSearchCriteria](#)  
[NextRow](#)  
[SetDST](#)  
[SetRow](#)

# Methods and Properties Supported by DM API Objects

[SetSearchCriteria](#)

[SetSearchObject](#)

**PCDGetForm**

[AddSearchLib](#)

[Execute](#)

[GetPropertyValues](#)

[SetDST](#)

[SetObjectType](#)

**PCDGetLoginLibs**

[Execute](#)

[GetAt](#)

[GetSize](#)

**PCDGetStream**

[BytesRead](#)

[GetPropertyValues](#)

[Read](#)

[Seek](#)

[SetComplete](#)

**PCDLogin**

[AddLogin](#)

[Execute](#)

[GetAliasList](#)

[GetDOCSUserName](#)

[GetDST](#)

[GetFailedLoginList](#)

[GetLoginLibrary](#)

[GetPrimaryGroup](#)

[SetDST](#)

# Methods and Properties Supported by DM API Objects

## PCDLookup

AddOrderByProperty  
AddSearchCriteria  
AddSearchLib  
AddUserFilterCriteria  
ClearOrderByProperties  
ClearUserFilterCriteria  
ColumnCount  
Execute  
GetMetaPropertyValue  
GetMetaRowsFound  
GetPropertyValueByIndex  
GetRowsFound  
GetSearchCriteria  
NextMetaRow  
NextRow  
ReleaseResults  
SetChunkFactor  
SetDST  
SetLookupId  
SetMaxRows  
SetMetaRow  
SetRow  
SetSearchCriteria  
SetSearchObject  
SetTargetProperty

## PCDNetAliasList

GetSize  
UnitName

# Methods and Properties Supported by DM API Objects

[UnitType](#)

[UserName](#)

**PCDNetworkInfo**

[GetDomainList](#)

[GetGroupList](#)

[GetGroupMembers](#)

[GetRowCount](#)

[GetUserFullName](#)

[GetUserGroups](#)

[GetUserList](#)

[GetValue](#)

[IsMemberOf](#)

[NextRow](#)

[SetDST](#)

**PCDPropertyList**

[AddProperty](#)

[BeginIter](#)

[DeleteProperty](#)

[GetCurrentPropertyName](#)

[GetCurrentPropertyValue](#)

[GetPropertyIndex](#)

[GetPropertyValue](#)

[GetSize](#)

[NextProperty](#)

**PCDPropertyLists**

[BeginIter](#)

[Execute](#)

[GetCurrentPropertyName](#)

# Methods and Properties Supported by DM API Objects

[GetCurrentPropertyValue](#)

[NewEnum](#)

[NextProperty](#)

[NextRow](#)

[SetChunkFactor](#)

[SetDST](#)

[SetObjectType](#)

[SetOptions](#)

[SetProperties](#)

[SetProperty](#)

## **PCDPutDoc**

[AddSearchCriteria](#)

[Execute](#)

[GetPropertyValue](#)

[GetReturnProperties](#)

[GetRowsFound](#)

[GetSearchCriteria](#)

[NextRow](#)

[SetDST](#)

[SetRow](#)

[SetSearchCriteria](#)

[SetSearchObject](#)

## **PCDPutStream**

[BytesWritten](#)

[GetPropertyValue](#)

[SetComplete](#)

[Write](#)

# Methods and Properties Supported by DM API Objects

PCDRecentDoc  
AddOrderByProperty  
AddReturnMetaProperty  
AddReturnProperty  
AddSearchCriteria  
AddSearchLib  
BeginGetBlock  
ColumnCount  
EndGetBlock  
Execute  
GetMetaPropertyValue  
GetMetaRowsFound  
GetPropertyValue  
GetPropertyValueByIndex  
GetReturnProperties  
GetRowsFound  
GetSearchCriteria  
NextMetaRow  
NextRow  
ReleaseResults  
SetChunkFactor  
SetDST  
SetMaxRows  
SetMetaRow  
SetReturnProperties  
SetRow  
SetSearchCriteria  
SetSearchObject

# Methods and Properties Supported by DM API Objects

## PCDSearch

AddOrderByProperty  
AddReturnMetaProperty  
AddReturnProperty  
AddSearchCriteria  
AddSearchLib  
BeginGetBlock  
ColumnCount  
EndGetBlock  
Execute  
GetMetaPropertyValue  
GetMetaRowsFound  
GetPropertyValue  
GetPropertyValueByIndex  
GetRowsFound  
NextMetaRow  
NextRow  
ReleaseResults  
SetChunkFactor  
SetDST  
SetMaxRows  
SetMetaRow  
SetReturnProperties  
SetRow  
SetSearchCriteria  
SetSearchObject

## PCDSQL

Execute  
GetColumnCount

# Methods and Properties Supported by DM API Objects

[GetColumnName](#)

[GetColumnValue](#)

[GetDBVendor](#)

[GetNextKey](#)

[GetRowCount](#)

[GetRowsAffected](#)

[GetSQLErrorCode](#)

[NextRow](#)

[ReleaseResults](#)

[SetDST](#)

[SetLibrary](#)

[SetRow](#)

## **PCDTrusteeList**

[AddTrustee](#)

[BeginIter](#)

[DeleteTrustee](#)

[GetCurrentTrusteeFlags](#)

[GetCurrentTrusteeName](#)

[GetCurrentTrusteeRights](#)

[GetSize](#)

[GetTrusteeIndex](#)

[GetTrusteeRights](#)

[NextTrustee](#)

[SetTrusteeRights](#)

# Tokens Supported by DM API Methods and Properties

## Tokens Supported by DM API Methods and Properties

Tokens are special identifiers that instruct the DM Server to perform specific actions. They are often used as a short-hand reference to an object that otherwise could only be described by a longer text string, such as a reference to a SQL table and column. For example, the %LOGIN\_DATE token can substitute for as a reference for the PEOPLE.LAST\_LOGIN\_DATE column of the SQL database.

Token identifiers are easily recognized. When used in application code, each token must be enclosed within double quote marks and must always be written in UPPERCASE. Each token begins with a percent sign (%), a syntax requirement that identifies them as DM system variables rather than local variables defined by the API program.

When used in application code, DM tokens perform one of three functions:

- Tokens set values either in the SQL database or in other variables. For example, the %CHECKOUT\_COMMENTS token can be used to identify a string that the program stores in the COMMENTS column of the CHECKOUT table.
- Tokens identify data items that API applications are to return when the application executes. For example, the %DATA token returns the data and metadata associated when an application program executes the PDCSQL operation.
- Tokens manipulate column values in SQL database tables. For example, the %CONTENTS\_MOVE\_TO\_TOP adjusts the items in a DM Folder so that the specified object is shown at the top of the presentation list and the other items in the folder are adjusted to lower positions in the list if necessary.

Chapter 5, “[DM API Tokens](#)” on page 301, lists the most commonly used DM tokens.

# Tokens Supported by DM API Methods and Properties

# 3

## DM API Objects

### In This Chapter

This chapter describes each of the DM objects, including their syntax, usage, and other related information.

# PCDASPFileUpload

## PCDASPFileUpload

Use this object only from scripts running inside Active Server Pages (ASP). This object is used to read from the ASP Request object and write to the supplied [PCDPutStream](#) object.

### Syntax

PCDASPFI L eUpI oad. *methodOrProperty*

### Usage

The PCDASPFileUpload object is a client-side interface (CSI) object used to extract form data (specifically, uploaded files) from the requested object when a form is submitted. When uploading a file, ENCTYPE is set to “multipart/form-data.” A server-side component is required to parse the information contained within the file. This utility is used in conjunction with the [PCDPutDoc](#) object to write the data stream to the document server.

To demonstrate the process, open the DM Webtop UPLOADDSP. ASP file. Locate the following line of code (where [...] markers indicate that the same line of code continues on the next line of the page):

```
<FORM NAME="upl oadForm" ACTION="<% = url %>?[...]  
=>%=Server. HTMLEncode(formArgs) %>" ENCTYPE=[...]  
mul ti part/form-data METHOD=post>  
  
<I NPUT type="button" Val ue="<% =getString(([...]  
i sl importVersi on)?" IMPORT_BUTTON1": "UPLOAD_[...]  
HEADER")%>" oncl i ck="submi tFunction()" >
```

Assuming that you are not using the DM Webtop Smart CheckIn/CheckOut feature, when the form is submitted, your Internet browser knows that you are uploading a file and will prompt you with an Input File dialog box that contains a Browse button. Select the file to be uploaded. The file itself is embedded as part of the form via the POST method. In UPLOADACT. ASP, the PCDASPFileUpload component is used to extract the file from the REQUEST object.

# PCDASPFileUpload

PCDASPFileUpload is a helper class/interface/object that supports the tying of the POST of an ASP Multipart/Form file into a [PCDPutStream](#). (The GET method is not supported in PCDPutStream, so always use POST.) It handles all the parsing of headers from the body content and streaming of the file that is to be uploaded into the DM [PCDPutStream](#) object.

The [OnStartPage](#) method initializes the COM interface pointers to all of the ASP objects, such as Session, Application, Request, Response, and Server. These are passed by the ASP engine from the interface pointer.

The [Execute](#) method reads from the ASP Request object that is obtained during [OnStartPage](#). It uses the ReadBinary method to write to the [PCDPutStream](#) object that the server supplies.

The [IsEmpty](#) method can be used to determine if the body of the content is empty, which would indicate that a zero-byte file was sent. [IsEmpty](#) looks at the posted stream, parses out the header, and determines if the document being sent is empty. If the content is empty, the success return value is set to TRUE, and the error state is set to PCD\_ERR\_ZERO\_BYTE\_UPLOAD. If the body content is not empty, the success return value is set to FALSE, and no error condition is set.

The [OnEndPage](#) method releases all the interface pointers acquired during the [OnStartPage](#) initialization and resets the error state.

## Example

The following is an example of creating an instance of the PCDASPFileUpload.

```
Dim pFile As Object  
.  
.  
.  
pFile = _  
Server.CreateObject("PCDCIent.PCDASPFileUpload")
```

# PCDASPFileUpload

## Related Items

See the following methods:

[Execute](#)  
[IsEmpty](#)  
[OnEndPage](#)  
[OnStartPage](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# PCDDocObject

## PCDDocObject

This object is one of the true workhorses of the DM API. Custom applications use it to manipulate Document objects. Document objects include such elements as Document Profiles, search forms, and versions.

### Syntax

PCDDocObject. *methodOrProperty*

### Example

The following is an example of creating an instance of the PCDDocObject.

```
    .  
    .  
    .  
pObj ect = _  
    Server. CreateObj ect("PCDCI i ent. PCDDocObj ect")  
    .  
    .  
    .
```

### Related Items

See the following methods:

[Create](#)  
[Delete](#)  
[Fetch](#)  
[FetchTrustees](#)  
[GetProperties](#)  
[GetProperty](#)  
[GetReturnProperties](#)  
[GetReturnProperty](#)  
[GetTrustee](#)  
[GetTrustees](#)  
[GrantRight](#)  
[HasRight](#)  
[RevokeRight](#)

# **PCDDocObject**

[SetDST](#)  
[SetObjectType](#)  
[SetProperties](#)  
[SetProperty](#)  
[SetTrustee](#)  
[SetTrustees](#)  
[Update](#)  
[UpdateTrustees](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# PCDEnumPropertyLists

## PCDEnumPropertyLists

This object allows you to iterate through collections of property lists. Most often used with documents and folders, PCDEnumPropertyLists also allows you to iterate through the property lists associated with collections of versions, root objects, and other items. For example, if a folder is deleted, all of the documents and folders it contained must have their properties updated to indicate that they are no longer contained in the deleted folder.

PCDEnumPropertyLists supports COM-standard enumeration interface methods. Every COM-standard enumeration object supports Clone, Next, Reset, and Skip methods.

### Syntax

PCDEnumPropertyLists. *methodOrProperty*

### Related Items

See the following methods:

[Clone](#)  
[Next](#)  
[Reset](#)  
[Skip](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# PCDError

## PCDError

This object is a base object for all other PCDClient objects. It provides common properties that you access through the other objects to get error information.

### Usage

You do not have to create an instance of this object. All other PCDClient methods access it directly.

### Example

All PCDClient objects can access the PCDError objects properties without creating a PCDError object in their application. Most of the sample code in this guide contains examples of how the [ErrNumber](#) and [ErrDescription](#) properties are accessed.

The section titled [Fetching a DM Document Object](#) in Chapter 1 is one of the many examples that illustrate how you can use PCDError properties in your custom applications.

### Related Items

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# PCDGetDoc

## PCDGetDoc

This object is used to manage the retrieval of a set of physical files that comprise the components of one version of a document. You use this object just as you would the [PCDSearch](#) object. Use [PCDGetDoc.AddSearchCriteria](#) to specify the criteria that identify which document and version you want. Normally, this would mean specifying criteria like "%DOCUMENT\_NUMBER" "79", "%VERSION\_ID" "3". The provided criteria must resolve to exactly one version of one document.

### Syntax

`PCDGetDoc. methodOrProperty`

### Returns

PCDGetDoc returns data stored in the COMPONENTS table, including the document number, version ID, size of the file, and the name of the file.

### Usage

The following steps show the general usage sequence for the PCDGetDoc object:

- 1** Specify the library where the document is stored.
- 2** Provide a DST.
- 3** Set search criteria that will precisely identify the item you wish to retrieve.
- 4** Call [PCDGetDoc.Execute](#) method to retrieve the document.
- 5** Iterate through them using [PCDGetDoc.NextRow](#) or call [PCDGetDoc.SetRow](#) to get to a specific item.
- 6** Call [PCDGetDoc.GetPropertyValue](#) to retrieve the file content by referencing the "%CONTENT" token. This will return a pointer to a Dispatch interface (a [PCDGetStream](#) object) that you can then use to read the physical file.

## **PCDGetDoc**

- 7** After file retrieval is complete, release memory associated with your PCDGetDoc object.

# PCDGetDoc

## Example

The following example shows you can use PCDGetDoc to retrieve the name of a file that contains a document in your DM repository.

```
.  
. .  
Dim objGetDoc As New PCDGetDoc  
objGetDoc.SetDST strDST  
objGetDoc.AddSearchCriteria _  
    "%TARGET_LIBRARY", strLib  
objGetDoc.AddSearchCriteria _  
    "%DOCUMENT_NUMBER", strDocNum  
objGetDoc.AddSearchCriteria _  
    "%VERSION_ID", strVersionID  
  
objGetDoc.Execute  
If objGetDoc.ErrNumber <> 0 Then  
    ' Error occurred.  
End If  
  
Dim lngRowCount As Long  
Dim strFileName As String  
' Dim bdata() As Byte  
' Dim indata As Variant  
  
lngRowCount = objGetDoc.GetRowsFound  
If objGetDoc.ErrNumber <> 0 Then  
    ' Error occurred.  
End If  
If lngRowCount <> 1 Then  
  
    ' Possible Error. Only 1 file expected.  
Else  
    objGetDoc.SetRow( 1 )  
    strFileName = objGetDoc.GetPropertyValues( _  
        PATH )  
    MsgBox "The name of the document is: " & _  
        strFileName  
End If  
.
```

# PCDGetDoc

## Related Items

See the following methods:

<a href="#">AddSearchCriteria</a>	<a href="#">NextRow</a>
<a href="#">Execute</a>	<a href="#">SetDST</a>
<a href="#">GetPropertyValue</a>	<a href="#">SetRow</a>
<a href="#">GetReturnProperties</a>	<a href="#">SetSearchCriteria</a>
<a href="#">GetRowsFound</a>	<a href="#">SetSearchObject</a>
<a href="#">GetSearchCriteria</a>	

See the following properties:

<a href="#">ErrDescription</a>
<a href="#">ErrNumber</a>

# **PCDGetForm**

## **PCDGetForm**

Use this object to retrieve information contained in the FORMS table in the SQL database. It is presently used specifically for the JavaForms interface.

### **Syntax**

`PCDGetForm. methodOrProperty`

### **Example**

The following example shows how to create an instance of this object.

```
pLis = Server.CreateObject("PCDCIent.PCDGetForm")
```

### **Related Items**

See the following methods:

[AddSearchLib](#)  
[Execute](#)  
[GetPropertyValue](#)  
[SetDST](#)  
[SetObjectType](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

## **PCDGetLoginLibs**

## **PCDGetLoginLibs**

Use this object to get a list of available logon libraries from the DM Server. This is a list of the libraries configured in DM Server Manager, and they are obtained from a PCDOCS.INI file in the system.

### **Syntax**

PCDGetLogi nLi bs. *methodOrProperty*

### **Usage**

This object allows you to determine which libraries are available for use by a user. Users can log on to any libraries this object returns. You also use it to select the current working library.

### **Example**

The following example assembles the libraries available to the current user and puts them into a ListBox.

```
Dim obj GetLi bs As New PCDGetLogi nLi bs  
  
Dim LNumOfLi bs As Long  
Dim strLibName() As String  
Dim LCounter  
Dim lStLibList As New ListBox  
  
' Set the DST.  
obj GetLi bs. SetDST strDST  
  
' Get a list of libraries available to the user.  
obj GetLi bs. Execute  
If (obj GetLi bs. ErrNumber <> 0) Then  
    ' Error occurred. Process it as appropriate.  
End If  
  
' Get the number of libraries.  
LNumOfLi bs = obj GetLi bs. GetSi ze - 1
```

# PCDGetLoginLibs

```
ReDim strLibName(LNumOfLibs)
For LCounter = 0 To LNumOfLibs
    strLibName(LCounter) = _
        objGetLibs.GetAt(LCounter)
    LibList.AddItem strLibName
Next

Set objGetLibs = Nothing
.
.
.
```

## Related Items

[Execute](#)  
[GetAt](#)  
[GetSize](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# PCDGetStream

## PCDGetStream

Use this object to provide the user with a way to read the contents of a physical file.

### Syntax

PCDGetStream. *methodOrProperty*

### Usage

When calling this object, after each [Read](#), you should check the [ErrNumber](#) property. If ErrNumber returns zero (indicating that no error occurred), you should check the [BytesRead](#) property to see how many bytes were actually returned by the [Read](#).

*Note:* If you are using a language such as Visual Basic or Visual C++, you can use the optional second parameter for Read to get the number of bytes read, instead of checking the BytesRead property.

### Example

The following example shows how PCDGetStream can be used to determine the length of a document in your DM Repository.

```
' Assumptions for this example:  
' - bstrDocNum already contains doc number.  
' - bstrDST already contains security token.  
' - bstrLib already contains Library name.  
' - bstrVerNum already contains the document  
'   version number.  
  
' Object to get Doc information  
Dim objDOC As New PCDDocObject  
' Create our Stream object  
Dim objGetStream As New PCDGetStream  
' Vars to hold byte counts.  
Dim blnLoopCtrl As Boolean  
Set blnLoopCtrl = False  
  
Dim lngCurCount As Long, lngTotCount As Long
```

# PCDGetStream

```
    lngCurCount = 0
    lngTotCount = 0

    ' Set our library
    obj DOC. SetProperty "%TARGET_LIBRARY", bstrLib

    ' Set the DST.
    obj DOC. SetDST bstrDST

    ' Set the Form (here the Default Profile Form).
    obj DOC. SetObj ectType "DEF_PROF"

    ' Get the document.
    obj DOC. SetProperty "%OBJECT_IDENTIFIER", _
        bstrDocNum
    obj DOC. Fetch
    If obj DOC. ErrNumber <> 0 Then
        ' Error occurred during Fetch. Process it.
    End If

    ' Create/Set-up object to get the document.
    Dim obj GetDoc As New PCDGetDoc
    obj GetDoc. SetDST bstrDST
    obj GetDoc. AddSearchCri teria "%TARGET_LIBRARY", _
        bstrLib
    obj GetDoc. AddSearchCri teria "%DOCUMENT_NUMBER",
    -
        bstrDocNum
    obj GetDoc. AddSearchCri teria "%VERS ION_ID", _
        bstrVerNum
    obj GetDoc. Execute
    If obj GetDoc. ErrNumber <> 0 Then
        ' Error occurred: Process it.
    End If

    Set obj GetStream = _
        obj GetDoc. GetPropertyVal ue("%CONTENT")
    bytInArray() = obj GetStream. Read(5120)
    lngCurCount = obj GetStream. BytesRead

    While ((obj GetStream. ErrNumber <> 0) And _
        lngCurCount > 0)
```

# PCDGetStream

```
    IngTotCount = IngTotCount + IngCurCount
    bytInArray = obj GetStream. Read(5120)
    IngCurCount = obj GetStream. BytesRead
Wend

If (obj GetStream <> 0) Then
    ' Error: Unexpected end to read loop.
Else
    If (IngTotCount > 0) Then
        MsgBox "Done. File is " & _
            CStr(IngTotCount) & _
            " Bytes in Length."
    Else
        ' Error: Read Failed. Process the error.
    End If
End If

Set obj Doc = Nothing
Set obj GetDoc = Nothing
Set obj GetStream = Nothing
```

.

.

## Related Items

See the following methods:

[GetPropertyValue](#)  
[Read](#)  
[Seek](#)  
[SetComplete](#)

See the following properties:

[BytesRead](#)  
[ErrDescription](#)  
[ErrNumber](#)

# **PCDLogin**

## **PCDLogin**

Use this object to create or append validated network aliases to a document security token (DST)

### **Syntax**

`PCDLogi n. methodOrProperty`

### **Example**

The section titled [Providing Library Access](#) in Chapter 1 illustrates how you can use the PCDLogin object in your documents.

### **Related Items**

See the following methods:

<a href="#">AddLogin</a>	<a href="#">GetFailedLoginList</a>
<a href="#">Execute</a>	<a href="#">GetLoginLibrary</a>
<a href="#">GetAliasList</a>	<a href="#">GetPrimaryGroup</a>
<a href="#">GetDOCSUserName</a>	<a href="#">SetDST</a>
<a href="#">GetDST</a>	

See the following properties:

<a href="#">ErrDescription</a>
<a href="#">ErrNumber</a>

# PCDLookup

## PCDLookup

PCDLookup allows you to execute a lookup of data stored in validated SQL columns, such as AUTHOR or DOCUMENTTYPE. You can use PCDLookup to do this in your custom application by specifying:

- the data in the fields on the form, and
- the lookup ID.

PCDLookup returns the same data that would be displayed in the list box of the DM lookup, plus any other columns that would be needed to update related fields on the base form. For example, if you use the Matter lookup definition, the column for the Client\_ID will also be returned.

### Syntax

`PCDLookup. methodOrProperty`

### Usage

This object works differently than the [PCDSearch](#) object in that you don't specify return properties. The server determines what they should be by looking at the lookup definition and the base form.

All the columns in the list box on the lookup will be included as return properties, plus the contents of any "related data" columns that need to be updated on the form when the lookup's target field changes.

Also, unlike [PCDSearch](#), you have to specify a target property. This is the field you are trying to fill in using the lookup (for example, Author or Matter).

After you [Execute](#) the lookup, you get back data and metadata. The metadata tells you what columns are in the data. The metadata columns include the following return properties: %PropertyName, %Title, %Visible, and %Data.

# PCDLookup

## Example

The following example demonstrates how you can use PCDLookup to create and process a Lookup search. It includes most of the methods that PCDLookup supports.

```
Sub Lookup( )
    ' Create our object
    Dim obj PCDLookup As New PCDLookup
    ' Create a property list
    Dim obj PCDPropList As New PCDPropertyList
    ' Set up our property list so it can be used later.
    obj PCDPropList.AddProperty "AUTHOR_ID", "J_SMI TH"

    ' Set up the parameters for the lookup.
    ' Set the DST.
    obj PCDLookup.SetDST strDST
    ' Set the Library.
    obj PCDLookup.AddSearchLib strLib
    ' Set the search object. This form must contain
    ' the lookup (such as client or matter).
    obj PCDLookup.SetSearchObject( "DEF_QBE1" )
    ' Set the Lookup name.
    obj PCDLookup.SetLookupId "PEOPLE"
    ' Set the target property to look up.
    obj PCDLookup.SetTargetProperty "AUTHOR_ID"
    ' Set the search criteria.
    obj PCDLookup.SetSearchCriteria obj PCDPropList
    ' Set the filter criteria.
    obj PCDLookup.AddUserFilterCriteria "AUTHOR_ID", "J**"

    ' Determine fields to search.
    Dim strAns As String, intAns As Integer
    Dim strPrompt As String, strTitle As String
    strTitle = "Author or Author/Typist Search " _
              & "Selection"
    strPrompt = "Current search is only in " _
               & "Author field." & vbCr & "Do you " _
               & "also wish to search for the person" & _
```

# PCDLookup

```
vbCr & "you selected in the Typist field?"  
strAns = MsgBox(strPrompt, vbYesNo, strTitle)  
intAns = CInt(strAns)  
If intAns = 6 Then  
    ' User answered "Yes." Broaden search.  
    obj Lookup.AddSearchCriteria "TYPIST_ID", J_SMITH  
    ' Also, delete filter on author name so it  
    ' does not exclude J_SMITH as typist.  
    obj Lookup.ClearUserFilterCriteria  
Else If intAns = 7  
    ' User answered "No." Search is OK as is.  
    MsgBox "No change to search criteria."  
End If  
  
' Set the sort order for results.  
strTitle = "SORT ORDER"  
strPrompt = "Select the number of "  
    & "the Sort Order: "  
    & vbCr & " 1 - Author, ascending sort "  
    & vbCr & " 2 - Author, descending sort "  
    & vbCr & " Other - Unsorted results "  
strAns = InputBox(strPrompt, strTitle)  
If IsNumeric(strAns) Then  
    intAns = CInt(strAns)  
Else  
    intAns = 9 ' Can be any integer.  
End If  
  
Select Case intAns  
Case 1  
    ' Sort by author, ascending order.  
    ' The Boolean value that follows AUTHOR_ID can be  
    ' anything except zero (or an expression that  
    ' evaluates to zero).  
    obj PCDLookup.AddOrderByProperty "AUTHOR_ID", 1  
Case 2  
    ' Sort by author, descending order.  
    obj PCDLookup.AddOrderByProperty "AUTHOR_ID", 0  
Case Else
```

# PCDLookup

```
' Unsorted. This assures unsorted results, but
' it may not be required unless there were
' previous searches.
obj PCDLookup.ClearOrderByProperties
End Select

' Set the maximum number of records search returns.
obj PCDLookup.SetMaxRows 500
' Set the number of records to be returned at
' one time to user's local cache.
obj PCDLookup.SetChunkFactor 10

' Execute the lookup
obj PCDLookup.Execute
If obj PCDLookup.ErrNumber <> 0 Then
    ' Error: process it.
End If

' Get the information from the result set
Dim intColCount As Integer
Dim lngRowsFound As Long, lngMetaFound As Long

' Get the number of data rows found by the lookup.
lngRowsFound = obj PCDLookup.GetRowsFound
MsgBox "The search returned " _
    & CStr(lngRowsFound) & " rows of data."

' Get the number of metadata rows lookup found.
lngMetaFound = obj PCDLookup.GetMetaRowsFound
MsgBox "The search returned " _
    & CStr(lngMetaFound) & " rows of metadata."

' Get the number of columns in the result data set.
intColCount = obj Lookup.ColumnCount
MsgBox "The search result data set contains " _
    & CStr(intColCount) & " columns of data."

' Clear the result set list box (if needed).
```

# PCDLookup

```
IstResultSet.Clear
' Set pointer position to row 0 in the result set.
' NextRow will then increment it to the first data
' row.
obj PCDLookup. SetRow( 0 )
Do While obj PCDLookup. NextRow
    If obj PCDLookup. ErrNumber <> 0 Then
        ' Error reading data row. Process it.
    End If
    ' Set pointer position to row 0 in the metadata
    ' result set. NextMetaRow will then increment
    ' it to the first data row.
    obj PCDLookup. SetMetaRow (0)
    Do While obj PCDLookup. NextMetaRow
        If obj PCDLookup. ErrNumber <> 0 Then
            ' Error reading metadata row. Process it.
        End If

        ' Retrieve short name of the metadata property.
        IstResultSet.AddItem _
            obj PCDLookup. GetMetaPropertyValue( _
                "PROPNAME")
        ' Shows whether column is Visible: 0=No
        IstResultSet.AddItem _
            obj PCDLookup. GetMetaPropertyValue( _
                "VISIBLE")
        ' Long Name of the metadata property.
        IstResultSet.AddItem _
            obj PCDLookup. GetMetaPropertyValue("TITLE")
        ' The lookup data associated with
        ' this metadata property.
        IstResultSet.AddItem _
            obj PCDLookup. GetMetaPropertyValue("DATA")
    IstResultSet.Show
    MsgBox "ListBox displays metadata " _
        "for current row."
    IstResultSet.Hide
    IstResultSet.Clear
```

# PCDLookup

```
    Loop  
    Loop  
  
        ' Clean up...  
        obj PCDLookup. ReleaseResults  
        Set obj PCDLookup = Nothing  
        Set obj PCDPropList = Nothing  
  
    End Sub
```

## Related Items

See the following methods:

AddOrderByProperty	NextMetaRow
AddSearchCriteria	NextRow
AddSearchLib	ReleaseResults
AddUserFilterCriteria	SetChunkFactor
ClearOrderByProperties	SetDST
ClearUserFilterCriteria	SetLookupId
ColumnCount	SetMaxRows
Execute	SetMetaRow
GetMetaPropertyValue	SetRow
GetMetaRowsFound	SetSearchCriteria
GetPropertyValuesByIndex	SetSearchObject
GetRowsFound	SetTargetProperty
GetSearchCriteria	

See the following properties:

ErrDescription
ErrNumber

# **PCDNetAliasList**

## **PCDNetAliasList**

The PCDNetAliasList object stores a list of network aliases. A network alias consists of the following:

- a Uni tType
- a Uni tName
- a UserName and Password

The Uni tType is a DM library, or a NetWare 5.x, NetWare 6.x, or Microsoft Network. Depending on the Uni tName, the Uni tType is either a DM library name, the NetWare server name, the NetWare NDS tree name, or the Windows network domain name, respectively. Note that this object accepts and stores passwords, but it does not allow them to be retrieved.

### **Syntax**

PCDNetAl i asLi st. *methodOrProperty*

### **Example**

The section titled [Providing Library Access](#) in Chapter 1 illustrates how you can use the PCDNetAliasList object in your custom applications.

### **Related Items**

See the following methods:

[GetSize](#)

[UnitName](#)

[UnitType](#)

[UserName](#)

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# PCDNetworkInfo

## PCDNetworkInfo

The PCDNetworkInfo object supports the integration of DM with your network-based security. The methods PCDNetworkInfo supports allow you to build tight network operating system integration into your DM document management system. These methods also allow you to browse network information about domains, groups, and users.

### Syntax

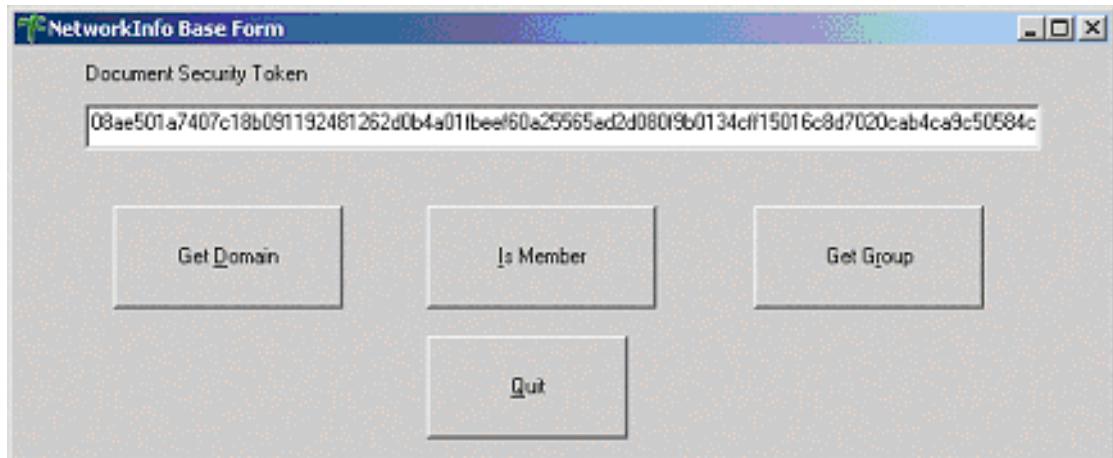
PCDNetworkInfo. MethodOrProperty

### Usage

Use methods supported by PCDNetworkInfo to make inquiries of current network elements within your network environment. You can retrieve information about Domains, Groups, and Users. Most of these methods load a result set with data. You retrieve data from these result sets by using a pair of methods in conjunction with one another: [NextRow](#) and [GetValue](#).

### Example

The following example illustrates the use of PCDNetworkInfo and the methods it supports.



# PCDNetworkInfo

```
Public sDST As String

Private Sub cbDomain_Click()
    DomainForm. oNWI nfo. SetDST (sDST)
    DomainForm. Show
End Sub

Private Sub cbGroupInfo_Click()
    GroupForm. sDST = sDST
    GroupForm. oGroupInfo. SetDST (sDST)
    GroupForm. Show
End Sub

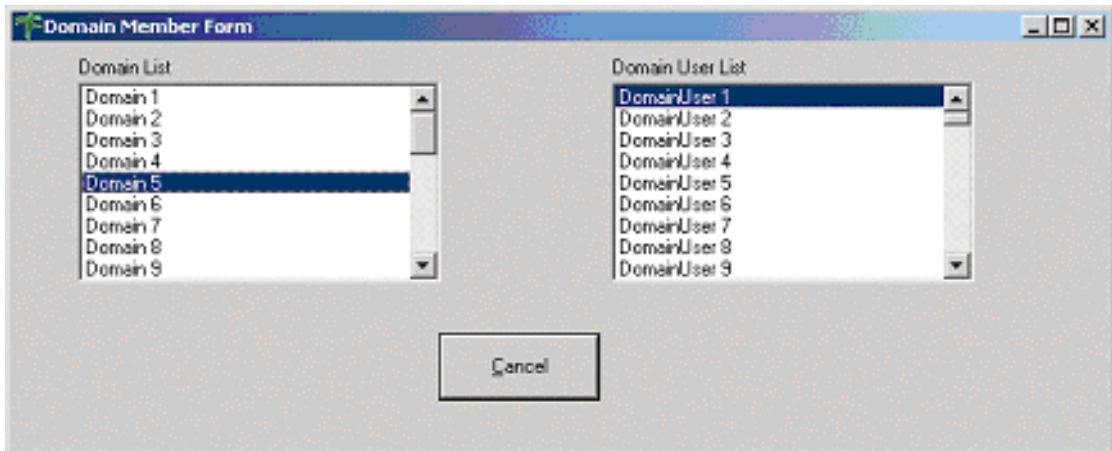
Private Sub cbQuiet_Click()
    Unload NetInfoBaseForm
End Sub

Private Sub Form_Load()
    ' Local Variabile declarations.
    Dim oLogin As New PCDLogin
    Dim nResult As Long
    Dim sTempBuf As String

    ' Login process.
    nResult = oLogin. AddLogin(0, "MyLibrary", "", "")
    nResult = oLogin. AddLogin(0, "MyDomain", _
        "MyUserID", "MyPassword")
    nResult = oLogin. Execute()
    sDST = oLogin. GetDST()

    ' Get And display the DST.
    txtDST. Text = sDST
End Sub
```

# PCDNetworkInfo



```
Public oNWIInfo As New PCDNetworkInfo

Private Sub cbCancel_Click()
    Unload DomainForm
End Sub

Private Sub Form_Load()

    Dim nResult As Long
    Dim nNumRows As Long

    ' Load the Domain List.
    ' %NI_NT indicates this is an NT based OS.
    ' %UNDEFINED returns all domains from the root.
    nResult = oNWIInfo.GetDomainList("%NI_NT", _
        "%UNDEFINED")

    nNumRows = 0

    ' If the Domain List has been retrieved,
    ' get the number of domains in the list.
    If nResult = 0 Then
```

# PCDNetworkInfo

```
nNumRows = oNWIInfo.GetRowCount()
End If

If nNumRows = 0 Then
    MsgBox "You do not have access to Domain " _
        & "information at this time."
Else
    ' Fill the listbox.
    For i = 1 To nNumRows
        nResult = oNWIInfo.NextRow()
        lstDomains.AddItem (oNWIInfo.GetValue())
    Next i

    ' Initialize this list to its first element.
    lstDomains.ListIndex = 0

    ' Fill up the UserList with Users in this list.
    nResult = oNWIInfo.GetUserList("%NI_NT", _
        lstDomains.Text)

nNumRows = 0

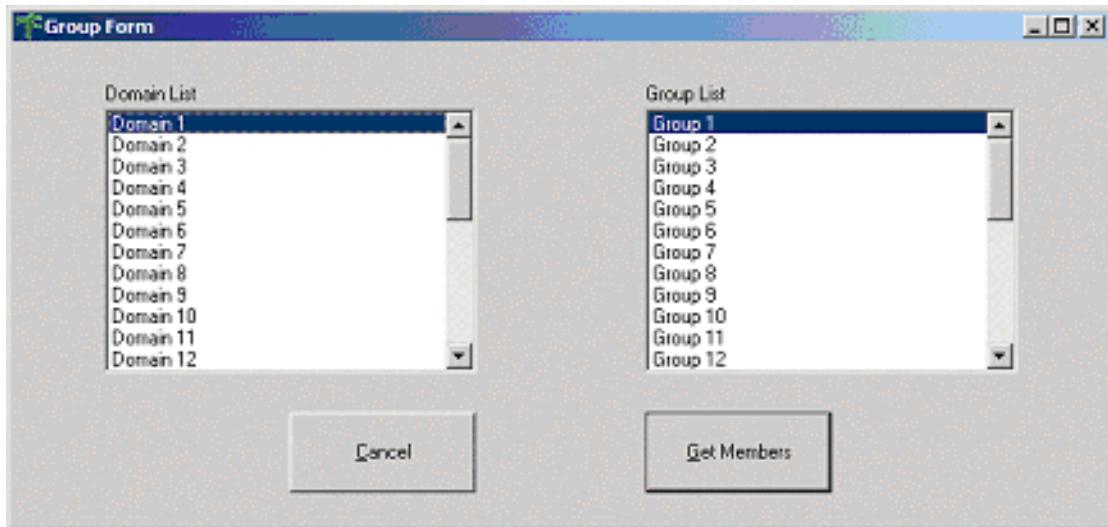
' Get number of rows that are returned.
If nResult = 0 Then
    nNumRows = oNWIInfo.GetRowCount()
End If
'Display the list of users in the
' lstDomainUsers listbox.
If nNumRows = 0 Then
    MsgBox "User information for this domain " _
        & "is not available to you."
Else
    For i = 1 To nNumRows
        nResult = oNWIInfo.NextRow()
        lstDomainUserList.AddItem ( _
            oNWIInfo.GetValue())
    Next i
```

# PCDNetworkInfo

```
' Pre-select the first item in the list.  
lstDomainUserList.ListIndex = 0  
  
End If  
  
End If  
  
End Sub  
  
Private Sub lstDomains_Click()  
  
Dim nResult As Long  
Dim nNumRows As Long  
  
' Clear listbox for results of this call.  
lstDomainUserList.Clear  
  
' Fill the UserList with Users within this list.  
nResult = oNWIInfo.GetUserList("%NI_NT", _  
    lstDomains.Text)  
  
nNumRows = 0  
  
' Get the number of rows that are returned.  
If nResult = 0 Then  
    nNumRows = oNWIInfo.GetRowCount()  
End If  
  
' Display users in the lstDomainUsers listbox  
If nNumRows = 0 Then  
    MsgBox "User information for this domain " _  
        & "is not available to you."  
Else  
    For i = 1 To nNumRows  
        nResult = oNWIInfo.NextRow()  
        lstDomainUserList.AddItem( _  
            oNWIInfo.GetValue())  
    Next i
```

# PCDNetworkInfo

```
' Pre-select the first item in the list.  
lstDomainUserList.ListIndex = 0  
  
End If  
  
End Sub
```



```
Public sDomainName As String  
Public sGroupName As String  
Public sDST As String  
Public oGroupInfo As New PCDNetworkInfo  
  
Private Sub cbCancel_Click()  
    Unload GroupForm  
End Sub  
  
Private Sub cbMember_Click()  
  
    Dim nResult As Long  
  
    nResult = lsMemberForm.lsMember.SetDST(sDST)
```

# PCDNetworkInfo

```
IsMemberForm.sDomainName = "MyDomain"
IsMemberForm.sGroupName = "MyGroup"

IsMemberForm.TxtUserID = "Jimmy Jones"
IsMemberForm.Show

End Sub

Private Sub cbGetMembers_Click()

Dim nResult As Long

' Before loading the next form use Load
' Load the next form's Group and Domain
' data members.
GroupMembersForm.sGroupName = sGroupName
GroupMembersForm.sDomainName = sDomainName

' Set the next form's object DST.
nResult = GroupMembersForm.oMembers.SetDST(sDST)

GroupMembersForm.Show

End Sub

Private Sub Form_Load()

Dim nResult As Long
Dim sName As String
Dim nNumRows As Long

' Load up the Domain List.
nResult = oGroupInfo.GetDomainList(_
    "%NI_NT", "%UNDEFINED")

nNumRows = 0

' If the Domain list has been retrieved,
' get the number of domains in the list.
```

# PCDNetworkInfo

```
If nResult = 0 Then
    nNumRows = oGroupInfo. GetRowCount()
End If

If nNumRows = 0 Then
    MsgBox "You do not have access to " _
        & "Domain information at this time."
Else
    ' Fill ListBox with Domain information.
    For i = 1 To nNumRows
        nResult = oGroupInfo. NextRow()
        lstDomains.AddItem (oGroupInfo. GetValue())
    Next i

    ' Initialize Domain list to first element.
    lstDomains.ListIndex = 0

    ' This Form data member holds currently
    ' selected Domain information.
    sDomainName = lstDomains.Text

    nResult = oGroupInfo. GetGroupList( _
        "%NI_NT", sDomainName)

    If nResult = 0 Then
        nNumRows = oGroupInfo. GetRowCount()
    End If

    ' If no rows are returned then place that
    ' information in the ListBox. Otherwise, place
    ' the list of all groups in the ListBox.
    If nNumRows = 0 Then
        lstGroups.AddItem "NO GROUPS FOR THIS DOMAIN"
    Else
        For i = 1 To nNumRows
            nResult = oGroupInfo. NextRow()
            lstDomains.AddItem (oGroupInfo. GetValue())
        Next i
```

# PCDNetworkInfo

```
End If
End If

cbGetMembers.Enabled = False

End Sub

Private Sub lstDomains_Click()
    sDomainName = lstDomains.Text

    lstGroups.Clear

    ' Fill the UserList with Users in this list.
    nResult = oGroupInfo.GetGroupList(_
        "%NI_NT", sDomainName)

    nNumRows = 0

    ' Get the rowcount.
    If nResult = 0 Then
        nNumRows = oGroupInfo.GetRowCount()
    End If

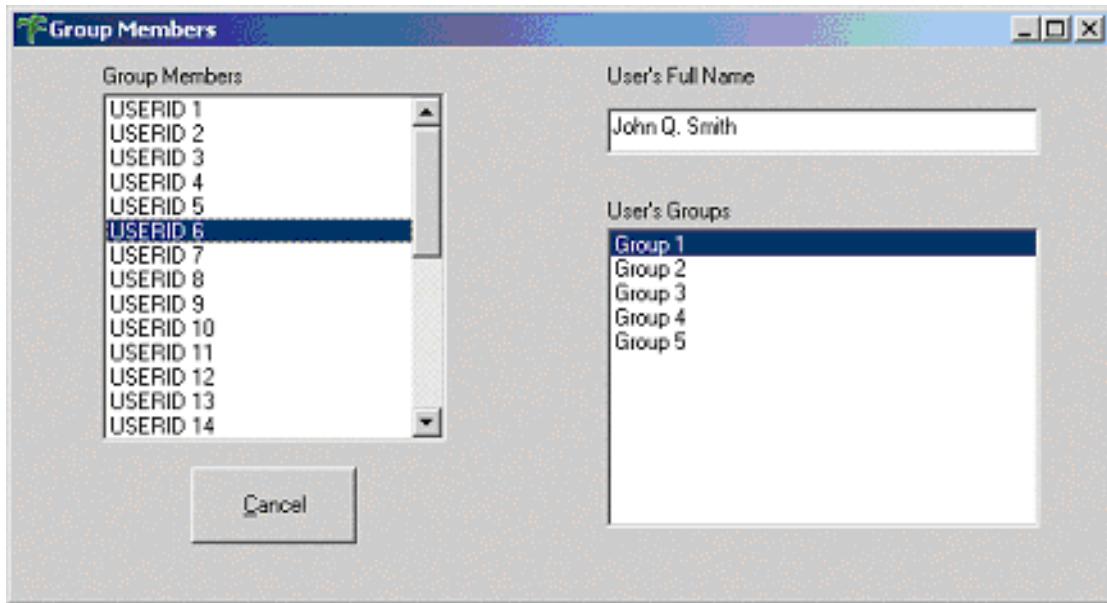
    ' Display user list in the lstGroups ListBox.
    If nNumRows = 0 Then
        MsgBox "User information for this domain " __
            "is not available to you."
    Else
        For i = 1 To nNumRows
            nResult = oGroupInfo.NextRow()
            lstGroups.AddItem (oGroupInfo.GetValue())
        Next i

        ' Pre-select the first item in the list.
        lstGroups.ListIndex = 0
        sGroupName = lstGroups.Text
        cbGetMembers.Enabled = True
    End If
```

# PCDNetworkInfo

```
End Sub

Private Sub IstGroups_Click()
    sGroupName = IstGroups.Text
    cbGetMembers.Enabled = True
End Sub
```



```
Public oMembers As New PCDNetworkInfo
Public sDomainName As String
Public nFirstTime As Long
Public sGroupName As String

Private Sub cbCancel_Click()
    Unload GroupMembersForm
End Sub

Private Sub Form_Load()

    Dim nResult As Long
```

# PCDNetworkInfo

```
Dim sMember As String
Dim nNumRows As Long

' Retrieve the GroupMembers from the network
nResult = oMembers.GetGroupMembers( _
    "%NI_NT", sDomainName, sGroupName)

If nResult = 0 Then

    ' Get the Group members.
    nNumRows = oMembers.GetRowCount()

    If nNumRows = 0 Then
        MsgBox "No access to Members for this group."
    Else

        ' Place all of the members of this group
        ' in the ListBox.
        For i = 1 To nNumRows
            nResult = oMembers.NextRow()
            lstMembers.AddItem (oMembers.GetValue())
        Next i

        ' Select the first member in the list.
        lstMembers.ListIndex = 0
    End If

Else
    ' No members in this group.
    lstMembers.AddItem sDomainName + "No Members"
End If

nFirstTime = 1

End Sub

Private Sub lstMembers_Click()

Dim nResult As Long
```

# PCDNetworkInfo

```
Dim nNumRows As Long

' If program is now checking the top of Member
' list from the opening of the form...
If nFirstTime > 0 Then

    ' ... Clear the Users Groups ListBox.
    lstUsersGroups.Clear

    ' Retrieve the selected user's full name.
    nResult = oMembers.GetUserFullName( _
        "%NI_NT", sDomainName, lstMembers.Text)
    nResult = oMembers.NextRow()

    txtFullName.Text = oMembers.GetValue()

    ' Retrieve all groups that include the
    ' selected user.
    nResult = oMembers.GetUserGroups( _
        "%NI_NT", sDomainName, lstMembers.Text)
    If nResult = 0 Then
        nNumRows = oMembers.GetRowCount()

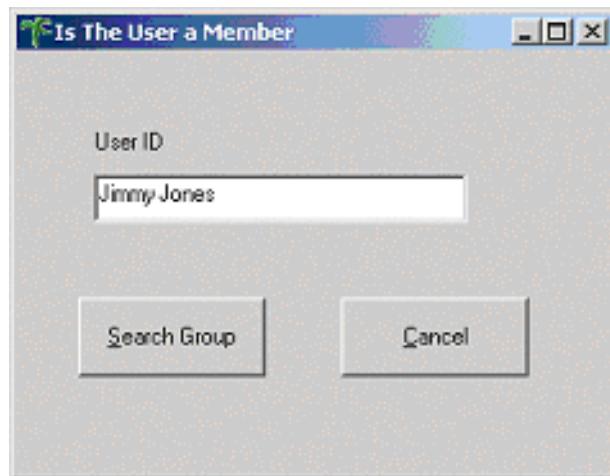
        If nNumRows > 0 Then

            ' Place all Groups that include this user
            ' into the UsersGroups ListBox.
            For i = 1 To nNumRows
                nResult = oMembers.NextRow()
                lstUsersGroups.AddItem( _
                    oMembers.GetValue())
            Next i
        Else
            lstUsersGroups.AddItem lstMembers.Text & _
                " is not a member of any groups."
        End If
    Else
        lstUsersGroups.AddItem "Unable to " _
```

# PCDNetworkInfo

```
    & "retrieve the Users Groups."
End If
Else
    ' This is the first time through. Do not
    ' retrieve any Group or name information
    ' for this user ID.
    nFirstTime = nFirstTime + 1
End If

End Sub
```



```
Public sGroupName As String
Public olMember As New PCDNetworkInfo
Public sDomainName As String

Private Sub cbCancel_Click()
    Unload IsMemberForm
End Sub

Private Sub cbSearchGroup_Click()

    Dim nResult As Long
    Dim dUserID As String
```

# PCDNetworkInfo

```
sUserID = TxtUserID.Text

nResult = oISMember.IsMemberOf( "%NI_NT", _
                               sDomainName, sUserID, sGroupName)

If oISMember.NextRow() Then
    sResultValue = oISMember.GetValue()
    MsgBox "Positive response to IsMember - " _
           & sResultValue & "."
Else
    MsgBox "NextRow returned false."
End If

End Sub
```

## Related Items

See the following methods:

<a href="#">GetDomainList</a>	<a href="#">GetUserList</a>
<a href="#">GetGroupList</a>	<a href="#">GetValue</a>
<a href="#">GetGroupMembers</a>	<a href="#">IsMemberOf</a>
<a href="#">GetRowCount</a>	<a href="#">NextRow</a>
<a href="#">GetUserFullName</a>	<a href="#">SetDST</a>
<a href="#">GetUserGroups</a>	

See the following properties:

<a href="#">ErrDescription</a>
<a href="#">ErrNumber</a>

# **PCDPropertyList**

## **PCDPropertyList**

PCDPropertyList is a Messenger object that allows you to store and manipulate a collection of property name/value pairs.

PCDPropertyList uses a zero-based index. Loading different name/value pairs into this object provides a mechanism for your custom applications to send descriptive information from the client interface to the server interface.

One example of the use of these properties occurs within the [PCDNetworkInfo](#) object. Many of the methods in PCDNetworkInfo set properties in the PCDPropertyList data member and then call a protected Execute method, which calls back to the server. The property list is broken out in the call to the server, into an integer value giving the size of the array, an array of string variables (BSTRs) representing the property names, and an array of Variants containing the values of these properties. PCDPropertyList methods contain elements that specify which methods the server should call, as well as the parameters for those methods.

Since the PCDPropertyList object is primarily used as a utility by the other objects that comprise the DM API, this object should not be implemented directly. Instead, it should be used as a tool that allows you to modify other objects that are supported within the DM API.

### **Syntax**

`PCDPropertyList. methodOrProperty`

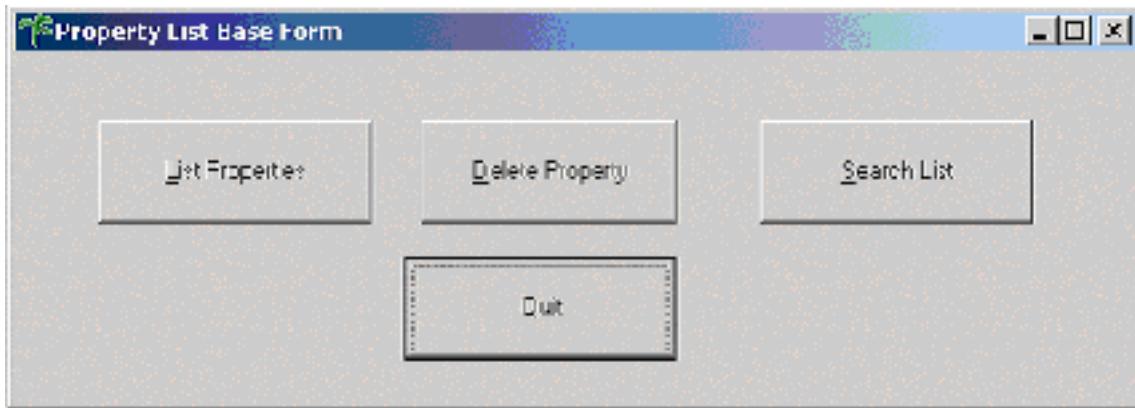
### **Usage**

Since the PCDPropertyList object is used to pass information between the client and server, direct calls to these methods provide you the ability to share unique information required by your custom application. However, any unique properties you create as part of your application will only work properly if they are correctly received and interpreted in both the client interface and the server interface. If you add or modify a client-side property, you must make appropriate modifications in your server-side application, or changes that you make will have no effect.

# PCDPropertyList

## Example

The following example shows how you can incorporate the functionality of the GetPropertyList object into an application.



```
Public oGlobal PropertyList As New PCDPropertyList

Private Sub cbDelete_Click()

    Dim nResult As Long
    Dim nTotal Elements As Integer

    nTotal Elements = oGlobal PropertyList.GetSize

    MsgBox "The Total number of elements in the " _
        & "PCDPropertyList - " + Str(nTotal Elements)

    For i = 1 To nTotal Elements
        nResult=oGlobal PropertyList.NextProperty()
        nResult=DeleteForm.oDeleteProp.AddProperty(
            oGlobal PropertyList.GetCurrentPropertyName(),
            oGlobal PropertyList.GetCurrentPropertyValue())
    Next i

    DeleteForm.Show
End Sub
```

# PCDPropertyList

```
End Sub

Private Sub cbListProperties_Click()
    Dim nResult As Long
    Dim nTotalElements As Integer

    nTotalElements = oGlobalPropertyList.GetSize

    MsgBox "The Total number of elements in the " _
        & "PCDPropertyList - " + Str(nTotalElements)

    For i = 1 To nTotalElements
        nResult = oGlobalPropertyList.NextProperty()
        nResult = ListForm.oListFormProps.AddProperty( _
            oGlobalPropertyList.GetCurrentPropertyName(), _
            oGlobalPropertyList.GetCurrentPropertyValue())
    Next i

    ListForm.Show
End Sub

Private Sub cbQuery_Click()
    Unload BaseForm
End Sub

Private Sub cbSearch_Click()
    Dim nTotalElements As Integer
    Dim nResult As Long

    nTotalElements = oGlobalPropertyList.GetSize

    MsgBox "The Total number of elements in the " _
        & "PCDPropertyList - " + Str(nTotalElements)

    nResult = oGlobalPropertyList.BeginIter()

```

# PCDPropertyList

```
For i = 1 To nTotalElements
    nResult = _
        ValueForm.oval ueFormProps. AddProperty( _
            oGlobal PropertyList. GetCurrentPropertyName(),_
            oGlobal PropertyList. GetCurrentPropertyValue())
    nResult = oGlobal PropertyList. NextProperty()
Next i

ValueForm. Show

End Sub

' During the form load, a series of performance
' metadata elements are being loaded into the
' Property list.

Private Sub Form_Load()

    Dim nResult As Long
    Dim vPropValue As Variant

    vPropValue = 85

    ' Load CPU metadata.
    nResult = oGlobal PropertyList. AddProperty( _
        "CPU", vPropValue)

    ' If successful, then load the Memory metadata.
    If nResult = 0 Then
        vPropValue = 512100
        nResult = oGlobal PropertyList. AddProperty( _
            "Memory", vPropValue)

        ' If successful, then load the Table Locks
        ' metadata
        If nResult = 0 Then
            vPropValue = 125
            nResult = _
```

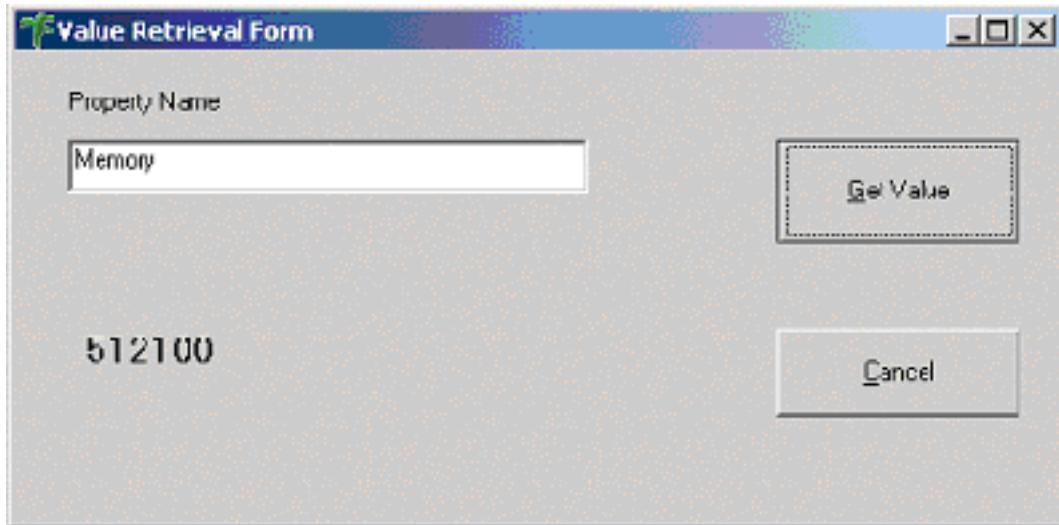
# PCDPropertyList

```
oGlobal PropertyList. AddProperty( _  
    "Table Locks", vPropValue)  
  
' If successful, then load the Pages Allocated  
' metadata.  
If nResult = 0 Then  
    vPropValue = 30  
    nResult =_  
        oGlobal PropertyList. AddProperty( _  
            "Pages Allocated", vPropValue)  
' If successful then load the Pages Input  
' metadata.  
  
If nResult = 0 Then  
    vPropValue = 100  
    nResult =_  
        oGlobal PropertyList. AddProperty( _  
            "Pages Input", vPropValue)  
  
If nResult <> 0 Then  
    MsgBox "Pages Input Property " _  
        & "addition failed."  
  
End If  
  
Else  
    MsgBox "Pages Allocated Property " _  
        & "addition failed"  
End If  
  
Else  
    MsgBox "Table Locks property " _  
        & "addition failed."  
End If  
  
Else  
    MsgBox "Memory Property addition failed"  
End If
```

# PCDPropertyList

```
Else
    MsgBox "CPU Property Addition has failed"
End If

End Sub
```



```
Public oValueFormProps As New PCDPropertyList

Private Sub cbCancel_Click()
    Unload ValueForm
End Sub

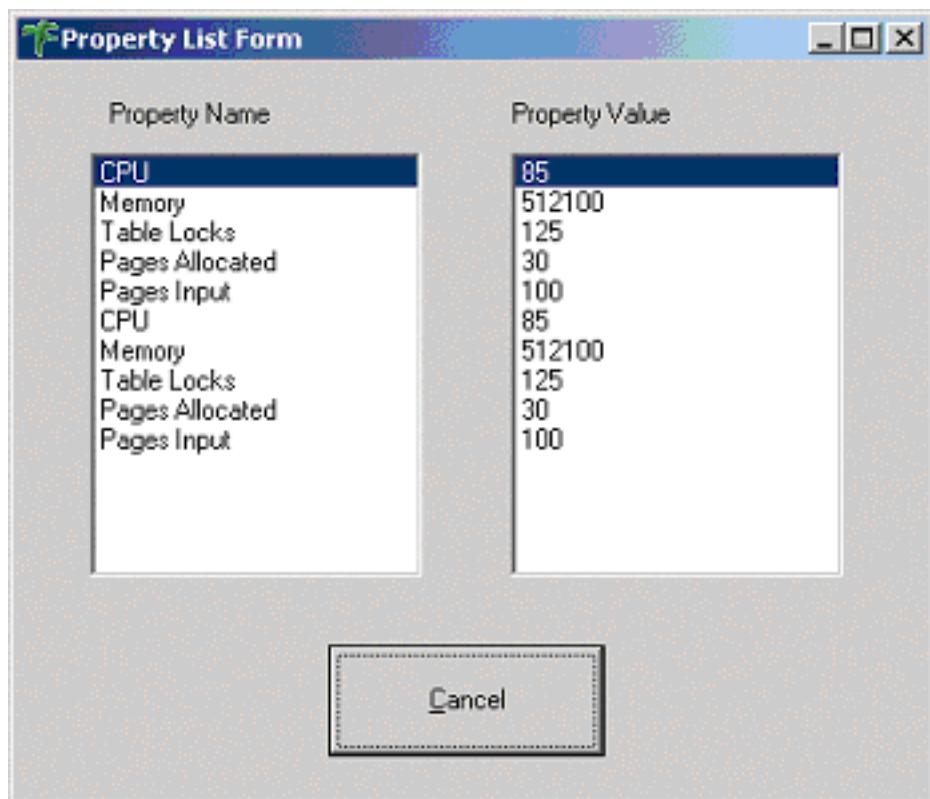
Private Sub cbFindValue_Click()

    Dim nResult As Long
    Dim sSearchString As String
    Dim vPropValue As Variant
    sSearchString = txtPropertyName.Text

    ' Use the value in the text box to retrieve
    ' the value of the text box.
    vPropValue = _
```

## PCDPropertyList

```
oVal ueFormProps. GetPropertyValue( _  
    sSearchString)  
  
    ' Report the value of the property.  
    Ibl Value. Caption = vPropValue  
    Ibl Value. FontSize = 12  
    Ibl Value. FontBold = True  
  
End Sub  
  
Private Sub Form_Load()  
    txtPropertyName. Text = Ibl PPropertyName. Caption  
End Sub
```



# PCDPropertyList

```
' Note that this example retrieves each value 2  
' times.
```

```
Public oListFormProps As New PCDPropertyList
```

```
Private Sub cbCancel_Click()  
    Unload ListForm  
End Sub
```

```
Private Sub Form_Load()
```

```
Dim nTotal Elements As Integer  
Dim nResult As Long
```

```
nTotal Elements = oListFormProps.GetSize
```

```
' Prime the pump by setting the index element to 0  
' so program can iterate through the PCDProperties  
' array.
```

```
nResult = oListFormProps.BeginIter
```

```
nTotal Elements = nTotal Elements * 2
```

```
' In this way we are able to go through the  
' entire list of elements.
```

```
For i = 1 To nTotal Elements
```

```
    lstPropertyName.AddItem _
```

```
        oListFormProps.GetCurrentPropertyName
```

```
    lstPropertyValue.AddItem _
```

```
        oListFormProps.GetCurrentPropertyValue
```

```
    nResult = oListFormProps.NextProperty
```

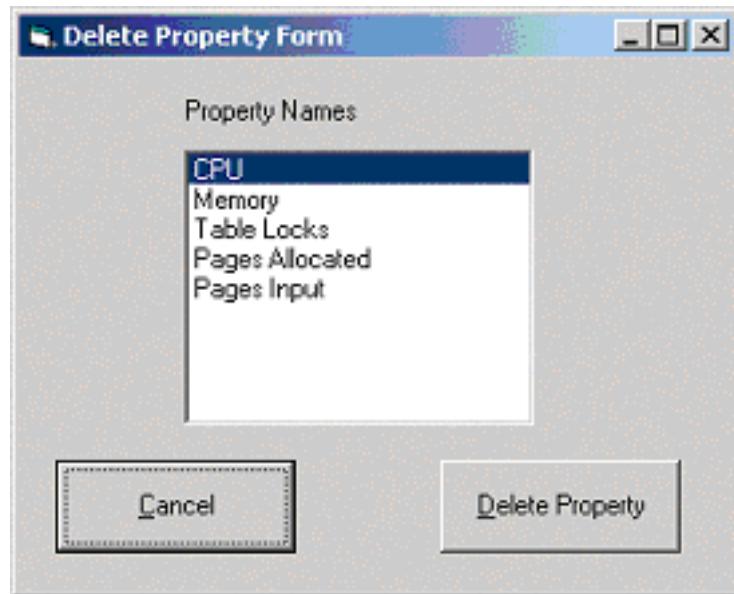
```
Next i
```

```
lstPropertyName.ListIndex = 0
```

```
lstPropertyValue.ListIndex = 0
```

```
End Sub
```

# PCDPropertyList



```
Public nActiveProperty As Integer
Public oDeleteProp As New PCDPropertyList

Private Sub cbCancel_Click()
    Unload DeleteForm
End Sub

Private Sub cbDelete_Click()

    Dim nResult As Long
    Dim nTotal Rows As Long

    ' Clear the listbox.
    lstProperties.Clear

    ' Delete the property that was selected.
    nResult = oDeleteProp.DeleteProperty -
        nActiveProperty
```

# PCDPropertyList

```
' Determine number of rows in the PCDPropertyList.  
nTotal Rows = oDeleteProp.GetSize()  
  
If nTotal Rows > 0 Then  
    ' If there are rows in the Property list, get  
    ' ready to retrieve properties from the list.  
    nResult = oDeleteProp.BeginIter()  
  
    ' Place the property names in the listbox.  
    For i = 1 To nTotal Rows  
        lstProperties.AddItem( _  
            oDeleteProp.GetCurrentPropertyName())  
        nResult = oDeleteProp.NextProperty()  
    Next i  
End If  
  
lstProperties.ListIndex = 0  
nActiveProperty = lstProperties.ListIndex  
  
End Sub  
  
Private Sub Form_Load()  
  
    Dim nResult As Long  
    Dim nTotal Rows As Long  
  
    ' Determine number of rows in the PCDPropertyList.  
    nTotal Rows = oDeleteProp.GetSize()  
  
    If nTotal Rows > 0 Then  
        ' If there are rows in the Property list, get  
        ' ready to retrieve properties from the list.  
        nResult = oDeleteProp.BeginIter()  
  
        ' Place the property names in the listbox.  
        For i = 1 To nTotal Rows  
            lstProperties.AddItem( _  
                oDeleteProp.GetCurrentPropertyName())
```

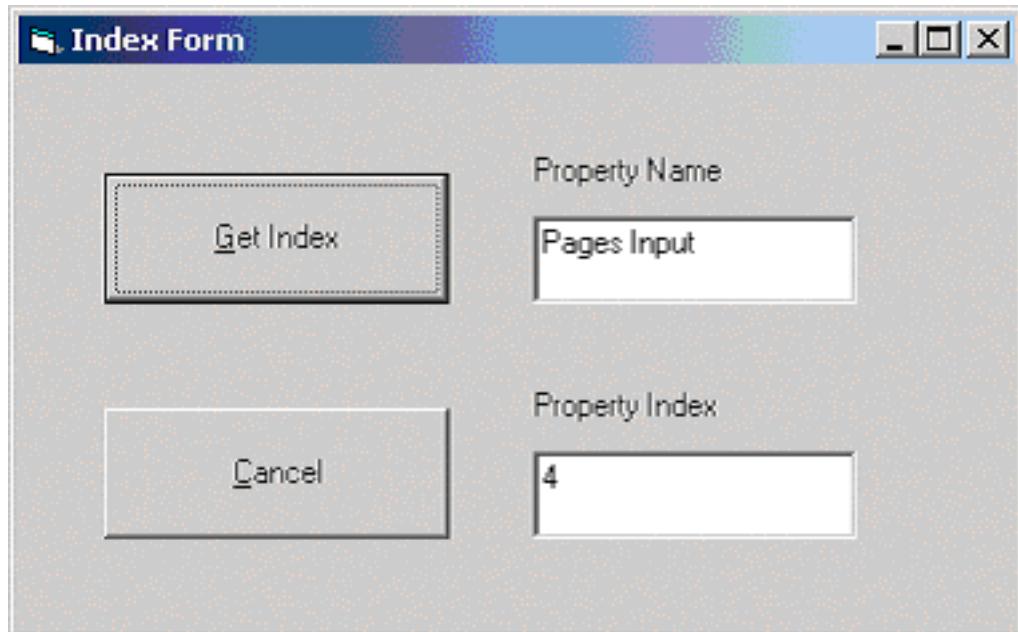
## PCDPropertyList

```
nResult = oDeleteProp.NextProperty()
Next i
End If

ListProperties.ListIndex = 0
nActiveProperty = ListProperties.ListIndex

End Sub

Private Sub ListProperties_Click()
nActiveProperty = ListProperties.ListIndex
End Sub
```



```
Private Sub cbGetIndex_Click()
' Note: This code above demonstrates the
' GetPropertyIndex call on the PCDPropertyList
' object. You enter the property name ( value ),
' and you are returned the zero-based index of
' that property.
```

# PCDPropertyList

```
Dim nResult As Long
Dim nIndex As Long
Dim nSize As Long
Dim sPropertyName As String

sPropertyName = txtPropertyName.Text

If sPropertyName = "None" Then
    MsgBox "Please enter a property before " _
        & "trying to retrieve the index."
Else
    nSize = nIndexFormProps.GetSize()
    If nSize > 0 Then
        nIndex = _
            nIndexFormProps.GetPropertyIndex( _
                sPropertyName)
        If nIndex >= 0 And nIndex < nSize Then
            txtPropertyIndex.Text = nIndex
        Else
            MsgBox "This is not a valid property " _
                & "or the property returned is out " _
                & "of range."
        End If
    Else
        MsgBox "The PropertyList is empty."
    End If
End If

End Sub
```

## Related Items

See the following methods:

- [AddProperty](#)
- [BeginIter](#)
- [DeleteProperty](#)
- [GetCurrentPropertyName](#)
- [GetCurrentPropertyValue](#)
- [GetPropertyIndex](#)

# **PCDPropertyList**

[GetPropertyValue](#)

[GetSize](#)

[NextProperty](#)

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# **PCDPropertyLists**

## **PCDPropertyLists**

This object lets you work with a collection of [PCDPropertyList](#) objects.

### **Syntax**

`PCDPropertyLists. methodOrProperty`

### **Related Items**

See the following methods:

[BeginIter](#)  
[Execute](#)  
[GetCurrentPropertyName](#)  
[GetCurrentPropertyValue](#)  
[NewEnum](#)  
[NextProperty](#)  
[NextRow](#)  
[SetChunkFactor](#)  
[SetDST](#)  
[SetObjectType](#)  
[SetOptions](#)  
[SetProperties](#)  
 [SetProperty](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# PCDPutDoc

## PCDPutDoc

PCDPutDoc allows you to obtain and manage a set of [PCDPutStream](#) object pointers that your customized client application can use to write the physical files that make up one version of a DM document to the appropriate DM repository.

### Syntax

`PCDPutDoc. methodOrProperty`

### Usage

The particular document/version that is to be written is determined by setting search criteria. Normally, this would be the document ID number, the version ID number, and possibly the sub-version identifier. The actual names of these properties is determined by how they are identified on the search form, rather than the names of the SQL database columns they reference.

*Note:* While the DM API allows you to change the name of fields on forms you create (or modify), you should only do this when absolutely necessary. You should never rename original fields on the standard forms that DM creates when you initially install it.

Also, you must set the NUM\_COMPONENTS property in the search criteria. This tells the server how many component files will make up the document version. The default is 1, but some applications (such as some CAD/CAM files) are comprised of several component files.

After calling [Execute](#), you can call [GetRowsFound](#) to return NUM\_COMPONENTS. Use [NextRow](#) or [SetRow](#) to iterate through the rows found. For each of the component files, use [GetPropertyValue](#)("CONTENT") to retrieve a [PCDPutStream](#) object pointer to use to write the file.

### Example

The following example shows how you can use the PCDPutDoc object in your custom applications.

```
Private Sub cmdCreateDoc_Click()
    Dim DocNumber As Long
```

# PCDPutDoc

```
Dim VersionID As Long
Dim TotalFileSize As Long
Dim TotalBytesWritten As Long
Dim Buffread As Long
' Set bdata ridiculously low for test purposes
' to increase the likelihood of corruption.
Dim bdata(16) As Byte

Dim objDoc As PCDDocObject
Set objDoc = New PCDDocObject

objDoc SetProperty "%TARGET_LIBRARY", Library
objDoc SetDST DST
' Set the Profile Form for this document.
objDoc SetObjectType "DEF_PROF"
objDoc SetProperty "DOCNAME", "Upload Test Doc"
objDoc SetProperty "APP_ID", "MS WORD"
objDoc SetProperty "AUTHOR_ID", "SMITH_J"
objDoc SetProperty "TYPEST_ID", "SMITH_J"
' Set the Document Type, Letter, Memo, etc.
objDoc SetProperty "TYPE_ID", "REPORT"

objDoc SetProperty "ABSTRACT", _
    "Imported via Custom Application"

objDoc.Create
If objDoc ErrNumber <> 0 Then
    Debug.Print objDoc ErrNumber, _
        objDoc ErrNumber
    Exit Sub
Else
    Debug.Print "Created Document Profile!"
End If

DocNumber = objDoc GetReturnProperty( _
    "%OBJECT_IDENTIFIER")
VersionID = objDoc GetReturnProperty( _
    "%VERSION_ID")

Debug.Print "Doc Number: ", DocNumber, _
    ", Version ID: ", VersionID

Dim objPutDoc As PCDPutDoc
Set objPutDoc = CreateObject( _
    "PCDClient.PCDPutDoc")
objPutDoc SetDST DST
```

# PCDPutDoc

```
obj PutDoc. AddSearchCriteria _  
    "%TARGET_LIBRARY", Library  
obj PutDoc. AddSearchCriteria _  
    "%DOCUMENT_NUMBER", DocNumber  
obj PutDoc. AddSearchCriteria "%VERSION_ID", _  
    VersionID  
obj PutDoc. Execute  
  
If obj PutDoc. ErrNumber <> 0 Then  
    Debug. Print obj PutDoc. ErrNumber, _  
        obj PutDoc. ErrDescription  
    Exit Sub  
End If  
  
obj PutDoc. NextRow  
  
Dim obj PutStream As Object  
Set obj PutStream = _  
    obj PutDoc. GetPropertyValue("%CONTENT")  
  
Dim FileName  
FileName = txtFileName.Text  
  
Debug. Print "File Name: ", FileName  
  
Open FileName For Binary Access Read As #1  
  
TotalFileSize = LOF(1)  
  
While (Not EOF(1))  
    TotalBytesWritten = TotalFileSize  
    If TotalBytesWritten > 0 Then  
  
        ' If you want to use Ubound, then you have  
        ' to increment the reads correctly. The  
        ' previous code read each successive group  
        ' of bytes and skipped the byte between the  
        ' reads. This occurred due to the way in  
        ' which VB reads binary data. (Read about  
        ' the Get method.)  
  
        If (TotalBytesWritten > UBound(bdata)) Then  
            TotalBytesWritten = UBound(bdata) + 1  
            ' The +1 was added because this only gets  
            ' called on successive reads, and not the  
            ' first read.
```

# PCDPutDoc

```
Else
    Total FileSize = Total BytesWritten
End If

Get #1, , bdata

obj PutStream.Write bdata, Total BytesWritten

Debug.Print Total BytesWritten, _
    obj PutStream.BytesWritten, _
    obj PutStream.ErrNumber, _
    obj PutStream.ErrDescription

Total BytesWritten = Total FileSize - _
    Total BytesWritten

End If
Wend

obj PutStream.SetCompliance
Close #1

Set obj PutSteam = Nothing
Set obj Doc = Nothing

Set obj Doc = New PCDDocObj ect
obj Doc.SetDST DST
obj Doc.SetObjectType "DEF_PROF"
obj Doc SetProperty "%TARGET_LIBRARY", Library
obj Doc SetProperty "%OBJECT_IDENTIFIER", _
    DocNumber
obj Doc SetProperty "%VERSION_ID", VersionID
obj Doc SetProperty "%STATUS", "%UNLOCK"
obj Doc.Update

If obj Doc.ErrNumber <> 0 Then
    Debug.Print obj Doc.ErrNumber, _
        obj Doc.ErrDescription
End If

Set obj PutDoc = Nothing
Set obj Doc = Nothing

Debug.Print "Upload completed!"

End Sub
```

# PCDPutDoc

## Related Items

See the following methods:

<a href="#">AddSearchCriteria</a>	<a href="#">NextRow</a>
<a href="#">Execute</a>	<a href="#">SetDST</a>
<a href="#">GetPropertyValue</a>	<a href="#">SetRow</a>
<a href="#">GetReturnProperties</a>	<a href="#">SetSearchCriteria</a>
<a href="#">GetRowsFound</a>	<a href="#">SetSearchObject</a>
<a href="#">GetSearchCriteria</a>	

See the following properties:

<a href="#">ErrDescription</a>
<a href="#">ErrNumber</a>

# **PCDPutStream**

## **PCDPutStream**

Use PCDPutStream to provide users with the ability to write the contents of physical files to disk.

### **Syntax**

`PCDPutStream. methodOrProperty`

### **Usage**

After each write, you should check the [ErrNumber](#) property, which should be zero (`S_OK`). If your write was error-free, you should check the [BytesWritten](#) property to see how many bytes were actually written to the physical file.

*Note:* If you are using a language other than JavaScript, you can use the optional second parameter for [Write](#) to get the number of bytes written. This allows you to avoid having to use the [BytesWritten](#) property to retrieve this information.

### **Example**

Because of the close relationship that PCDPutStream has with [PCDPutDoc](#), the [Example](#) in [PCDPutDoc](#) also illustrates how you can use PCDPutStream.

### **Related Items**

See the following methods:

[GetPropertyValue](#)  
[SetComplete](#)  
[Write](#)

See the following properties:

[BytesWritten](#)  
[ErrDescription](#)  
[ErrNumber](#)

# PCDRecentDoc

## PCDRecentDoc

Use the PCDRecentDoc object to retrieve recently edited documents from your DM Repository.

### Syntax

PCDRecentDoc. *methodOrProperty*

### Usage

PCDRecentDoc is similar to [PCDSearch](#) in how it operates, except that it automatically links to the ACTI VI TYLOG table based on the value of the AUTHOR and TYP1 ST columns. It filters any rows it returns to only include those that have unique entries. Thus, it is the caller's responsibility to specify search criteria, return properties, and an order-by property that satisfies the real meaning of a list of recent documents. For example, the LAST\_EDIT\_DATE column in the PROFILE table is often used as an [AddOrderByProperty](#) method. PCDRecentDoc does *not* automatically filter on the ACTI VI TY\_TYPE column in the ACTI VI TYLOG table.

### Example

The following example shows how to create an instance of PCDRecentDoc.

```
    .  
    .  
    .  
    pRecent = Server.CreateObject(  
        "PCDClient.PCDRecentDoc")  
    .  
    .  
    .
```

# PCRecentDoc

## Related Items

See the following methods:

[AddOrderByProperty](#)  
[AddReturnMetaProperty](#)  
[AddReturnProperty](#)  
[AddSearchCriteria](#)  
[AddSearchLib](#)  
[BeginGetBlock](#)  
[ColumnCount](#)  
[EndGetBlock](#)  
[Execute](#)  
[GetMetaPropertyValue](#)  
[GetMetaRowsFound](#)  
[GetPropertyValue](#)  
[GetReturnProperties](#)

[GetRowsFound](#)  
[GetSearchCriteria](#)  
[NextMetaRow](#)  
[NextRow](#)  
[ReleaseResults](#)  
[SetChunkFactor](#)  
[SetDST](#)  
[SetMaxRows](#)  
[SetMetaRow](#)  
[SetReturnProperties](#)  
[SetRow](#)  
[SetSearchCriteria](#)  
[SetSearchObject](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# PCDSearch

## PCDSearch

PCDSearch provides the user with the ability to enter criteria to execute a search of the available document servers in your DM file store. You can use this object to execute the search, then retrieve the results by randomly or sequentially accessing the data rows that are returned. You access a returned data item by referring to its property name.

### Syntax

PCDSearch. *methodOrProperty*

### Example

The following example shows how to create an instance of the PCDSearch object.

```
pSearch = Server.CreateObject(_  
    "PCDCIent.PCDSearch")
```

### Related Items

See the following methods:

AddOrderByProperty	GetRowsFound
AddReturnMetaProperty	NextMetaRow
AddReturnProperty	NextRow
AddSearchCriteria	ReleaseResults
AddSearchLib	SetChunkFactor
BeginGetBlock	SetDST
ColumnCount	SetMaxRows
EndGetBlock	SetMetaRow
Execute	SetReturnProperties
GetMetaPropertyValue	SetRow
GetMetaRowsFound	SetSearchCriteria
GetPropertyValue	SetSearchObject
GetPropertyValueByIndex	

# PCDSearc

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# PCDSQL

## PCDSQL

The PCDSQL object allows direct access to SQL tables that support the DM document management system.

**Caution:** The PCDSQL object and the methods it supports provide a set of powerful tools that allow you to modify the SQL database that supports your DM system. However, with this power also comes the power to damage or destroy the SQL database tables that DM uses to control your document management environment. Other objects in the DM API offer some degree of protection to safeguard against your accidental or inadvertent corruption of the SQL database tables. By contrast, PCDSQL methods provide your custom applications complete flexibility to modify any SQL database table in the DM library.

Unless you use SQL pass-through features correctly, you may damage your DM document management system. Examples of damage that can occur include the corruption of data in the SQL databases used by DM, the loss of system data maintained by DM, and the inability of DM to recover document objects stored in your DM repository.

Because of the inherent danger associated with the improper use of the PCDSQL object and the methods it supports, Hummingbird recommends that you incorporate this object into your custom applications only when other DM API objects do not provide the program functionality you require.

### Syntax

PCDSQL. *methodOrProperty*

### Usage

PCDSQL supports custom access to any DM libraries. The methods it exposes allow you to submit any structured query language queries to the SQL database. Metadata can be retrieved about that query and various different database objects. New database objects can be created using the tools that PCDSQL supports.

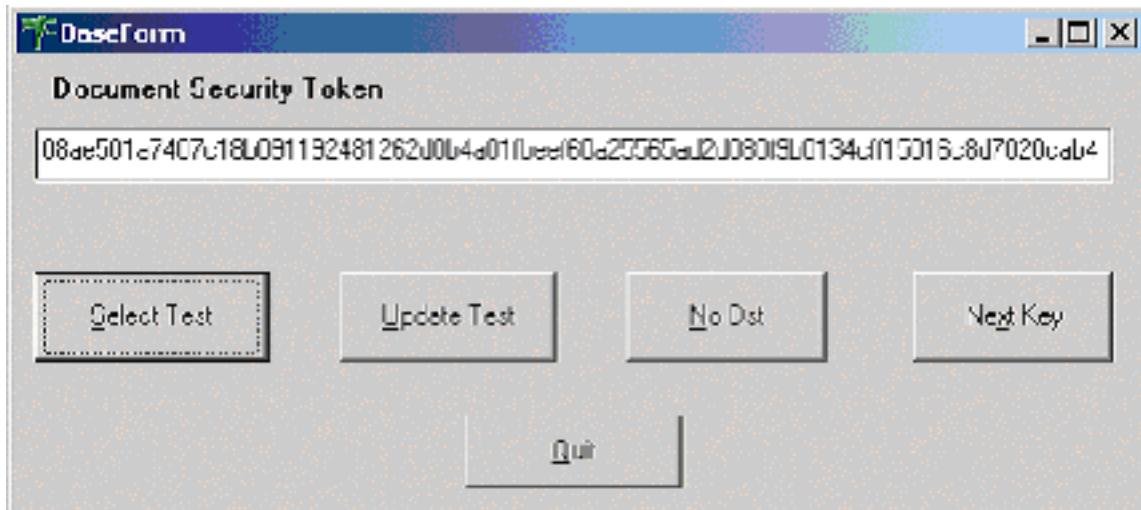
### Example

The following example exercises the methods supported by the PCDSQL object. In addition to providing several event-driven forms and the source code behind them, you should also note that the DocInternalClass object has been created as an independent object

# PCDSQL

that allows the document security token (DST) to be shared throughout the entire application.

```
' These are the Project-wide Global variables.  
Private msDST As String  
  
Private Sub Class_Initialize()  
    ' msDST will be used to handle the DST.  
    msDST = ""  
End Sub  
  
' Used to set the DST within the DocInternal Class.  
Public Sub SetDST(sInputString As String)  
    msDST = sInputString  
End Sub  
  
' Used to retrieve the DST from the DocInternal  
' Class.  
Public Function GetDST() As String  
    GetDST = msDST  
End Function
```



' Global Variable declaration section.

# PCDSQL

```
' This will be shared throughout this form.  
Public oDIC As New DocInternal Class  
  
Private Sub cbNextKey_Click()  
    NextKeyForm.oNextKey.SetDST(oDIC.GetDST())  
    NextKeyForm.Show  
End Sub  
  
Private Sub cbNoDST_Click()  
    DSTForm.Show  
End Sub  
  
Private Sub cbQuit_Click()  
    Dim oForm As Form  
    For Each oForm In Forms  
        Unload oForm  
    Next oForm  
End Sub  
  
Private Sub cbSelectTest_Click()  
    ' Pass the DST from the Base form to  
    ' the SelectForm's PCDSQL object.  
    SelectForm.oSelectSQL.SetDST(oDIC.GetDST())  
    SelectForm.Show  
End Sub  
  
Private Sub cbUpdateTest_Click()  
    ' This allows the DST has to make it from  
    ' one form to another.  
    UpdateForm.oUpdateSQL.SetDST(oDIC.GetDST())  
  
    ' Call the other form.  
    UpdateForm.Show  
  
End Sub  
  
' This is the base form for this application.  
Private Sub Form_Load()  
    ' Local Variable declarations
```

# PCDSQL

```
Dim sBuffer As String
Dim oLogin As New PCDLogin
Dim nResult As Long
Dim sTempBuf As String

' Login process.
nResult = oLogin.AddLogin(0, "MyLibrary", "", "")
nResult = oLogin.AddLogin(0, "MyDomain", _
    "MyUserID", "MyPassword")
'

' If your API application is licensed by Hummingbird
' to use SQL pass-through functionality when SQL
' pass-through functionality is disabled for general
' users, you should replace the previous 2 lines of
' code with the following two lines of code:
'   nResult = oLogin.AddLoginLicensed(0, _
'       "MyLibrary", "", _
'       "", "SQLpassthruID", _
'       "SQLpassthruLicenseID")
'   nResult = oLogin.AddLoginLicensed(0, _
'       "MyDomain", _
'       "MyUserID", _
'       "MyPassword", _
'       "SQLpassthruID", _
'       "SQLpassthruLicenseID")

nResult = oLogin.Execute()
sBuffer = oLogin.GetDST()

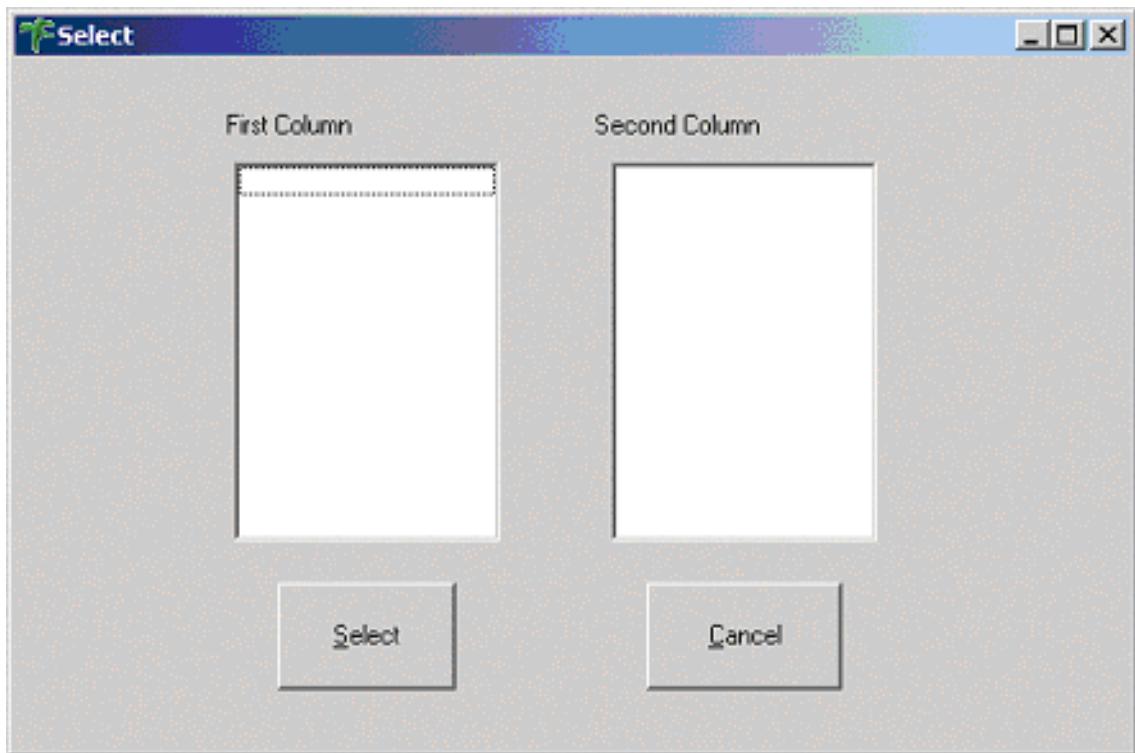
' Get And display the DST.
oDIC.SetDST(sBuffer)
tbDST.Text = oDIC.GetDST()

End Sub

' Programming Note:
' Without calling the oSelectSQL.SetDST, the
' oSelectSQL.Execute call would have failed,
```

# PCDSQL

' because without the Document Security Token  
' being set within the object, the verification of  
' the 'MyUserName' would have failed. The call  
' oSelectSQL.**SetDST** sets the value of the m\_zDST  
' protected Data Member. This holds true throughout  
' this application.



```
Public oSelectSQL As New PCDSQL

Private Sub Cancel_Click()
    Dim nResult As Long

    ' Release the result set all resources associated
    ' with the result set and the PCDSQL object.
```

# PCDSQL

```
nResult = oSelectSQL.ReleaseResults()

Unload SelectForm

End Sub

Private Sub cbSelect_Click()
' Local Variable Declarations.
Dim nNumRows As Long
Dim nColumnCount As Long
Dim nResult As Long
Dim sTempBuf As String

' Run a simple Query to return a couple of
' columns of data.
nResult = oSelectSQL.Execute("SELECT USER_ID, _
                           FULL_NAME FROM DOCSADM.PEOPLE")

' Retrieve column count from the result set.
nColumnCount = oSelectSQL.GetColumnCount()

' Insert the names of the respective columns
' into the labels above the list boxes containing
' result set data.
For i = 1 To nColumnCount
    lblColumnName(i - 1).Caption = _
        oSelectSQL.GetColumnName(i)
Next i

' Get number of rows in the result set.
nNumRows = oSelectSQL.GetRowCount()

' Populate the two lists with data returned in
' the result set.
For i = 1 To nNumRows
    nResult = oSelectSQL.SetRow(i)
    For j = 1 To nColumnCount
        lstReturnedData(j - 1).AddItem( _
```

# PCDSQL

```
oSelectSQL. GetColumnValue(j))

    Next j
    Next i

End Sub

Private Sub Form_Load()

    ' Set the default values for the labels above
    ' the columns.
    lblColumnName(0).Caption = "First Column"
    lblColumnName(1).Caption = "Second Column"

    ' Set the library that the query will use.
    oSelectSQL. SetLibrary("MyLibrary")

End Sub

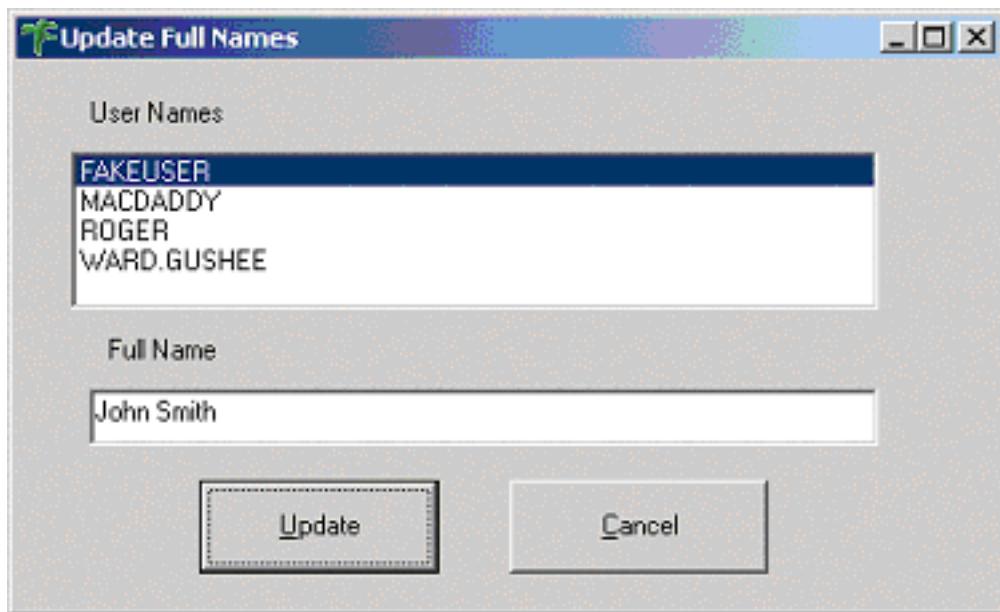
Private Sub Form_Unload(Cancel As Integer)

    Dim nResult As Long
    Dim sBuf As String

    ' Clear the result set as the object goes
    ' out of scope.
    nResult = oSelectSQL. ReleaseResults

End Sub
```

# PCDSQL



```
Public oUpdateSQL As New PCDSQL

Private Sub cbCancel_Click()
    Dim nResult As Long

    ' Release the result set for this instance of
    ' the PCDSQL object.
    nResult = oUpdateSQL.ReleaseResults()

    Unload UpdateForm

End Sub

Private Sub cbUpdate_Click()
    Dim sTempBuf As String
    Dim sSQL As String
    Dim nRowNumber As Long
```

# PCDSQL

```
Dim nResult As Long
Dim nRowCount As Long

' Determine which row is being updated.
' Then, load the data from the text box.
nRowNumber = UpdateForm.lstUserIDs.ListIndex
nRowNumber = nRowNumber + 1
nResult = oUpdateSQL.SetRow(nRowNumber)
sTempBuf = oUpdateSQL.GetColumnName(1)

sSQL = "UPDATE DOCSADM. PEOPLE SET FULL_NAME = '" _
& txtFullName.Text & "' WHERE USER_ID = '" _
sTempBuf + "'"

nResult = oUpdateSQL.Execute(sSQL)

If nResult <> 0 Then
    sTempBuf = _
        "Native SQL Error from the Update call - " _
        & Str(oUpdateSQL.GetSQLErrorCode())
    MsgBox sTempBuf
End If

' Verify that a single row was updated.
nResult = oUpdateSQL.GetRowsAffected()
If nResult = 1 Then
    oUpdateSQL.Execute("SELECT USER_ID, " _
        & "FULL_NAME FROM DOCSADM. PEOPLE")
End If

End Sub

Private Sub Form_Load()

Dim sSQL As String
Dim nNumRows As Long
Dim nResult As Long
Dim sTempBuf As String
```

# PCDSQL

```
' Set the Library.  
If nResult = 0 Then  
    nResult = oUpdateSQL.SetLibrary("MyNewLibrary")  
Else  
    MsgBox ("The SetDST call failed.")  
End If  
  
' This function does not return a SUCCESS/FAILURE  
' value. Instead, it returns the numeric value of  
' the DB Vendor.  
If nResult = 0 Then  
    nResult = oUpdateSQL.GetDBVendor()  
Else  
    MsgBox ("SetLibrary call failed")  
End If  
  
' This SQL statement retrieves two columns.  
nResult = oUpdateSQL.Execute( "SELECT " _  
    & "USER_ID, FULL_NAME FROM DOCSADM.PEOPLE")  
  
' Load put data from the SELECT into the list Box.  
' Then, determine which list box item is in focus.  
nNumRows = oUpdateSQL.GetRowCount()  
  
' Fill up the ListBox with result set data.  
For i = 1 To nNumRows  
    nResult = oUpdateSQL.SetRow(i)  
    ' Place the User ID in the buffer.  
    sTempBuf = oUpdateSQL.GetColumnValue(1)  
    UpdateForm.lstUserIDs.AddItem(sTempBuf)  
Next i  
  
' Select the first element in the listbox.  
lstUserIDs.Selected(0) = True  
  
' Match the item selected in the list box with  
' the data in the textbox.  
nResult = oUpdateSQL.SetRow(1)
```

```
txtFullName.Text = oUpdateSQL.GetColumnName(2)

End Sub

Private Sub lstUserIDs_Click()

Dim nListIndex As Long
Dim sTempBuf As String
Dim nRowIndex As Long

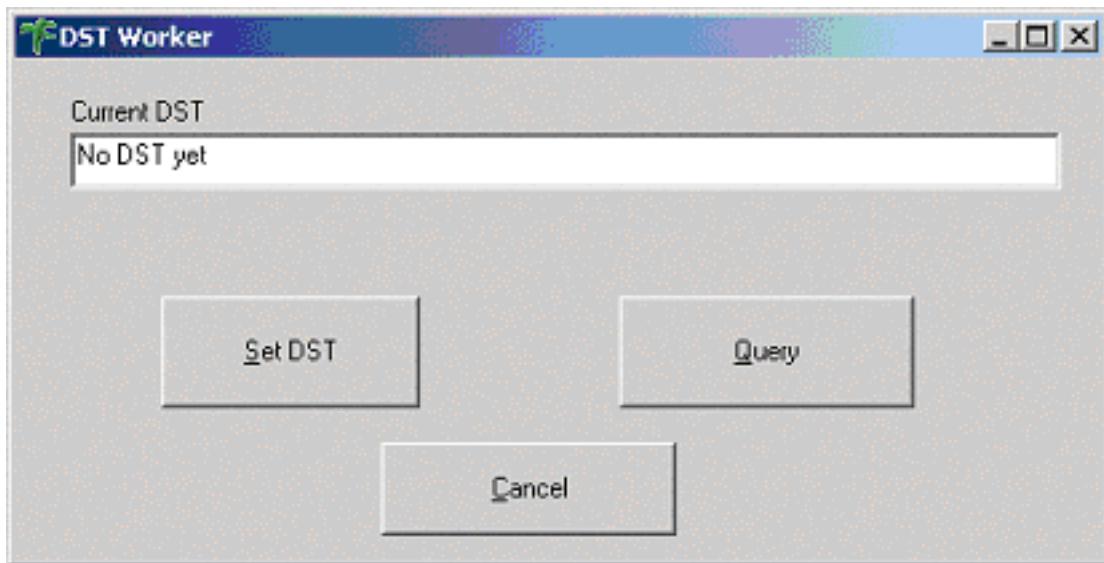
nListIndex = lstUserIDs.ListIndex

' The SQL row is one larger than the list index,
' so, add one to set the SQL row index.
nRowIndex = nListIndex + 1
nResult = oUpdateSQL.SetRow(nRowIndex)

' This is the full name column value.
sTempBuf = oUpdateSQL.GetColumnName(2)
txtFullName.Text = sTempBuf

End Sub
```

# PCDSQL



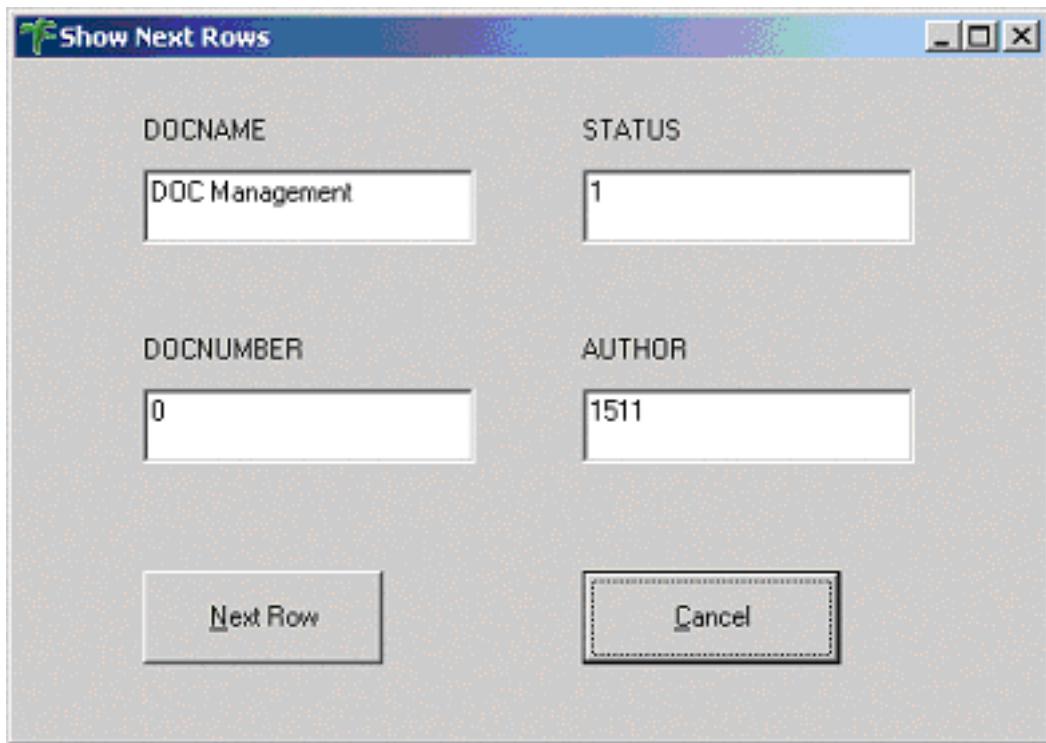
```
Dim sDST As String
Dim oDSTDIC As New DocInternal Class
Dim oDSTLogin As New PCDLogin
Public oDSTS SQL As New PCDSQL

Private Sub cbCancel_Click()
    Dim nResult As Long
    sDST = ""
    nResult = oDSTS SQL.ReleaseResults()
    Unload DSTForm
End Sub

Private Sub cbDoQuery_Click()
    nResult = oDSTS SQL.Execute("SELECT DOCNAME, " _
        & "DOCNUMBER, STATUS, AUTHOR from " _
```

```
& "DOCSADM. PROFILE")  
  
If nResult = 0 Then  
    NextRowForm.sNRDST = oDSTDIC.GetDST()  
    NextRowForm.Show  
Else  
    MsgBox ("Query Failed")  
End If  
  
End Sub  
  
Private Sub cbSetDST_Click()  
  
Dim nResult As Long  
  
' Do mandatory login process in order to get a DST.  
nResult = oDSTLogin.AddLogin(0, "MyLibrary", _  
                           "", "")  
nResult = oDSTLogin.AddLogin(0, "MyDomain", _  
                           "MyUser", "MyPassword")  
  
nResult = oDSTLogin.Execute()  
sDST = oDSTLogin.GetDST()  
  
' Get and display the DST.  
oDSTS SQL.SetDST(sDST)  
oDSTDIC.SetDST(sDST)  
  
' Show the DST in the Text box.  
txtDST.Text = sDST  
  
End Sub  
  
Private Sub Form_Load()  
    txtDST.Text = "No DST yet."  
End Sub
```

# PCDSQL



```
Public oNRSQL As New PCDSQL
Public nNumRows As Long
Public nNumCols As Long
Public nRowsViewed As Long

Private Sub cbCancel_Click()
    Dim nResult As Long

    ' Tidy up by releasing the result set.
    sNRSRST = ""
    nResult = oNRSQL.ReleaseResults()

    ' Close this form.
    Unload NextRowForm
```

```
End Sub

Private Sub cbNextRow_Click()
    Dim nResult As Long

    ' Increment the nRowsViewed count.
    nRowsViewed = nRowsViewed + 1

    ' Go to the next row.
    nResult = oNRSQL.NextRow()

    For i = 1 To nNumCols
        lblResultData(i - 1).Caption = _
            oNRSQL.GetColumnName(i)
        txtResultData(i - 1).Text = _
            oNRSQL.GetColumnValue(i)
    Next i

    ' If at end of record set, deactivate the button.
    If nRowsViewed = nNumRows Then
        cbNextRow.Enabled = False
    End If

End Sub

Private Sub Form_Load()
    Dim nResult As Long
    Dim sTempBuf As String
    Dim nRowsAffected As Long

    nResult = oNRSQL.Execute("SELECT DOCNAME, " _
        & "DOCNUMBER, STATUS, AUTHOR from " _
        & "DOCSADM.PROFILE")

    ' If the Query is successful, place the first
    ' row into the text boxes.
```

# PCDSQL

```
If nResult = 0 Then
    nNumRows = oNRSQL.GetRowCount()

    ' If there are more than one row then lets do it.
    If nNumRows > 0 Then
        nNumCols = oNRSQL.GetColumnCount()

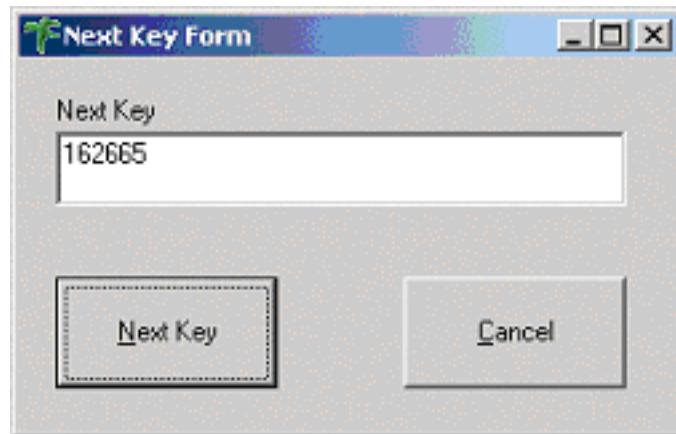
        ' Add the appropriate elements to the
        ' control array.
        For i = 1 To nNumCols
            lblResultData(i - 1).Caption = _
                oNRSQL.GetColumnName(i)
            txtResultData(i - 1).Text = _
                oNRSQL.GetColumnValue(i)
        Next i
    End If
End If

' Check the number of rows coming back.
' If more than one, then enable the
' cbNextRow button.
If nNumRows > 1 Then
    cbNextRow.Enabled = True
Else
    cbNextRow.Enabled = False
End If

nRowsViewed = 1

End Sub
```

# PCDSQL



```
Public oNextKey As New PCDSQL

Private Sub cbCancel_Click()
    Unload NextKeyForm
End Sub

Private Sub cbNextKey_Click()

    Dim sResult As String
    sResult = oNextKey.GetNextKey("")
    txtNextKey.Text = sResult

End Sub
```

# PCDSQL

## Related Items

See the following methods:

<a href="#">Execute</a>	<a href="#">GetRowsAffected</a>
<a href="#">GetColumnCount</a>	<a href="#">GetSQLErrorCode</a>
<a href="#">GetColumnName</a>	<a href="#">NextRow</a>
<a href="#">GetColumnValue</a>	<a href="#">ReleaseResults</a>
<a href="#">GetDBVendor</a>	<a href="#">SetDST</a>
<a href="#">GetNextKey</a>	<a href="#">SetLibrary</a>
<a href="#">GetRowCount</a>	<a href="#">SetRow</a>

See the following properties:

<a href="#">ErrDescription</a>
<a href="#">ErrNumber</a>

# PCDTrusteeList

## PCDTrusteeList

Use PCDTrusteeList to manipulate the trustee list associated with DM.

### Syntax

`PCDTrusteeList. methodOrProperty`

### Example

The following example shows how to create an instance of the PCDTrusteeList object.

```
pTrustees = Server.CreateObject(_  
    "PCDCIent.PCDTrusteeList")
```

### Related Items

See the following methods:

<a href="#">AddTrustee</a>	<a href="#">GetSize</a>
<a href="#">BeginIter</a>	<a href="#">GetTrusteeIndex</a>
<a href="#">DeleteTrustee</a>	<a href="#">GetTrusteeRights</a>
<a href="#">GetCurrentTrusteeFlags</a>	<a href="#">NextTrustee</a>
<a href="#">GetCurrentTrusteeName</a>	<a href="#">SetTrusteeRights</a>
<a href="#">GetCurrentTrusteeRights</a>	

See the following properties:

<a href="#">ErrDescription</a>
<a href="#">ErrNumber</a>

# **PCDTrusteeList**

# 4

## DM API Methods and Properties

### In This Chapter

This chapter describes the methods and properties associated with the objects that the DM supports, including their syntax, usage, and other related information.

# AddLogin

## AddLogin

Use this method to set the logon information that gets passed to the DM Server. The DM Server authenticates the logon information when the [Execute\(\)](#) method is called.

### Syntax

```
PCDLogi n. AddLogi n( intNetworkType, _  
    strUni tName, _  
    strUserName, _  
    strPassword )
```

### Parameters

**intNetworkType** The network type of the network alias. Valid values are:

Li brary Logon (No Network)	0
NetWare Bi ndery	1
NetWare NDS	2
Mi crosoft Network	8

**strUni tName** The DM library name, NetWare server name, NDS tree name, or Windows domain, workgroup, or server name.

**strUserName** The user name for the specified unit.

**strPassword** The password for the specified user.

### Usage

Before you do a call to `AddLogi n`, you need to get the logon libraries available on the DM Server. (See [PCDGetLoginLibs](#).)

### Example

The following is a simple example that uses a user name with an Attaché password.

```
' Avai labl e Logon types for the DM Server,
```

# AddLogin

```
' dimensions as constants.  
Const iLibraryLogon As Integer = 0  
Const iNetWareBindery As Integer = 1  
Const iNetWareNDS As Integer = 2  
Const iMicrosoftNetwork As Integer = 8  
  
' Create an instance of the Logon object.  
pClient = Server.CreateObject("PCDClient.PCDLogin")  
  
' Use the Attaché password to log on to the  
' Legal Law library.  
hr = pClient.AddLogin iLibraryLogon, _  
    "Legal Law", "t_rex", "roar"  
  
' The DM Server authenticates the Logon.  
hr = pClient.Execute()  
. . .
```

The following example captures the logon information a user enters on a form. Again, it uses an Attaché password.

```
. . .  
  
Private Type LOGIN_TYPE  
    LIBRARY_LOGIN As Integer  
    NETWORK_BINDERY As Integer  
    NETWORK_NDS As Integer  
    BANYAN_VINES As Integer  
    MS_NETWORK As Integer  
End Type  
  
' Module scope variables.  
Dim MyLogonType As LOGIN_TYPE  
  
Private Sub Command1_Click()  
    ' Set up MyLOGIN_TYPE with Logons that the
```

# AddLogin

```
' DM API supports.  
MyLogonType. LIBRARY_LOGIN = 0  
MyLogonType. NETWORK_BINDERY = 1  
MyLogonType. NETWORK_NDS = 2  
MyLogonType. MS_NETWORK = 8  
  
' Create an instance of the Logon object.  
Dim pClient As PCDLogin  
Set pClient = New PCDLogin  
  
' Get the Library, User Name, and Password from  
' the form.  
Dim Library As String  
Dim User As String  
Dim Password As String  
  
Library = txtLib.Text  
User = txtUser.Text  
Password = txtPassword.Text  
  
' Use the Attaché password to log on to  
' the indicated library.  
Dim rc As Long  
rc = pClient.AddLogin( _  
    MyLogonType. LIBRARY_LOGIN, _  
    Library, _  
    User, _  
    Password )  
  
' The DM Server authenticates  
' the logon.  
rc = pClient.Execute  
  
' Check the return code for success.  
If rc <> 0 Then MsgBox "A problem occurred."  
  
' Clean up.  
Set pClient = Nothing
```

# AddLogin

End Sub

.

## Related Items

See the following objects:

[PCDLogin](#)  
[PCDGetLoginLibs](#)

See the following methods:

[AddLoginLicensed](#)  
[Execute](#)  
[GetAliasList](#)  
[GetDOCSUserName](#)  
[GetDST](#)  
[GetFailedLoginList](#)  
[GetLoginLibrary](#)  
[GetPrimaryGroup](#)  
[SetDST](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

## AddLoginLicensed

## AddLoginLicensed

The “Allow SQL Passthrough” setting in the DM system parameter settings dialog can be used to deny access to the SQL passthrough functions that the PCDSQL object supports. However, commercial applications by third-party partners of Hummingbird can receive a license key that allows their applications to access the PCDSQL object regardless of the SQL passthrough setting.

Use this method to set the logon information that gets passed to the DM Server if the API application is licensed to use the PCDSQL object even if the “Allow SQL Passthrough” option has been disabled by the DM administrator. The DM Server authenticates the logon information when the PCDLogin.[Execute\(\)](#) method is called.

Other than the two additional input parameters, this object functions in the same way that the [AddLogin\(\)](#) method functions.

### Syntax

```
PCDLogi n. AddLogi nLi censed( intNetworkType, _  
                                strUnitName, _  
                                strUserName, _  
                                strPassword , _  
                                strAppl i cati onID, _  
                                strAppLi censeKey)
```

# AddLoginLicensed

## Parameters

intNetworkType	The network type of the network alias. Valid values are: Library Logon (No Network) 0 NetWare Bindery 1 NetWare NDS 2 Microsoft Network 8
strUnitName	The DM library name, NetWare server name, NDS tree name, or Windows domain, workgroup, or server name.
strUserName	The user name for the specified unit.
strPassword	The password for the specified user.
strApplicationID	The application ID that was assigned by Hummingbird.
strAppLicenseKey	The application key that was assigned by Hummingbird.

## Usage

Before you do a call to AddLoginLicensed, you need to get the logon libraries available on the DM Server. (See [PCDGetLoginLibs](#).)

## Example

The following is a simple example that uses a user name with an Attaché password.

```
' The API application method must receive an
' application ID and a license key from Hummingbird
' Ltd. in order to use this login method.
Dim oPCDLogin As New PCDLogin
Dim lngResult As Long
lngResult = oPCDLogin.AddLoginLicensed( 0 , _
                                         strMyLibrary, " ", " ", _
                                         strApplD, strLicenseKey )
lngResult = oPCDLogin.AddLoginLicensed( 0 ,
```

# AddLoginLicensed

```
    strMyDomain, strMyUserID, _  
    strMyPassword, strAppID, _  
    strLicenseKey )  
oPCDLogin.Execute  
. . .
```

## Related Items

See the following objects:

[PCDLogin](#)  
[PCDGetLoginLibs](#)

See the following methods:

[AddLogin](#)  
[Execute](#)  
[GetAliasList](#)  
[GetDOCSUserName](#)  
[GetDST](#)  
[GetFailedLoginList](#)  
[GetLoginLibrary](#)  
[GetPrimaryGroup](#)  
[SetDST](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# AddOrderByProperty

## AddOrderByProperty

AddOrderByProperty allows you to set the order that search results are returned to you.

### Syntax

```
PCDLookup. AddOrderByProperty( strPropertyName, _  
    bl nAscending )
```

```
PCDRecentDoc. AddOrderByProperty( strPropertyName, _  
    bl nAscending )
```

```
PCDSearch. AddOrderByProperty( strPropertyName, _  
    bl nAscending )
```

### Parameters

strPropertyName      The name of the property by which to order the results of a search.

bl nAscending      A Boolean flag indicating whether to sort on the name property in ascending or descending order. FALSE (or zero) sorts in descending order. Any other value (not FALSE) sorts in ascending order.

### Returns

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Example

The following example uses the AUTHOR\_ID value to display search results in ascending order.

```
...  
...  
' Save the document security token (DST) from the  
' DM Server so the search can be done.  
strDST = objLogIn.GetDST
```

# AddOrderByProperty

```
' Create the search object.  
Dim pSearch As New PCDSearch  
  
' Set the document security token (DST).  
pSearch. SetDST strDST  
  
' Add a search library to search.  
pSearch. AddSearchLib "Legal Law"  
  
' Set which form to use for the search.  
pSearch. SetSearchObject "def_qbe"  
  
' Add properties you want the search to return.  
pSearch. AddReturnProperty "docname"  
pSearch. AddReturnProperty "docnum"  
pSearch. AddReturnProperty "AUTHOR_ID"  
  
' Sort the search results by AUTHOR_ID in  
' ascending order.  
pSearch. AddOrderByProperty "AUTHOR_ID", TRUE  
  
' Execute the Search.  
pSearch. Execute  
. . .
```

## Related Items

See the following objects:

[PCDLookup](#)  
[PCDRecentDoc](#)  
[PCDSearch](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# AddProperty

## AddProperty

Use this method to add a property/value pair to a property value list.

### Syntax

```
PCDPropertyList.AddProperty( strPropertyName, _  
                           vntPropVal )
```

### Parameters

strPropertyName	The string that identifies the name of the property to add to the list.
vntPropVal	The VARIANT that contains the value of the property that is to be added to the list.

### Returns

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Example

The [Example](#) for the [PCDLookup](#) Object uses the AddProperty method.

### Related Items

See the [PCDPropertyList](#) object.

See the following methods:

[DeleteProperty](#)  
[GetPropertyIndex](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

## AddReturnMetaProperty

## AddReturnMetaProperty

The AddReturnMetaProperty method allows you to retrieve information about the data requested in a search. For example, your custom application can use AddReturnMetaProperty to determine if an object returned in your search result set is supposed to be visible to the user or not.

The available metadata properties are %PropertyName, %Title, %Visible, and %Data. Each row in the metadata property list offers the following information:

- %PropertyName is the name of the property as it is identified on the base form. It can have the value “\_UNKNOWN\_” if there is no corresponding property on the base form.
- %Title is the title of this column. The %Title property often contains the prompt that describes the data when it is presented to the user. It can be blank.
- %Visible is a flag indicating whether or not this column should be displayed to the user.
- %Data is the value of this column in the current row of the data (as opposed to the metadata). For example, %PropertyName could return “AUTHOR\_ID”, and %Data could return “J\_SMI TH”.

*Note:* For compatibility purposes, AddReturnMetaProperty also supports these property names as an alternative to the preferred terms shown above: PROPNAME, TITLE, VI SIBLE, and DATA.

### Syntax

```
PCDRecentDoc. AddReturnMetaProperty( strPropName  
 )
```

```
PCDSearch. AddReturnMetaProperty( strPropName )
```

# AddReturnMetaProperty

## Parameter

`strPropertyName` The name of the property to be returned in the results. This should be one of the four properties listed above.

## Returns

Returns a VARIANT return value that contains the value of the requested property.

## Related Items

See the following objects:

[PCDRecentDoc](#)

[PCDSearch](#)

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# AddReturnProperty

## AddReturnProperty

Use this method to add a property to the list of objects that you want a search to return to you. No data is returned by default. Your search will return no data unless you call AddReturnProperty at least one time before you execute your search.

### Syntax

```
PCDRecentDoc. AddReturnProperty( strPropertyName )
```

```
PCDSearch. AddReturnProperty( strPropertyName )
```

### Parameter

strPropertyName	The name of the property to be returned in the results.
-----------------	---

### Returns

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Example

The following example shows how to add return properties as part of the execution of a search.

```
' Save the document security token (DST) from the
' DM Server so the search will execute
' properly.
strDST = pClient. GetDST()

' Create the search object.
pSearch = Server.CreateObject( _
    "PCDClient.PCDSearch")

' Pass the DST to the search object.
```

# AddReturnProperty

```
pSearch. SetDST( strDST )  
  
' Add the Library you want to search.  
pSearch. AddSearchLib( "Legal Law" )  
  
' Specify the form to use for this search.  
pSearch. SetSearchObject( "def_qbe" )  
  
' Add the properties you want the search  
' to return.  
pSearch. AddReturnProperty "docname"  
pSearch. AddReturnProperty "docnum"  
pSearch. AddReturnProperty "AUTHOR_ID"  
  
' Execute the Search.  
pSearch. Execute  
. . .
```

## Related Items

See the following objects:

[PCDRecentDoc](#)  
[PCDSearch](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# AddSearchCriteria

## AddSearchCriteria

Use this method to add a property name with search criteria to the search.

### Syntax

```
PCDGetDoc. AddSearchCriteria( strPropertyName, _  
                           strCriteria )  
  
PCDLookup. AddSearchCriteria( strPropertyName, _  
                           strCriteria )  
  
PCDPutDoc. AddSearchCriteria( strPropertyName, _  
                           strCriteria )  
  
PCDRecentDoc. AddSearchCriteria( strPropertyName, _  
                           strCriteria )  
  
PCDSearch. AddSearchCriteria( strPropertyName, _  
                           strCriteria )
```

### Parameters

`strPropertyName`      The name of the property on which to base the search.

`strCriteria`      The property value on which to search.

### Returns

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Example

The following example adds search criteria to a search. The user has entered a Document number and Author ID for which to search.

```
'  
.  
.  
.  
' Save the document security token (DST) from the  
' DM Server so the search can execute.
```

# AddSearchCriteria

```
DST = pClient.GetDST()

' Now that you have the DST, create a search object.
pSearch = Server.CreateObject(_
    "PCDClient.PCDSearch")

' Pass the DST to the search object.
pSearch.SetDST( DST )

' Add a search Library to the search.
pSearch.AddSearchLib("Legal Law")

' Specify the form to use for this search.
pSearch.SetSearchObject("def_qbe")

' Add the return properties you want returned.
pSearch.AddReturnProperty("docname")
pSearch.AddReturnProperty("docnum")
pSearch.AddReturnProperty("AUTHOR_ID")

' Before setting search criteria, get the
' values from the form the user submitted.
AuthorValue = txtAuthor.Text
DocnumValue = CInt( txtDocNum.Text )

' Add whatever search criteria you want.
pSearch.AddSearchCriteria("%MAXDAYS", "30")
If (AuthorValue <> "") Then
    pSearch.AddSearchCriteria(_
        "AUTHOR_ID", AuthorValue)
If (DocnumValue <> "") Then
    pSearch.AddSearchCriteria("docnum", DocnumValue)

' Execute the search.
psearch.Execute
.
.
```

# AddSearchCriteria

## Related Items

See the following objects:

[PCDGetDoc](#)  
[PCDLookup](#)  
[PCDPutDoc](#)  
[PCDRecentDoc](#)  
[PCDSearch](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# AddSearchLib

## AddSearchLib

Use this method to add a library name to the list of libraries to search. Your search will return no data unless you call AddSearchLib at least one time before you execute your search.

### Syntax

```
PCDLookup. AddSearchLib( strLibName )
PCDRecentDoc. AddSearchLib( strLibName )
PCDSearch. AddSearchLib( strLibName )
```

### Parameter

strLibName	The name of the library to search. This name has to match a value in the LIBRARY_NAME column of the REMOTE_LIBRARIES table of the DM library where the user is currently logged on.
------------	---

### Returns

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Example

The following adds a search library from a library selected on a form the user has submitted.

```
' Save the document security token (DST) from the
'DM Server so the search can execute.
strDST = pClient.GetDST()

' Now that you have the DST, create a search object.
pSearch = Server.CreateObject(
    "PCDClient.PCDSearch")
```

# AddSearchLib

```
' Pass the DST to the search object.  
pSearch. SetDST( DST )  
  
' Get the search library from the form.  
searchLib = txtLibrary.Text  
  
' Add the Search Library to the Search object.  
pSearch. AddSearchLib(searchLib)  
  
' Execute the search.  
pSearch. Execute
```

## Related Items

See the following objects:

[PCDLookup](#)  
[PCDRecentDoc](#)  
[PCDSearch](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# AddTrustee

## AddTrustee

Use this method to add a trustee to the trustee list. If there is already a trustee in the list that matches the trustee name and flags, the rights for the existing entry will be updated rather than a new entry added.

### Syntax

```
PCDTrusteeList.AddTrustee( strTrusteeName, _  
                           intTrusteeFlags, _  
                           intTrusteeRights )
```

### Parameters

strTrusteeName	The (BSTR) input string with the name of the trustee.
intTrusteeFlags	Input integer with the trustee flags. Supported values are as follows:  PCD_TRUSTEE_UNKNOWN_TYPE = 0 PCD_TRUSTEE_GROUP_TYPE = 1 PCD_TRUSTEE_PERSON_TYPE = 2
intTrusteeRights	Input integer with the trustee rights. These values are taken from the ACCESSRIGHTS column in the SECURITY table of the SQL database.

### Returns

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Example

The section titled [Getting and Updating Trustee Information](#) in Chapter 1 illustrates the use of the AddTrustee method.

### Related Items

See the [PCDTrusteeList](#) object.

See the following methods:

# AddTrustee

[BeginIter](#)  
[DeleteTrustee](#)  
[GetCurrentTrusteeFlags](#)  
[GetCurrentTrusteeName](#)  
[GetCurrentTrusteeRights](#)  
[GetSize](#)  
[GetTrusteeIndex](#)  
[GetTrusteeRights](#)  
[NextTrustee](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# AddUserFilterCriteria

## AddUserFilterCriteria

Use this method to add a property name with user filter criteria to a lookup.

### Syntax

```
PCDLookup.AddUserFilterCriteria( strPropertyName, _  
strCriteria )
```

### Parameters

**strPropertyName**      The name of the property on which to base the search.

**strCriteria**      The property value on which to search.

### Returns

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Example

The [Example](#) in the discussion of the [PCDLookup](#) object illustrates the use of the AddUserFilterCriteria method.

### Related Items

See the [PCDLookup](#) object.

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# **BeginGetBlock**

## **BeginGetBlock**

Use this method to tell the object that you are beginning a block of “Get” operations so that it will hold the results until [EndGetBlock](#) is called. Normally, the result set is released and recreated on each call to, for example, [GetPropertyValue](#). The results remain cached on the server, but the interface is normally released to prevent it from timing out. If you are doing a block of operations such as [GetPropertyValue](#), with no intervening user interaction or other operations that could cause an indefinite wait between calls, you can use [BeginGetBlock](#) and [EndGetBlock](#) to improve performance.

### **Syntax**

`PCDRecentDoc. BeginGetBlock()`

`PCDSearch. BeginGetBlock()`

### **Returns**

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### **Related Items**

See the following objects:

[PCDRecentDoc](#)

[PCDSearch](#)

See the following method: [EndGetBlock](#).

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# **BeginIter**

## **BeginIter**

Use this method to position the current pointer to the first position in a list. This method begins a BeginIter()/[NextProperty\(\)](#), a BeginIter/[NextRow](#), or a BeginIter()/[NextTrustee\(\)](#) loop.

### **Syntax**

PCDPropertyList. BeginIter()

PCDPropertyLists. BeginIter()

PCDTrusteeList. BeginIter()

### **Returns**

It returns S\_OK as an HRESULT if it successfully positioned on a list entry, and returns PCD\_S\_END\_OF\_LIST if the list is empty. JavaScript, Visual Basic, and VBScript return this as a function value. This value is also available in the [ErrNumber](#) property.

### **Example**

The BeginIter method is used in the [Example](#) for the [PCDPropertyList](#) object.

### **Related Items**

See the following objects:

[PCDPropertyList](#)

[PCDPropertyLists](#)

[PCDTrusteeList](#)

See the following methods:

[NextProperty](#)

[NextRow](#)

[NextTrustee](#)

See the following properties:

# BeginIter

[ErrDescription](#)  
[ErrNumber](#)

# BytesRead

## BytesRead

This property is always set during a [Read](#) operation. It contains the number of bytes returned by the call to the [Read](#) method.

### Syntax

`PCDGetStream. BytesRead()`

### Returns

Returns the number of bytes read during a [Read](#) operation. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Example

The section titled [Fetching a DM Document Object](#) in Chapter 1 illustrates how to use this method.

### Related Items

See the [PCDGetStream](#) object.

See the following methods:

[GetPropertyValue](#)

[Read](#)

[Seek](#)

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# BytesWritten

## BytesWritten

This property is always set during a [Write](#) operation. It contains the number of bytes written to disk by the call to the [Write](#) method.

### Syntax

`PCDPutStream.BytesWritten()`

### Returns

Returns the number of bytes written during a [Write](#) operation.

Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Example

The [Example](#) in [PCDPutDoc](#) illustrates the use of this object.

### Related Items

See the [PCDPutStream](#) object.

See the following methods:

[GetPropertyValue](#)

[Write](#)

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# **ClearOrderByProperties**

## **ClearOrderByProperties**

Use this method to clear properties used to order results from a lookup.

### **Syntax**

`PCDLookup.ClearOrderByProperties()`

### **Returns**

Returns an `HRESULT` to receive the result of the call. `S_OK` indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### **Example**

The [Example](#) in the discussion of the `PCDLookup` object illustrates how you can use this method in your custom applications.

### **Related Items**

See the [PCDLookup](#) object.

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# **ClearUserFilterCriteria**

## **ClearUserFilterCriteria**

Use this method to clear user filter criteria from a lookup.

### **Syntax**

```
PCDLookup. ClearUserFilterCriteria()
```

### **Returns**

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### **Example**

The [Example](#) in the discussion of the PCDLookup object illustrates the use of this method.

### **Related Items**

See the [PCDLookup](#) object.

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# Clone

## Clone

This method clones a duplicate copy of the current [PCDEnumPropertyLists](#) object.

### Syntax

```
PCDEnumPropertyLists.Clone()
```

### Returns

This object returns a pointer to the newly cloned [PCDEnumPropertyLists](#) object.

### Related Items

See the [PCDEnumPropertyLists](#) object.

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# ColumnCount

## ColumnCount

Use this method to return the number of columns in the return data. This will always be equal to or greater than the number of times [AddReturnProperty](#) is called. (In some situations, the DM API returns extra columns by default, which can increase the ColumnCount value.)

### Syntax

`PCDLookup.ColumnCount()`

`PCDRecentDoc.ColumnCount()`

`PCDSearch.ColumnCount()`

### Returns

Returns the column count as an integer.

### Example

The [Example](#) in the discussion of the PCDLookup object illustrates the use of this method.

### Related Items

See the following objects:

[PCDLookup](#)

[PCDRecentDoc](#)

[PCDSearch](#)

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# Create

## Create

Use this method to create the object that has been identified by the [SetObjectType](#) method.

### Syntax

```
PCDDocObj ect. Create()
```

### Returns

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Example

The [Example](#) in the discussion of the [PCDPutDoc](#) object illustrates the use of this method.

### Related Items

See the [PCDDocObject](#) object.

See the following methods:

[Delete](#)

[Fetch](#)

[Update](#)

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# Delete

## Delete

Use this method to delete the DM object described by the information that has been set.

### Syntax

`PCDDocObject.Delete()`

### Returns

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Related Items

See the [PCDDocObject](#) object.

See the following methods:

[Create](#)

[Fetch](#)

[Update](#)

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# DeleteProperty

## DeleteProperty

Use this method to delete a property from the property list based on its index (relative) position in the list. DeleteProperty uses a zero-based index. After an item is deleted from the list, the list repacks itself.

### Syntax

```
PCDPropertyList.DeleteProperty( LongNdx )
```

### Parameter

LongNdx	The index value that identifies the relative position of the property to delete.
---------	--

### Returns

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Usage

Indexes in the list are not guaranteed to match the position in which they were first added to the list. For example, assume you have a [PCDPropertyList](#) of four items, with index values from 0 to 3. Perhaps you want to delete the item with index value 1 and update the item with index value 3. After you delete the first item, the list repacks itself. Now the item you want to update has an index value of 2, not 3. For this reason, you should use the [GetPropertyIndex](#) method to determine the index item to delete.

### Example

See the [Example](#) in the discussion of the [PCDPropertyList](#) object for sample code that uses the DeleteProperty method.

### Related Items

See the [PCDPropertyList](#) object.

See the following methods:

# DeleteProperty

[AddProperty](#)  
[BeginIter](#)  
[GetCurrentPropertyName](#)  
[GetCurrentPropertyValue](#)  
[GetPropertyIndex](#)  
[NextProperty](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# DeleteTrustee

## DeleteTrustee

Use this method to delete from the list of trustees a trustee at a given offset.

### Syntax

```
PCDTrusteeList.DeleteTrustee( LongNdx )
```

### Parameter

LongNdx	An unsigned long integer that is set to the offset into the list of trustees that identifies the trustee that is to be deleted.
---------	---

### Returns

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Related Items

See the [PCDTrusteeList](#) object.

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# **EndGetBlock**

## **EndGetBlock**

EndGetBlock informs the server that you finished retrieving a series of items. This allows the DM API to release the result set to minimize the possibility of time outs in other operations that may be pending.

### **Syntax**

PCDRecentDoc. EndGetBlock()

PCDSearch. EndGetBlock()

### **Returns**

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### **Related Items**

See the following objects:

[PCDRecentDoc](#)

[PCDSearch](#)

See the [BeginGetBlock](#) method.

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# **ErrDescription**

## **ErrDescription**

This property provides a text description of the status from the last call. All DM API objects inherit this property.

### **Syntax**

`PCDError. ErrDescription`

### **Returns**

Returns a string value (BSTR) to receive the error description. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### **Example**

This reference guide contains many examples that show the use of the ErrDescription property, starting with the [Providing Library Access](#) section in Chapter 1.

### **Related Items**

See the [PCDError](#) object.

See the [ErrNumber](#) property.

# ErrNumber

## ErrNumber

This property provides a numeric status that identifies any error condition resulting from the last call.

### Syntax

`PCDError. ErrNumber`

### Returns

Returns a pointer to a long integer to receive the numeric status. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Example

This reference guide contains many examples that show the use of the ErrNumber property, starting with the [The Logon Process](#) section in Chapter 1.

### Related Items

See the [PCDError](#) object.

See the [ErrDescription](#) property.

# Execute

## Execute

The Execute method is supported for use in many API objects. After you set any necessary criteria or other values (for example, logon information or a search form), use this method to execute the request on the DM Server.

### Syntax

```
PCDASPFInfo. Execute()  
PCDGetDoc. Execute()  
PCDGetLogins. Execute()  
PCDLogin. Execute()  
PCDLookup. Execute()  
PCDPropertyLists. Execute()  
PCDPutDoc. Execute()  
PCDRecentDoc. Execute()  
PCDSearch. Execute()  
PCDSQL. Execute( strSQLStatement )
```

### Parameter

**strSQLStatement** A valid SQL statement that is correctly formatted to run against the current DM library. The syntax used must conform to the requirements of your SQL software.  
This parameter is only supported within the [PCDSQL](#) object. The Execute method takes no parameters when used with the other DM API objects that support it.

### Returns

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

# Execute

## Example

This reference guide contains many examples that show the use of the Execute method, starting with the [The Logon Process](#) section in Chapter 1.

## Related Items

See the following objects:

[PCDGetDoc](#)  
[PCDGetForm](#)  
[PCDGetLoginLibs](#)  
[PCDLogin](#)  
[PCDLookup](#)  
[PCDPropertyLists](#)  
[PCDPutDoc](#)  
[PCDRecentDoc](#)  
[PCDSearch](#)  
[PCDSQL](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# Fetch

## Fetch

Use this method to retrieve information about an object.

### Syntax

```
PCDDocObj ect. Fetch()
```

### Returns

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Example

Several examples in this reference guide use the Fetch method. The section titled [Fetching a DM Document Object](#) in Chapter 1 is particularly illustrative.

### Related Items

See the [PCDDocObject](#) object.

See the following methods:

[Create](#)

[Delete](#)

[Update](#)

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# FetchTrustees

## FetchTrustees

Use this method to fetch trustees from the DM Server.

### Syntax

```
PCDDocObj ect. FetchTrustees()
```

### Returns

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Example

The following example illustrates how you can use the FetchTrustees method when managing trustee rights on your document objects.

```
'  
'  
'.  
  
Private Sub cmdAddTrustees_Click()  
    ' This subroutine demonstrates adding trustees  
    ' to a document. You can add trustees in two  
    ' ways: You can use the SetTrustee method that  
    ' DocObject supports, or you can use the  
    ' AddTrustee method that the PCDTrusteeList  
    ' supports. This example demonstrates the first  
    ' method.  
  
    ' Use the following flags when adding trustees:  
    ' 0 = Unknown (Let the server figure it out.)  
    ' 1 = Group  
    ' 2 = User  
  
    ' Create the PCDDocObject. Note that late  
    ' binding is used to create the PCDDocObject  
    ' because the HasRight method will bomb if  
    ' early binding is used.  
    ListResultSet.Clear  
    Dim obj DOC As Object
```

# FetchTrustees

```
Set obj DOC = CreateObject( _
    "PCDClient.PCDDocObject")

' Create the property list object.
Dim obj PropList As New PCDPropertyList

' Set the library.
obj DOC. SetProperty "%TARGET_LIBRARY", strLib

' Set the DST.
obj DOC. SetDST strDST

' Set the Form.
obj DOC. SetObjectType "DEF_PROF"

' Identify the document object whose trustee
' properties are to be updated.
obj DOC. SetProperty "%OBJECT_IDENTIFIER", _
    txtDocNum. Text

' Create variables to store security data.
Dim LDefaultRights As Long
Dim LEffectiveRights As Long
Dim LAccessControl As Long

' Get the document.
obj DOC. Fetch
' Check for any errors.
If obj DOC. ErrNumber <> 0 Then
    lstResultSet.AddItem "Unable to fetch doc: " _
        & txtDocNum. Text
    GoTo bye
End If

' Retrieve properties from the fetched document.
Set obj PropList = obj DOC. GetReturnProperties

' Store the results.
```

# FetchTrustees

```
' Does the document have security?  
' 0 = no; 1 = yes  
LDefaultRights = _  
    obj PropList. GetPropertyvalue("SECURITY")  
  
' Determine the effective rights of the user  
' who is currently logged on.  
  
LEffectiveRights = _  
    obj PropList. GetPropertyvalue( _  
        "%EFFECTIVERIGHTS")  
  
' Determine whether the user who is currently  
' logged on has rights to edit the security  
' settings of this document: 0 = no; 1 = yes  
LAccessControl = obj DOC. HasRight( _  
    "%PR_ACCESS_CONTROL", LEeffectiveRights)  
  
If (LAccessControl) Then  
    ' Show user the list of trustees.  
    obj DOC. FetchTrustees  
  
    ' Update the trustees.  
    Dim sTrustee As String  
    sTrustee = InputBox("Enter user or group " _  
        & "to add as a trustee to document " _  
        & txtDocNum.Text, "Add Trustee", strUser)  
    obj DOC. SetTrustee sTrustee, 0, 128  
    obj DOC. UpdateTrustees  
    If (obj DOC. ErrNumber <> 0) Then  
        lstResultSet.AddItem _  
            "Unable to update trustees."  
        GoTo bye  
    End If  
    obj DOC. Update  
Else  
    lstResultSet.AddItem "You do not have " _
```

# FetchTrustees

```
& "permission to edit the security of " _  
& "this document: " & txtDocNum.Text  
End If  
lstResultSet.AddItem "Trustees were updated."  
  
bye:  
Set objPropList = Nothing  
Set objDOC = Nothing  
  
End Sub
```

## Related Items

See the following objects:

[PCDDocObject](#)  
[PCDTrusteeList](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# GetAliasList

## GetAliasList

Use this method to get a pointer to the [PCDNetAliasList](#) object that contains a list of all the network aliases that the DM Server has validated.

### Syntax

```
PCDLogi n. GetAl i asLi st()
```

### Returns

Returns a pointer to a [PCDNetAliasList](#) object. Check the [ErrNumber](#) property for any errors that may have occurred.

### Example

The GetAliasList method is used in the section about [Providing Library Access](#) in Chapter 1.

### Related Items

See the following objects:

[PCDLogin](#)  
[PCDNetAliasList](#)

See the following methods:

[Execute](#)  
[GetFailedLoginList](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

## GetAt

## GetAt

Use this method to get a Library name string at a specific index in the list of logon libraries.

### Syntax

```
PCDGetLogi nLi bs. GetAt( l ngI ndx )
```

### Parameters

l ngI ndx	A long integer that identifies the index value of the library being requested.
-----------	--

### Example

The section that discusses [Getting a List of Available Libraries](#) in Chapter 1 shows how you can use the GetAt method in your applications.

### Returns

Returns a pointer to a VARI ANT to receive the library name string. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Related Items

See the [PCDGetLoginLibs](#) object.

See the following methods:

[Execute](#)

[GetSize](#)

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

## GetColumnCount

## GetColumnCount

The GetColumnCount method retrieves the number of SQL columns in a result set. It references the current result set within the [PCDSQL](#) object.

### Syntax

```
PCDSQL. GetColumnCount()
```

### Returns

The GetColumnCount method returns a long integer that shows the number of columns in the current result set. If an INSERT or UPDATE SQL command is run, then GetColumnCount returns the number of columns in the insert or update call.

### Example

The example in the discussion of the [PCDSQL](#) object in Chapter 3 illustrates the use of GetColumnCount.

### Related Items

See the [PCDSQL](#) object.

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# GetColumnName

## GetColumnName

This method retrieves the name of the specified column in the current SQL result set.

### Syntax

```
PCDSQL. GetColumnName( LongColNum )
```

### Parameter

LongColNum	A long integer that corresponds to the column number of the desired column name.
------------	--

### Returns

The GetColumnName method returns a variant variable that contains the name of the SQL column of interest. GetColumnName returns an empty variant when asked to return erroneous data.

### Example

The example in the discussion of the [PCDSQL](#) object in Chapter 3 illustrates the use of GetColumnName.

### Related Items

See the [PCDSQL](#) object.

See the [GetSQLErrorCode](#) method.

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

## GetColumnValue

## GetColumnValue

This method retrieves the data contents of the specified column in the current SQL result set.

### Syntax

```
PCDSQL. GetColumnValue( LongColNum )
```

### Parameter

LongColNum	A long integer that corresponds to the column number of the desired column value.
------------	---

### Returns

The GetColumnValue method returns a variant variable that contains the data contained in the SQL column of interest. GetColumnValue returns an empty value when asked to return erroneous data.

### Example

The example in the discussion of the [PCDSQL](#) object in Chapter 3 illustrates the use of GetColumnValue.

### Related Items

See the [PCDSQL](#) object.

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# GetCurrentPropertyName

## GetCurrentPropertyName

Use this method to get the name of the property to which the current property pointer points. The [PCDPropertyList](#) object maintains a current property pointer so that you can iterate through the entire list of stored properties without knowing their names. The [PCDPropertyLists](#) object also supports this method as a way to access properties in a collection of property lists.

### Syntax

```
PCDPropertyList. GetCurrentPropertyName()
```

```
PCDPropertyLists. GetCurrentPropertyName()
```

### Returns

Returns a string (BSTR) variable that contains the name of the property at the current position in the list.

### Example

The [Example](#) in the discussion of the [PCDPropertyList](#) object shows how you can use this method.

### Related Items

See the following objects:

[PCDPropertyList](#)  
[PCDPropertyLists](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# **GetCurrentPropertyValue**

## **GetCurrentPropertyValue**

Use this method to get the value of the property to which the current property pointer points. The [PCDPropertyList](#) maintains a current property pointer so that you can iterate through the entire list of stored properties without knowing their names. The PCDPropertyLists object also supports this method as a way to access property values in a collection of property lists.

### **Syntax**

```
PCDPropertyList. GetCurrentPropertyValue()
```

```
PCDPropertyLists. GetCurrentPropertyValue()
```

### **Returns**

Returns a VARIANT value with the contents of the current property list element.

### **Example**

The [Example](#) contained in the [PCDPropertyList](#) object presentation in Chapter 3 shows how you can use the `GetCurrentPropertyName` method in your custom applications.

### **Related Items**

See the following objects:

[PCDPropertyList](#)  
[PCDPropertyLists](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# GetCurrentTrusteeFlags

## GetCurrentTrusteeFlags

Use this method in a [BeginIter/NextTrustee](#) loop to get the trustee flags set for the current entry in the list.

### Syntax

```
PCDTrusteeList.GetCurrentTrusteeFlags()
```

### Returns

Returns a pointer to an integer to receive the trustee flags that have been set. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value. GetCurrentTrusteeFlags supports the following values:

Trustee Flag	Value
PCD_TRUSTEE_UNKNOWN_TYPE	0
PCD_TRUSTEE_GROUP_TYPE	1
PCD_TRUSTEE_PERSON_TYPE	2

### Related Items

See the [PCDTrusteeList](#) object.

See the following methods:

[BeginIter](#)  
[GetCurrentTrusteeName](#)  
[GetCurrentTrusteeRights](#)  
[NextTrustee](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

## GetCurrentTrusteeName

## GetCurrentTrusteeName

Use this method in a [BeginIter/NextTrustee](#) loop to get the trustee name for the current entry in the list.

### Syntax

`PCDTrusteeList.GetCurrentTrusteeName()`

### Returns

Returns a string (BSTR) with the trustee name. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Related Items

See the [PCDTrusteeList](#) object.

See the following methods:

[BeginIter](#)  
[GetCurrentTrusteeFlags](#)  
[GetCurrentTrusteeRights](#)  
[NextTrustee](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# GetCurrentTrusteeRights

## GetCurrentTrusteeRights

Use this method in a [BeginIter/NextTrustee](#) loop to get the trustee rights for the current entry in the list.

### Syntax

```
PCDTrusteeList.GetCurrentTrusteeRights()
```

### Returns

Returns a pointer to an integer to receive the trustee rights for the current trustee. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Related Items

See the [PCDTrusteeList](#) object.

See the following methods:

[BeginIter](#)  
[GetCurrentTrusteeFlags](#)  
[GetCurrentTrusteeName](#)  
[NextTrustee](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

## GetDBVendor

## GetDBVendor

The GetDBVendor method retrieves the numeric identifier associated with the SQL database software in use with the current DM library.

### Syntax

```
PCDSQL. GetDBVendor()
```

### Returns

GetDBVendor returns a long integer that identifies the type of the current SQL database software. The following table lists the currently supported SQL software products and their associated identification numbers:

Identifier	Name of SQL Software
3	Microsoft SQL Server
5	Oracle Database Server
6	Sybase SQL Server

### Example

The [Example](#) in the discussion of the [PCDSQL](#) object illustrates the use of the GetDBVendor method.

### Related Items

See the [PCDSQL](#) object.

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# GetDomainList

## GetDomainList

This method allows you to retrieve a list of all other network domains that are accessible from the network identified as the root domain.

### Syntax

```
PCDNetworkInfo.GetDomainList( strNetworkOS,  
                             strRootDomain )
```

### Parameters

**strNetworkOS** A string variable that identifies the network operating system that is currently running. Valid values are:

- %NI\_ADS  
(Active Directory Services)
- %NI\_NDS (a Novell network)
- %NI\_NT (a Microsoft network)

**strRootDomain** A string variable that identifies the root domain. Use the value "%UNDEFINED" to identify all domains accessible from the current user's base domain.

### Returns

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Example

See the [Example](#) in the discussion of the [PCDNetworkInfo](#) object for sample code that illustrates how you can use the GetDomainList method in your custom applications.

### Related Items

See the [PCDNetworkInfo](#) object.

See the following properties:

# GetDomainList

ErrDescription  
ErrNumber

# GetDOCSUserName

## GetDOCSUserName

After calling [PCDLogin.Execute](#), use this method to determine the DM USER\_ID with which the user was logged on to the DM library.

### Syntax

```
PCDLogin.GetDOCSUserName()
```

### Returns

Returns a string value that contains the DM USER\_ID.

### Related Items

See the [PCDLogin](#) object.

See the following methods:

[GetAliasList](#)

[GetLoginLibrary](#)

[GetPrimaryGroup](#)

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# GetDST

## GetDST

Use this method to get the document security token (DST) that the server returns after a call to the [Execute](#) method of the [PCDLogin](#) object. You need to save the returned DST to use it in subsequent transactions.

### Syntax

```
PCDLogi n. GetDST()
```

### Returns

Returns a string value that contains the DST.

### Usage

The DST is a unique string returned from the DM Server and is valid while the user is logged on to the server. All subsequent transactions, such as search requests and lookups, require the DST to verify that the user is authorized to perform the transaction.

### Example

The sample code in the [Providing Library Access](#) section in Chapter 1 illustrates how you can use the GetDST method in your custom applications.

### Related Items

See the [PCDLogin](#) object.

See the following methods:

[AddLogin](#)

[Execute](#)

[SetDST](#)

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# GetFailedLoginList

## GetFailedLoginList

Use this method to get a [PCDNetAliasList](#) object pointer to an object containing a list of the supplied network aliases that did not pass network-level validation. This is only useful after [Execute](#) method of the [PCDLogin](#) object has been called.

### Syntax

```
PCDLogi n. GetFai l edLogi nLi st()
```

### Returns

The [GetFailedLoginList](#) method returns a variant object that contains information about failed logon attempts. This allows you to access [UnitName](#), [UnitType](#), and [UserName](#) methods that are supported by the [PCDNetAliasList](#) object.

### Usage

The [GetFailedLoginList](#) is useful when there are a number of libraries or other network resources that cannot be validated.

### Example

The sample code in the [Providing Library Access](#) section in Chapter 1 illustrates how you can use the [GetFailedLoginList](#) method in your custom applications.

### Related Items

See the following objects:

[PCDLogin](#)  
[PCDNetAliasList](#)

See the following methods:

[GetAliasList](#)  
[GetDOCSUserName](#)  
[GetLoginLibrary](#)  
[GetPrimaryGroup](#)

## **GetFailedLoginList**

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# GetGroupList

## GetGroupList

The GetGroupList method populates the result set with a list of the various groups that exist within a particular domain. There are two types of groups: distribution and security. Each is an instance of the Group object, and each contains a collection of attributes that describe group properties and group members.

### Syntax

```
PCDNetworkInfo.GetGroupList( strNetwkOS, _  
                           strDomainName )
```

### Parameters

strNetwkOS	A string variable that identifies the network operating system that is currently running. Valid values are: <ul style="list-style-type: none"><li>• %NI_ADS (Active Directory Services)</li><li>• %NI_NDS (a Novell network)</li><li>• %NI_NT (a Microsoft network)</li></ul>
strRootDomain	A string that identifies the root domain.

### Returns

GetGroupList returns an HRESULT value that indicates either success or the numeric error code that identifies the reason why the method did not execute successfully. This method also populates the result set with group values.

### Example

See the [Example](#) in the discussion of the [PCDNetworkInfo](#) object for sample code that illustrates how you can use the GetGroupList method in your custom applications.

### Related Items

See the [PCDNetworkInfo](#) object.

See the following properties:

## **GetGroupList**

[ErrDescription](#)  
[ErrNumber](#)

# GetGroupMembers

## GetGroupMembers

This method retrieves the user IDs that are present in the specified group. There is a hierarchy running from Domain to Group.

### Syntax

```
PCDNetworkInfo. GetGroupMembers( strNetworkOS, _  
                                strDomainName, _  
                                strGroupName )
```

### Parameters

strNetworkOS	A string variable that identifies the network operating system that is currently running. Valid values are: <ul style="list-style-type: none"><li>• %NI_ADS (Active Directory Services)</li><li>• %NI_NT (a Microsoft network)</li><li>• %NI_NDS (a Novell network)</li></ul>
strDomainName	A string that identifies the domain that contains the group.
strGroupName	A string that contains the name of the group from which members are to be retrieved.

### Returns

GetGroupMembers returns an **HRESULT** value that indicates either success (zero) or the numeric error code that identifies the reason why the method did not execute successfully. This method also populates the result set with the user IDs of the group members. No results are returned if either the Domain name or the Group name is blank.

### Example

See the [Example](#) in the discussion of the [PCDNetworkInfo](#) object for sample code that illustrates how you can use the GetGroupMembers method in your custom applications.

### Related Items

See the [PCDNetworkInfo](#) object.

## **GetGroupMembers**

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# GetLoginLibrary

## GetLoginLibrary

Use this method to retrieve the name of the DM library where the user is currently logged on.

### Syntax

```
PCDLogi n. GetLogi nLi brary()
```

### Returns

Returns a string that contains the name of the DM library where the user is currently logged on.

### Example

The [Providing Library Access](#) section in Chapter 1 illustrates the use of the GetLoginLibrary method.

### Related Items

See the [PCDLogin](#) object.

See the following methods:

- [AddLogin](#)
- [GetAliasList](#)
- [GetDOCSUserName](#)
- [GetFailedLoginList](#)
- [GetPrimaryGroup](#)

See the following properties:

- [ErrDescription](#)
- [ErrNumber](#)

# GetMetaPropertyValue

## GetMetaPropertyValue

Use this method to get the value of a property from metadata that is returned when a search or lookup is executed.

### Syntax

```
PCDLookup. GetMetaPropertyValue( strPropertyName )
```

```
PCDRecentDoc. GetMetaPropertyValue( strPropertyName )
```

```
PCDSearch. GetMetaPropertyValue( strPropertyName )
```

### Parameter

strPropertyName	The name of the property to be returned in the results. It should be one of those described below.
-----------------	--

### Returns

GetMetaPropertyValue returns a VARIANT value that contains the value of the requested property.

### Usage

You specify the metadata property for which you want data returned. The available metadata properties are %PropertyName, %Title, %Visible, and %Data. However, their meaning varies based on whether the GetMetaPropertyValue method is being used by the PCDLookup object or a different object.

For [PCDLookup](#), each row in the metadata property list offers the following information:

- %PropertyName is the name of the property as it is shown on the base form. It can have the value “\_UNKNOWN\_” if there is no corresponding property on the base form.
- %Title is the title in the Lookup list box for this column. The %Title property often contains the prompt that describes the data when it is presented to the user. It can be blank.

# GetMetaPropertyValue

- `%Visible` is a flag indicating whether or not this Lookup list box column should be displayed to the user.
- `%Data` is the value of this column in the current row of the Lookup data (as opposed to the metadata). For example, `%PropertyName` could return “AUTHOR\_ID”, and `%Data` could return “J\_SMI TH”.

For [PCDRecentDoc](#) and [PCDSearch](#), each row in the metadata property list offers the following information:

- `%PropertyName` is the name of the property as shown on the base form.
- `%Title` is the prompt for that field as shown on the base form.
- `%Visible` is undefined.
- `%Data` has no meaning.

*Note:* For compatibility purposes, PCDLookup also supports these property names as an alternative to the preferred terms shown above: PROPNAME, TITLE, VISIBLE, and DATA.

## Example

The [Example](#) in the discussion of the [PCDLookup](#) object in Chapter 3 illustrates how you can use the GetMetaPropertyValue method in custom applications you create.

## Related Items

See the following objects:

[PCDLookup](#)  
[PCDRecentDoc](#)  
[PCDSearch](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# GetMetaRowsFound

## GetMetaRowsFound

Use this method to identify the total number of rows of metadata that are returned when your search or lookup executes.

### Syntax

PCDLookup. GetMetaRowsFound()

PCDRecentDoc. GetMetaRowsFound()

PCDSearch. GetMetaRowsFound()

### Returns

Returns a long integer that contains the number of rows of metadata in the result set that your search or lookup returned.

### Usage

You can use the row count value that GetMetaRowsFound returns in a “For 1 to Total Rows” loop to iterate through the metadata result set. While use of the number returned by GetMetaRowsFound is sometimes appropriate, there are performance considerations. Calling this method causes the server to wait until the search has completely finished retrieving all data from the database. The server will not return control to your application until all processing of the search or lookup is complete.

Those performance considerations suggest that it may be preferable to use an alternate method of iterating through the metadata result set. For example, instead of a “For” loop, you can use the [SetRow](#) method to position the pointer to row zero, and then use a “While YourObj. [NextRow](#)” loop to iterate through the metadata result set. Using this procedure, the server retrieves large metadata result sets in smaller “chunks” according to parameters set by the [SetChunkFactor](#) method.

### Example

The GetMetaRowsFound method is used in the [Example](#) in the [PCDLookup](#) discussion in Chapter 3.

# GetMetaRowsFound

## Related Items

See the following objects:

[PCDLookup](#)  
[PCDRecentDoc](#)  
[PCDSearch](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# **GetNextKey**

## **GetNextKey**

GetNextKey generates the next numeric value for a primary key column in the SQL database. This is true for all SQL database software that DM supports. However, the manner in which the GetNextKey method functions in an Oracle database environment differs slightly from how it functions in a Microsoft or a Sybase database environment. The Usage section will note these differences.

### **Syntax**

```
PCDSQL. GetNextKey( strTabl eName )
```

### **Parameter**

<code>strTabl eName</code>	A string variable that contains the name of the SQL database table for which the next numeric key value is requested. Passing no table name value results in the default primary key value being incremented.
----------------------------	---

### **Returns**

The GetNextKey method returns a long integer that contains the key value after the previous value for the specified (or default) key is incremented by 1.

### **Usage**

In order to understand how the GetNextKey input parameter is used, the API user must know how the GetNextKey method and the SQL database work together. This discussion summarizes the operational link as it applies to Microsoft and Sybase SQL database software. A note explains differences that affect how Oracle SQL database software operates.

The DOCS\_UNI QUE\_KEYS table in the SQL database stores the most recently generated key value for each unique key in the database. The DOCS\_UNI QUE\_KEYS table has only two columns: a TBNAME column that normally stores the name of the table for a given key, and a LASTKEY column that stores the most recently assigned key value.

## GetNextKey

Most of the tables in the DM SQL database have a SYSTEM\_ID column as their primary key. DM generates primary key values for new rows in those tables by calling the GetNextKey method with no input parameter. Primary key values that are generated with no input parameter receive their key value from the TBNAME column that has a value of SYSTEMKEY. Because so many tables obtain their primary key values from this pool of numbers, the generic name of SYSTEMKEY is used. There is no SYSTEMKEY table.

You are not required to store primary key values generated in this manner—with no input parameter—in a column named SYSTEM\_ID. For example, the LOOKUPS table uses the METHODNUM column as its primary key, but it is generated in the same manner as other tables that use the more common SYSTEM\_ID column as a primary key. You can do the same thing with tables that your custom applications require, whether you name your primary key column SYSTEM\_ID or something different.

Some DM SQL database tables maintain primary keys that do not generate key values by incrementing the SYSTEMKEY column value. One example of this occurs in the PROFILE table where the DOCNUMBER column is shown as an “alternate” primary key. DM uses this primary key to manage the Document File Store, where the actual document content resides. You will see this in the DOCS\_UNI\_QUE\_KEYS table with its TBNAME column set to “DOCSADM. PROFILE”.

As a developer using the DM API to create your customized applications or enhancements to the basic product, you have complete flexibility to manage primary keys for the tables you create. You can maintain separate key entries in the DOCS\_UNI\_QUE\_KEYS table for the tables you create, or you can retrieve key values from the SYSTEMKEY pool that DM supports. With a maximum key value in excess of 2 billion, neither the SYSTEMKEY nor other DOCS\_UNI\_QUE\_KEYS table entries are likely to exhaust the available pool of key values.

*Note:* Oracle supports system-generated sequences through use of a CREATE SEQUENCE SQL statement. DM uses this instead of the DOCS\_UNI\_QUE\_KEYS table to manage primary key values in Oracle databases. These columns are stored in the DUAL table. The previous discussion of primary key usage is all relevant to Oracle users, with the

# GetNextKey

following exceptions:

- The SYSTEMKEY column is maintained in Oracle as the SEQSYSTEMKEY sequence key.
- Other primary keys have the table name that was passed to the GetNextKey method, but with "SEQ" as a prefix. So, the NEEDS\_INDEXING primary key is stored in Oracle in the SEQNEEDS\_INDEXING sequence.

However, your custom applications access the GetNextKey method in the exact same manner regardless of the database software you are using. You never need to modify your custom applications to accommodate differences in SQL database software.

## Example

You can see how GetNextKey is used by referring to the [Example](#) in the discussion of the [PCDSQL](#) method.

## Related Items

See the [PCDSQL](#) object.

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# GetPrimaryGroup

## GetPrimaryGroup

Use this method to get the DM GROUP\_ID that identifies the user's primary group. You must make a call to [PCDLogin.Execute](#) before you can get the primary group.

### Syntax

```
PCDLogi n. GetPri maryGroup()
```

### Returns

Returns a string that contains the primary group ID.

### Usage

A user can be a member of many groups in a DM library; however, a user's primary group determines which library maintenance system parameter settings apply to that user.

### Example

The section titled [Providing Library Access](#) in Chapter 1 illustrates how to use this method.

### Related Items

See [PCDLogin](#) object.

See the following methods:

- [AddLogin](#)
- [GetAliasList](#)
- [GetDOCSUserName](#)
- [GetFailedLoginList](#)
- [GetLoginLibrary](#)

See the following properties:

- [ErrDescription](#)
- [ErrNumber](#)

# GetProperties

## GetProperties

Use this method to create a [PCDPropertyList](#) object and to copy the properties in the [PCDDocObject](#)'s internal list into it.

### Syntax

`PCDDocObject.GetProperties()`

`PCDPropertyLists.GetProperties()`

### Returns

Returns an output buffer to receive the [PCDPropertyList](#) object pointer. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Related Items

See the following objects:

[PCDDocObject](#)

[PCDPropertyLists](#)

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# GetProperty

## GetProperty

Use this method to get the value of a property. This value may have been previously set with the  [SetProperty](#) method, or it may have been returned in a call to the  [Fetch](#) method.

### Syntax

```
PCDDocObj ect. GetProperty( strPropName )
```

```
PCDPropertyLists. GetProperty( strPropName )
```

### Parameter

strPropName	The string (BSTR) variable that contains the name of the property for which GetProperty is to return the value.
-------------	---

### Returns

Returns a pointer to a VARI ANT structure where the requested value will be put. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value. This value is also available in the  [ErrNumber](#) property.

### Related Items

See the following objects:

[PCDDocObject](#)  
 [PCDPropertyLists](#)

See the following properties:

[ErrDescription](#)  
 [ErrNumber](#)

# **GetPropertyIndex**

## **GetPropertyIndex**

Use this method to get the property index of a named property. Check the [ErrNumber](#) property after calling this method to verify that the return value is valid.

### **Syntax**

```
PCDPropertyList. GetPropertyIndex( strPropertyName )
```

### **Parameter**

<b>strPropertyName</b>	A string value that contains the name of the property for which GetPropertyIndex is to return the index value.
------------------------	--

### **Returns**

Returns an integer that identifies the location of the specified property in the index array. This index is zero-based.

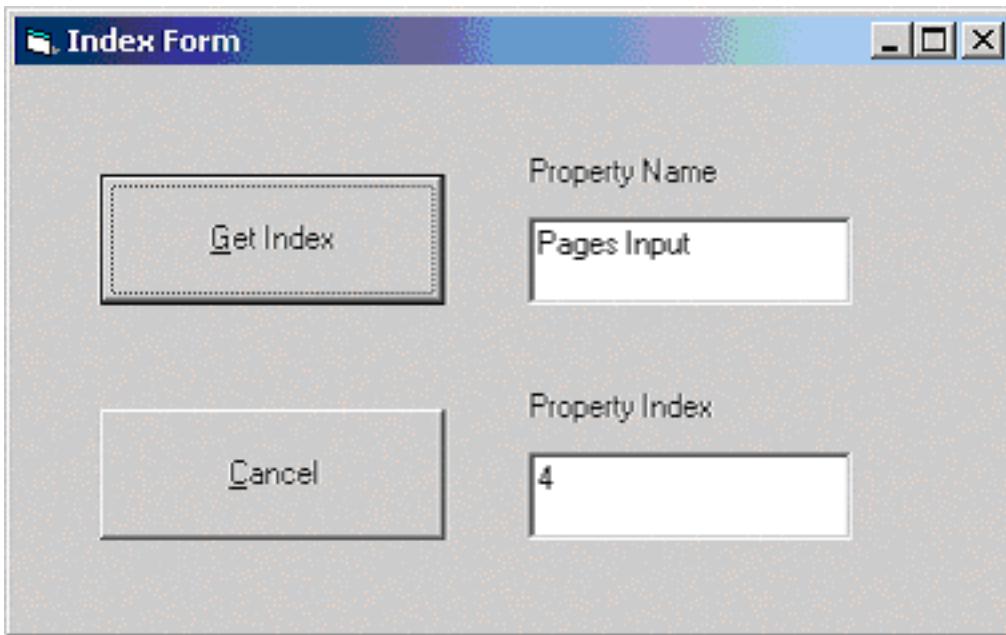
### **Usage**

If the requested property is not present in the property index array, the return value will be set to a value of negative one (-1). This will set an error in [PCDError](#), while still returning S\_OK.

# GetPropertyIndex

## Example

This sample code demonstrates the GetPropertyIndex call on the [PCDPropertyList](#) object. The form accepts the property name as the input value, and it returns the zero-based index of that property.



```
Private Sub cbGetIndex_Click()
    Dim nResult As Long
    Dim nIndex As Long
    Dim nSize As Long
    Dim sPropertyName As String

    sPropertyName = txtPropertyName.Text

    If sPropertyName = "None" Then
        MsgBox "Please enter a property before " & _
               "trying to retrieve the index."
    Else
        nSize = nIndexFormProps.GetSize()
        If nSize > 0 Then
```

# GetPropertyIndex

```
nIndex = _
    nIndexFormProps.GetPropertyIndex( _
        sPropertyName )
If nIndex >= 0 And nIndex < nSize Then
    txtPropertyIndex.Text = nIndex
Else
    MsgBox "This is not a valid property " _
        & "or the property returned is " _
        & "out of range."
End If
Else
    MsgBox "The PropertyList is empty."
End If
End If

End Sub
```

## Related Items

See the [PCDPropertyList](#) object.

See the following methods:

[BeginIter](#)  
[DeleteProperty](#)  
[GetCurrentPropertyName](#)  
[GetCurrentPropertyValue](#)  
[NextTrustee](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# GetPropertyValue

## GetPropertyValue

Use this method to get the value of the named property from the current row in the results set.

### Syntax

```
PCDGetDoc. GetPropertyValue( strPropertyName )
PCDGetForm. GetPropertyValue( strPropertyName )
PCDGetStream. GetPropertyValue( strPropertyName )
PCDPropertyList. GetPropertyValue( strPropertyName )
PCDPropertyLists. GetPropertyValue( strPropertyName )
PCDPutDoc. GetPropertyValue( strPropertyName )
PCDPutStream. GetPropertyValue( strPropertyName )
PCDRecentDoc. GetPropertyValue( strPropertyName )
PCDSearch. GetPropertyValue( strPropertyName )
```

### Parameter

strPropertyName	The name of the property for which GetPropertyValue is to retrieve the value.
-----------------	---

### Returns

Returns a VARIANT variable that contains the property value that you requested when you called GetPropertyValue.

### Usage

You should call [NextRow](#) or [SetRow](#) before you call this method. After making this call to get the property value, you should check the [ErrNumber](#) property. It should be zero (S\_OK) unless there was an error. Possible errors can include an invalid property name or not being set to a valid row.

### Example

The [Example](#) in the discussion of the [PCDGetDoc](#) object illustrates how you can use this method.

### Related Items

See the following objects:

# GetPropertyValue

[PCDGetDoc](#)  
[PCDGetForm](#)  
[PCDGetStream](#)  
[PCDPropertyList](#)  
[PCDPropertyLists](#)  
[PCDPutDoc](#)  
[PCDPutStream](#)  
[PCDRecentDoc](#)  
[PCDSearch](#)

See the following methods:

[NextRow](#)  
[SetRow](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# GetPropertyVal ueByIndex

## GetPropertyVal ueByIndex

Use this method to get the value of a return property based on its location in the return properties index.

### Syntax

```
PCDLookup. GetPropertyVal ueByIndex( intPropNdx )
PCDRecentDoc. GetPropertyVal ueByIndex( intPropNdx )
PCDSearch. GetPropertyVal ueByIndex( intPropNdx )
```

### Parameter

`intPropNdx`      The zero-based index of the return property whose value is desired.

### Returns

Returns a VARIANT that contains the property value data from the requested index position in the return array.

### Usage

The position of the requested property value is determined by when it was added to the return property data set by use of the [AddReturnProperty](#) method. For example, if the first call to [AddReturnProperty](#) is `AddReturnProperty("DocName")`, then, after calling [Execute](#) followed by [NextRow](#), a call to `GetPropertyVal ueByIndex( 0 )` would retrieve the value of the DocName property for the first row in the results set.

### Related Items

See the following objects:

[PCDLookup](#)  
[PCDRecentDoc](#)  
[PCDSearch](#)

See the following properties:

# GetPropertyValueByIndex

[ErrDescription](#)  
[ErrNumber](#)

# GetReturnProperties

## GetReturnProperties

Use this method to get a copy of the list of properties that the server will return when the search is executed.

### Syntax

```
PCDDocObj ect. GetReturnProperti es()  
PCDGetDoc. GetReturnProperti es()  
PCDPutDoc. GetReturnProperti es()  
PCDRecentDoc. GetReturnProperti es()
```

### Returns

Returns a pointer to a [PCDPropertyList](#) object.

### Usage

The `GetReturnProperties` method is used to retrieve data in the result set array that is returned by the [Fetch](#) method. You identify items that the `Fetch` returns by use of the [AddProperty](#) method prior to calling the `Fetch`. You can also use the [SetReturnProperties](#) method to set more than one return property with one method. You initialize the retrieval process by calling the [BeginIter](#) method. Use  [SetProperty](#) to set the pointer to the first array element. Use [NextProperty](#) to advance to the next array element. Monitor whether or not an error occurs to determine when all array elements have been retrieved.

### Related Items

See the following objects:

[PCDGetDoc](#)  
[PCDPropertyList](#)  
[PCDPutDoc](#)  
[PCDRecentDoc](#)

See the following methods:

[AddProperty](#)  
[BeginIter](#)

# **GetReturnProperties**

[NextProperty](#)  
[SetProperty](#)  
[SetReturnProperties](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# GetReturnProperty

## GetReturnProperty

Use this method to get the value of a return property.

### Syntax

```
PCDDocObj ect. GetReturnProperty( strPropertyName )
```

### Parameter

**strPropertyName** A string (BSTR) that contains the name of the property that the GetReturnProperty method is to return.

### Returns

Returns a pointer to a VARIANT structure where the value of the property is put. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value. This value is also available in the [ErrNumber](#) property.

### Example

The [Example](#) in the discussion of the [PCDPutDoc](#) object illustrates how you can use the GetReturnProperty method in your custom applications.

### Related Items

See the [PCDDocObject](#) object.

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# GetRowCount

## GetRowCount

Use the GetRowCount method to determine the number of rows that a result data set returns.

### Syntax

```
PCDNetworkInfo.GetRowCount()  
PCDSQL.GetRowCount()
```

### Returns

The GetRowCount method returns a long integer that contains the number of rows in the current result set.

### Example

The [Example](#) in the discussion of the [PCDNetworkInfo](#) object illustrates how you can use GetRowCount in your custom applications.

### Related Items

See the following objects:

[PCDNetworkInfo](#)  
[PCDSQL](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

## GetRowsAffected

## GetRowsAffected

This method reports the number of rows in the SQL database that were affected by the latest DELETE, INSERT, or UPDATE structured query language statement executed against the DM library.

### Syntax

```
PCDSQL. GetRowsAffected()
```

### Returns

The GetRowsAffected method returns a long integer that contains the number of rows affected by the most recent SQL statement that resulted in a DELETE, INSERT, or UPDATE operation.

### Example

The [Example](#) in [PCDSQL](#) illustrates how you can use GetRowsAffected in your custom applications.

### Related Items

See the [PCDSQL](#) object.

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# GetRowsFound

## GetRowsFound

Use this method to get the number of rows found as a result of a search operation.

### Syntax

```
PCDGetDoc. GetRowsFound()  
PCDLookup. GetRowsFound()  
PCDPutDoc. GetRowsFound()  
PCDRecentDoc. GetRowsFound()  
PCDSearch. GetRowsFound()
```

### Returns

Returns a long integer that contains the number of rows in the result set.

### Usage

You should exercise caution before calling this method because the server will not return the information until the search has completed processing and has retrieved all data from the database. Large data sets can subject the user to an unacceptably long wait. If this is a possibility, you can use SetRow and NextRow methods to begin iteration through a data set before all processing completes.

### Example

The [Example](#) in [PCDGetDoc](#) illustrates how you can use GetRowsFound in your custom applications.

### Related Items

See the following objects:

[PCDGetDoc](#)  
[PCDLookup](#)  
[PCDPutDoc](#)  
[PCDRecentDoc](#)

# GetRowsFound

[PCDSearch](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# GetSearchCriteria

## GetSearchCriteria

Use this method to get a [PCDPropertyList](#) object pointer to a copy of the list of search criteria that the search object is currently using.

### Syntax

```
PCDGetDoc. GetSearchCriteria()  
PCDLookup. GetSearchCriteria()  
PCDPutDoc. GetSearchCriteria()  
PCDRecentDoc. GetSearchCriteria()
```

### Returns

Returns a [PCDPropertyList](#) object pointer.

### Related Items

See the following objects:

[PCDGetDoc](#)  
[PCDLookup](#)  
[PCDPutDoc](#)  
[PCDRecentDoc](#)

See the [SetSearchCriteria](#) method.

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# GetSize

## GetSize

Use this method to get the number items in a list. For example, you can use GetSize to request the number of available libraries on the DM Server or the number of properties in a list of properties.

### Syntax

```
PCDGetLogi nLi bs. GetSi ze()  
PCDNetAl i asLi st. GetSi ze()  
PCDPropertyLi st. GetSi ze()  
PCDTrusteeLi st. GetSi ze()
```

### Returns

Returns a long integer to receive the number of items in the return data set list. Each of the items is a name-value pair that you can access by checking for the name and then retrieving the associated value. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Example

The [Example](#) in [PCDGetLoginLibs](#) illustrates how you can use GetSize in your custom applications.

### Related Items

See the following objects:

[PCDGetLoginLibs](#)  
[PCDNetAliasList](#)  
[PCDPropertyList](#)  
[PCDTrusteeList](#)

See the following methods:

[Execute](#)  
[GetAt](#)

## GetSize

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# GetSQLErrorCode

## GetSQLErrorCode

This method retrieves the native SQL error code as returned by the library's database.

### Syntax

```
PCDSQL. GetSQLErrorCode()
```

### Returns

This method returns a long integer that contains the error code number associated with the most recent SQL error. If zero is returned, no error occurred.

### Usage

The GetSQLErrorCode method differs from the [ErrNumber](#) method. If, for example, the Execute method returns an error, the ErrNumber property should be used to identify the error number. ErrNumber identifies a large number of error conditions, not all of which are related to a structured query language operation. The [ErrDescription](#) property returns the description for the most recent error. If an error occurred, but it was not a SQL error, then GetSQLErrorCode returns the value of zero.

If the ErrDescription or ErrNumber properties indicate that a SQL error occurred, then the GetSQLErrorCode method will return one of the following SQL-specific error codes:

Code	Description of SQL Error Code
0	No error
1	Missing DLL
2	Bad Connection
3	General Error
4	Fatal Error
5	Syntax Error
6	Invalid Logon

# GetSQLErrorCode

Code	Description of SQL Error Code
7	Server Not Available
8	Invalid Sybase Database
9	No Rows
10	Invalid Column Number
11	Invalid Row Number
12	Invalid Table Name
13	Invalid Column Name
14	Invalid Index Name
15	Duplicate Key in Index
16	Constraint Violation
17	Object Already Exists
18	Already Indexed
19	Not DBA
20	Stored Procedure Failed
21	No Memory
22	Cancelled
23	Permission Denied
24	Deadlocked

## Example

The [Example](#) in [PCDSQL](#) illustrates the use of the GetSQLErrorCode.

## Related Items

See the [PCDSQL](#) object.

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# GetTrustee

## GetTrustee

Use this method to get the trustee rights value for the trustee you specify.

### Syntax

```
PCDDocObject.GetTrustee( strTrusteeName, _  
    intTrusteeFlag )
```

### Parameters

strTrusteeName	String (BSTR) that identifies the name of the trustee.
intTrusteeFlag	A long integer that identifies the flag setting for the trustee. Supported values are as follows:  PCD_TRUSTEE_UNKNOWN_TYPE = 0 PCD_TRUSTEE_GROUP_TYPE = 1 PCD_TRUSTEE_PERSON_TYPE = 2

### Returns

Returns an integer that identifies the rights for the specified trustee. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Related Items

See the following objects:

[PCDDocObject](#)  
[PCDTrusteeList](#)

See the following methods:

[FetchTrustees](#)  
[GetTrustees](#)  
[SetTrustee](#)  
[SetTrustees](#)  
[UpdateTrustees](#)

## GetTrustee

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

## GetTrusteeIndex

# GetTrusteeIndex

Use this method to locate the index location of a trustee in a list of trustees.

### Syntax

```
PCDTrusteeList.GetTrusteeIndex( strTrusteeName,  
                               -  
                               intTrusteeFlags  
                               )
```

### Parameters

strTrusteeName	String (BSTR) that identifies the name of the trustee.
intTrusteeFlags	A long integer that identifies the flag setting for the trustee. Supported values are as follows:  PCD_TRUSTEE_UNKNOWN_TYPE = 0 PCD_TRUSTEE_GROUP_TYPE = 1 PCD_TRUSTEE_PERSON_TYPE = 2

### Returns

Returns an unsigned long integer that contains the offset in the list of trustees. JavaScript, Visual Basic, and VBScript return this as a function value. If the entry cannot be found, it returns a PCD\_ERR\_NAME\_NOT\_FOUND error condition.

### Related Items

See the [PCDTrusteeList](#) object.

See the following methods:

[GetSize](#)  
[GetTrusteeRights](#)  
[SetTrusteeRights](#)

See the following properties:

## GetTrusteeIndex

ErrDescription  
ErrNumber

## GetTrusteeRights

# GetTrusteeRights

Use this method to get the trustee rights for an entry in a trustee list at an index offset that you specify.

### Syntax

```
PCDTrusteeList.GetTrusteeRights( intNdx )
```

### Parameter

intNdx	An unsigned long integer that identifies the offset of the trustee entry you specify in a list of trustees.
--------	---

### Returns

Returns a an integer that contains the trustee rights. Languages such as JavaScript and Visual Basic return this as a function value.

### Related Items

See the [PCDTrusteeList](#) object.

See the following methods:

[GetSize](#)  
[GetTrusteeIndex](#)  
[SetTrusteeRights](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# GetTrustees

## GetTrustees

Once a list of trustees has been retrieved from the SQL database by use of [FetchTrustees](#), use this method to populate a [PCDTrusteeList](#) object so that you can iterate through the trustee entries.

### Syntax

```
PCDDocObj ect. GetTrustees()
```

### Returns

Returns a variant to the trustee list. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Related Items

See the following objects:

[PCDDocObject](#)  
[PCDTrusteeList](#)

See the following methods:

[FetchTrustees](#)  
[GetTrustee](#)  
[SetTrustee](#)  
[SetTrustees](#)  
[UpdateTrustees](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# GetUserFullName

## GetUserFullName

This method retrieves the full name of the user from the network operating system.

### Syntax

```
PCDNetworkInfo.GetUserFull Name( strNetworkType,  
—  
strDomainName, _  
strMemberID )
```

### Parameters

**strNetworkType** A string that identifies the type of network operating system that is currently running. Options are:

- %NI\_ADS  
(Active Directory Services)
- %NI\_NDS (a Novell network)
- %NI\_NT (a Microsoft network)

**strDomainName** A string that identifies the network domain that contains this user.

**strMemberID** A string that identifies the user within the network domain.

### Returns

GetUserFullName returns an HRESULT that indicates whether the method successfully retrieved the user's full name. S\_OK indicates success.

### Usage

If the GetUserFullName method was successful, you can retrieve the full name of the user by using the [GetValue](#) method.

### Example

The [Example](#) in the discussion of the PCDNetworkInfo object shows how you can use this method.

# GetUserFullName

## Related Items

See the [PCDNetworkInfo](#) object.

See the [GetValue](#) method.

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# GetUserGroups

## GetUserGroups

This method allows you to load a result set with the security groups that include the network user ID that you specify.

### Syntax

```
PCDNetworkInfo.GetUserGroups( strNetworkType, _  
                               strDomainName, _  
                               strUserID )
```

### Parameters

strNetworkType	A string that identifies the type of network operating system that is currently running. Options are: <ul style="list-style-type: none"><li>• %NI_ADS (Active Directory Services)</li><li>• %NI_NDS (a Novell network)</li><li>• %NI_NT (a Microsoft network)</li></ul>
strDomainName	A string that identifies the network domain that contains this user.
strUserID	A string that identifies the user within the network domain.

### Returns

The GetUserGroups method returns an HRESULT that indicates whether the method successfully retrieved the list of groups that include the user as a member. S\_OK indicates success.

### Usage

If the GetUserGroups method was successful, you can use the [GetRowCount](#) method to set up a loop. Then, use the [NextRow](#) method to iterate through the user group list. For each user group, use the [GetValue](#) method to retrieve the name of each group.

# GetUserGroups

## Example

The [Example](#) in the discussion of the PCDNetworkInfo object shows how you can use this method.

## Related Items

See the [PCDNetworkInfo](#) object.

See the following methods:

[GetRowCount](#)

[GetValue](#)

[NextRow](#)

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# GetUserList

## GetUserList

This method retrieves the set of user ID values that exist within the network domain that you specify.

### Syntax

```
PCDNetworkInfo.GetUserList( strNetworkType, _  
    strDomainName )
```

### Parameters

strNetworkType	A string that identifies the type of network operating system that is currently running. Options are: <ul style="list-style-type: none"><li>• %NI_ADS (Active Directory Services)</li><li>• %NI_NDS (a Novell network)</li><li>• %NI_NT (a Microsoft network)</li></ul>
strDomainName	A string that identifies the network domain from which a user list is to be retrieved.

### Returns

The GetUserList method returns an HRESULT that indicates whether the method successfully retrieved the list of users for the specified network domain. S\_OK indicates success.

### Usage

If the GetUserList method was successful, you can use the [GetRowCount](#) method to set up a loop. Then, use the [NextRow](#) method to iterate through the list of users. For each user, use the [GetValue](#) method to retrieve the user ID value.

You can query the root domain by using the “%UNDEFINED” token as the value for the name of the network domain. Doing this returns the user IDs on that specific computer, as opposed to all of the users in the network domain.

# GetUserList

## Example

The [Example](#) in the discussion of the PCDNetworkInfo object shows how you can use this method.

## Related Items

See the [PCDNetworkInfo](#) object.

See the following methods:

[GetRowCount](#)  
[GetValue](#)  
[NextRow](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# GetValue

## GetValue

The GetValue method retrieves the next value from the current result set.

### Syntax

```
PCDNetworkInfo.GetValue()
```

### Returns

GetValue returns a string that contains the value of the current element of the result set. It always returns a string, regardless of the actual data type.

### Usage

The [NextRow](#) method must be called before GetValue is called.

### Example

The [Example](#) in the discussion of the PCDNetworkInfo object shows how you can use this method.

### Related Items

See the [PCDNetworkInfo](#) object.

See the [NextRow](#) method.

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# **GrantRight**

## **GrantRight**

Use this method to set the named bit in the rights mask for the specified user or group.

### **Syntax**

```
PCDDocObj ect. GrantRi ght( strRi ghtName, _  
    i ntRi ghtsIn )
```

### **Parameters**

**strRi ghtName**      The name of the right to grant.

**i ntRi ghtsIn**      The rights mask in which you want to grant the named right.

### **Returns**

Returns an integer that contains the rights mask. This information is stored in the ACCESSRIGHTS column in the SECURITY table of the SQL database that supports your DM system.

### **Usage**

The Profile and QuickSearch rights are the lower 16 bits of a 32-bit integer. Bit settings are OR'ed together. For example, a user with rights to view profiles (binary 0000000000000001) and edit profiles (binary 0000000000000010) would have a rights setting of 3 (binary 0000000000000011).

The following Profile rights are supported:

<b>Value</b>	<b>Token</b>	<b>Description</b>
1	%PR_VIEW	View Profile
2	%PR_EDIT	Edit Profile
4	%PR_CONTENT_VIEW	View Document Content
8	%PR_CONTENT_RETRIEVE	Retrieve Document Content
16	%PR_CONTENT_EDIT	Edit Document Content
32	%PRCONTENT_COPY	Copy Document Content

# GrantRight

Value	Token	Description
64	%PR_DELETE	Delete Document
128	%PR_ACCESS_CONTROL	Control Access to Document
256	%RI GHT8	Assign to File
512	%RI GHT9	View Only Published

The following QuickSearch rights are supported:

Value	Token	Description
1	%QS_VI EW	View Search
2	%QS_EDI T	Edit Search
4	%QS_DELETE	Delete Search

## Related Items

See the [PCDDocObject](#) object.

See the following methods:

[HasRight](#)  
[RevokeRight](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# HasRight

## HasRight

Use this method to determine if the specified user or group rights mask permits the right you specify.

### Syntax

```
PCDDocObj ect. HasRi ght( strRi ghtName, i ntRi ghtsIn  
)
```

### Parameters

`strRi ghtName` A string value that identifies the name of the right.

`i ntRi ghtsIn` The rights mask in which you want to identify whether the right is available to the user.

### Returns

Returns TRUE if the named right is available to the user.

### Usage

It is important that you use late binding when creating a [PCDDocObject](#) object that will use the `HasRi ght` method. Using early binding to create the `PCDDocObject` can result in the `HasRi ght` method generating an error.

The Profile and QuickSearch rights are the lower 16 bits of a 32-bit integer. Bit settings are OR'ed together. For example, a user with rights to view profiles (binary 0000000000000001) and edit profiles (binary 0000000000000010) would have a rights setting of 3 (binary 0000000000000011).

The following Profile rights are supported:

Value	Token	Description
1	%PR_VI EW	View Profile
2	%PR_EDI T	Edit Profile
4	%PR_CONTENT_VI EW	View Document Content
8	%PR_CONTENT_RETRIEVE	Retrieve Document Content

# HasRight

Value	Token	Description
16	%PR_CONTENT_EDIT	Edit Document Content
32	%PRCONTENT_COPY	Copy Document Content
64	%PR_DELETE	Delete Document
128	%PR_ACCESS_CONTROL	Control Access to Document
256	%RI_GHT8	Assign to File
512	%RI_GHT9	View Only Published

The following QuickSearch rights are supported:

Value	Token	Description
1	%QS_VIEW	Use Search
2	%QS_EDIT	Edit Search
4	%QS_DELETE	Delete Search

## Example

The [Example](#) in the discussion of the [FetchTrustees](#) method shows how you can use the HasRight method.

## Related Items

See the [PCDDocObject](#) object.

See the following methods:

[FetchTrustees](#)  
[GrantRight](#)  
[RevokeRight](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# IsEmpty

## IsEmpty

Use this method to check whether an empty document was uploaded.

### Syntax

`PCDASPFileUpload.IsEmpty()`

### Returns

`IsEmpty` returns a Boolean value. TRUE indicates that an empty document was uploaded.

### Usage

In Netscape and Internet Explorer, users can erroneously attempt to upload a non-existent file by entering a file name that does not exist on their system. The `IsEmpty` method provides you with a way to test for this before the system tries to create the document in a DM file store.

### Related Items

See the [PCDASPFileUpload](#) object.

## **IsMemberOf**

### **IsMemberOf**

This method reports whether or not a user ID is a member of the network group that you specify.

#### **Syntax**

```
PCDNetworkInfo. IsMemberOf( strNetworkOS, _  
                           strDomainName, _  
                           strUserID, _  
                           strGroupName )
```

#### **Parameters**

<b>strNetworkOS</b>	A string variable that identifies the network operating system that is currently running. Valid values are:
	<ul style="list-style-type: none"><li>• %NI_ADS (Active Directory Services)</li><li>• %NI_NT (a Microsoft network)</li><li>• %NI_NDS (a Novell network)</li></ul>
<b>strDomainName</b>	A string that identifies the domain that is being searched.
<b>strUserID</b>	A string that identifies the network ID of the user whose membership in the group is being checked.
<b>strGroupName</b>	A string that identifies the name of the group that is being checked to see if the current user is a member.

#### **Returns**

The IsMemberOf method returns an HRESULT value that indicates either success or the numeric error code that identifies the reason why the method did not execute successfully. S\_OK indicates success.

#### **Usage**

Use the NextRow method to test whether the user is a member of the specified group. If the user is not a member of the specified group, then

# IsMemberOf

a call to NextRow returns FALSE. No results are returned if either the Domain name or the Group name is blank.

## Example

See the [Example](#) in the discussion of the [PCDNetworkInfo](#) object for sample code that illustrates how you can use the IsMemberOf method in your custom applications.

## Related Items

See the [PCDNetworkInfo](#) object.

See the [NextRow](#) method.

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# NewEnum

## NewEnum

This method provides standard-style C++ COM Enum access to the object.

### Syntax

```
PCDPropertyLists. NewEnum()
```

### Returns

This method returns a pointer to a [PCDEnumPropertyLists](#) object.

### Related Items

See the [PCDEnumPropertyLists](#) object.

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

## Next

## Next

This method returns [PCDPropertyList](#) items from the [PCDPropertyLists](#) object. You specify the number of items that you want returned.

### Syntax

```
PCDEnumPropertyLists.Next( LongNmbr )
```

### Parameters

LongNmbr	A long integer that identifies the number of <a href="#">PCDPropertyList</a> items that you want returned to you.
----------	---

### Returns

The Next method returns two parameters: an array of [PCDPropertyList](#) objects and a long integer that indicates that number of PCDPropertyList objects that were returned by the call.

### Related Items

See the [PCDEnumPropertyLists](#) object.

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# NextMetaRow

## NextMetaRow

Use this method to increment the current pointer in the metadata results set.

### Syntax

```
PCDLookup. NextMetaRow()  
PCDRecentDoc. NextMetaRow()  
PCDSearch. NextMetaRow()
```

### Returns

Returns a Boolean value of TRUE, unless there are no more rows, in which case it returns FALSE.

### Usage

This is useful only after calling [Execute](#). Call this or [SetMetaRow](#) after calling [Execute](#), but before the first call to [GetMetaPropertyValue](#).

### Related Items

See the following objects:

[PCDLookup](#)  
[PCDRecentDoc](#)  
[PCDSearch](#)

See the following methods:

[Execute](#)  
[GetMetaPropertyValue](#)  
[SetMetaRow](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

## **NextProperty**

### **NextProperty**

Use this method to increment the internal pointer for the current property list so it points to the next property in the list.

#### **Syntax**

`PCDPropertyList. NextProperty()`

`PCDPropertyLists. NextProperty()`

#### **Returns**

Returns an HRESULT return value to indicate the status of the method. If the method increments past the end of the list, the return value will be `PCD_S_END_OF_LIST`. Otherwise, it returns `S_OK`.

#### **Related Items**

See the following objects:

[PCDPropertyList](#)

[PCDPropertyLists](#)

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# NextRow

## NextRow

This method increments the current row pointer.

### Syntax

```
PCDGetDoc. NextRow()  
PCDLookup. NextRow()  
PCDNetworkInfo. NextRow()  
PCDPropertyLists. NextRow()  
PCDPutDoc. NextRow()  
PCDRecentDoc. NextRow()  
PCDSearch. NextRow()  
PCDSQL. NextRow()
```

### Returns

Returns a Boolean value of TRUE if pointer incremented without error. It returns FALSE if the row number is less than 1 or greater than the number of rows in the result set.

### Usage

You can make a call to NextRow or SetRow only after calling [Execute](#). You must make a call to NextRow or SetRow before the first call to [GetPropertyValue](#), [GetMetaPropertyValue](#), or [GetValue](#).

### Example

The section titled [Retrieving Recently Edited Documents](#) in Chapter 1 illustrates the use of the NextRow method.

### Related Items

See the following objects:

[PCDGetDoc](#)  
[PCDLookup](#)  
[PCDNetworkInfo](#)  
[PCDPropertyLists](#)

## **NextRow**

[PCDPutDoc](#)  
[PCDRecentDoc](#)  
[PCDSearch](#)  
[PCDSQL](#)

See the following methods:

[Execute](#)  
[GetMetaPropertyValue](#)  
[GetMetaRowsFound](#)  
[GetPropertyValue](#)  
[GetRowsFound](#)  
[GetValue](#)  
[SetRow](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# NextTrustee

## NextTrustee

Use this method to iterate through the trustees in the trustee list.

### Syntax

```
PCDTrusteeList.NextTrustee()
```

### Returns

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Usage

Checking the [ErrNumber](#) property allows you to determine whether or not the pointer was able to iterate to the next trustee. ErrNumber returns S\_OK if it successfully positioned on a list entry. It returns PCD\_S\_END\_OF\_LIST if it is at the end of the list.

### Related Items

See the [PCDTrusteeList](#) object.

See the following methods:

[BeginIter](#)  
[GetCurrentTrusteeFlags](#)  
[GetCurrentTrusteeName](#)  
[GetCurrentTrusteeRights](#)  
[GetTrusteeIndex](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

## OnEndPage

## OnEndPage

The Active Server Pages (ASP) engine calls this method to clean up ASP objects.

### Syntax

PCDASPFileUpload. OnEndPage()

*Caution:* While the OnEndPage method is visible to you in the DM API, you should never call it in your custom applications. Only the ASP engine should call this method.

### Related Items

See the [PCDASPFileUpload](#) object.

See the [OnStartPage](#) method.

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# OnStartPage

## OnStartPage

The Active Server Pages (ASP) engine calls this method to initialize pointers to ASP objects that the [Execute](#) method uses.

### Syntax

```
PCDASPFInterfaceUpload.OnStartPage( intUnk )
```

*Caution:* While the OnStartPage method is visible to you in the DM API, you should never call it in your custom applications. Only the ASP engine should call this method.

### Parameter

intUnk	An integer pointer that is passed in from ASP.
--------	--

### Related Items

See the [PCDASPFileUpload](#) object.

See the [OnEndPage](#) method.

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# Read

## Read

Use this method to read binary data from a physical file.

### Syntax

```
PCDGetStream. Read( LongBytes, [ LongBytesRead ] )
```

### Parameters

**LongBytes** An unsigned long integer that contains the number of bytes to attempt to read.

**LongBytesRead** An unsigned long set to the number of bytes read during the call. This parameter is optional.

### Returns

Returns a VARIANT containing a SAFEARRAY of bytes that is the actual data read. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Usage

Some programming languages, such as C++ and Visual Basic, support the optional second parameter for this method. In programming languages that do not support it, you can follow the Read method with the [BytesRead](#) method to obtain the number of bytes read.

### Related Items

See the [PCDGetStream](#) object.

See the following methods:

[GetPropertyValue](#)  
[Seek](#)

See the following properties:

[BytesRead](#)  
[ErrDescription](#)  
[ErrNumber](#)

# ReleaseResults

## ReleaseResults

Use this method to release the results that are currently held in memory as a result of the most recently generated SQL results set. Memory and system resources used to store the SQL result set are freed for later use.

### Syntax

```
PCDLookup. Rel easeResul ts()  
PCDRecentDoc. Rel easeResul ts()  
PCDSearch. Rel easeResul ts()  
PCDSQL. Rel easeResul ts()
```

### Returns

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Related Items

See the following objects:

[PCDLookup](#)  
[PCDRecentDoc](#)  
[PCDSearch](#)  
[PCDSQL](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# Reset

## Reset

This method resets the [PCDEnumPropertyLists](#) object's pointer to the first entry in the list of [PCDPropertyList](#) objects in the [PCDPropertyLists](#) collection.

### Syntax

```
PCDEnumPropertyLists.Reset
```

### Returns

This method returns an HRESULT that indicates whether the method successfully reset the enumeration list pointer. S\_OK indicates success.

### Related Items

See the following objects:

[PCDEnumPropertyLists](#)  
[PCDPropertyList](#)  
[PCDPropertyLists](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# RevokeRight

## RevokeRight

Use this method to turn off a named bit in the supplied rights mask.

### Syntax

```
PCDDocObj ect. RevokeRi ght( strRi ghtName, _  
intRi ghtsIn )
```

### Parameters

**strRi ghtName** The name of the right to turn off.

**intRi ghtsIn** The rights mask in which you want to turn off the named right.

### Usage

The Profile and QuickSearch rights are the lower 16 bits of a 32-bit integer. Bit settings are OR'ed together. For example, a user with rights to view profiles (binary 0000000000000001) and edit profiles (binary 0000000000000010) would have a rights setting of 3 (binary 0000000000000011). Sending an integer value of 2 would remove the user's rights to edit profile forms.

The following Profile rights are supported:

Value	Token	Description
1	%PR_VI EW	View Profile
2	%PR_EDI T	Edit Profile
4	%PR_CONTENT_VI EW	View Document Content
8	%PR_CONTENT_RETRI EVE	Retrieve Document Content
16	%PR_CONTENT_EDI T	Edit Document Content
32	%PRCONTENT_COPY	Copy Document Content
64	%PR_DELETE	Delete Document
128	%PR_ACCESS_CONTROL	Control Access to Document
256	%RI GHT8	Assign to File
512	%RI GHT9	View Only Published

# RevokeRight

The following QuickSearch rights are supported:

Value	Token	Description
1	%QS_VI EW	View Search
2	%QS_EDIT	Edit Search
4	%QS_DELETE	Delete Search

## Returns

Returns an integer that contains the rights mask as it exists after the specified access right has been revoked. This information is stored in the ACCESSRIGHTS column in the SECURITY table of the SQL database that supports your DM system.

## Related Items

See the [PCDDocObject](#) object.

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# Seek

## Seek

Use this method to position the file's current position pointer to a specific byte offset within the physical file.

### Syntax

```
PCDGetStream. Seek( LongSeekOffset )
```

### Parameter

LongSeekOffset    An unsigned long integer that specifies the offset from the beginning of the file object where you want the pointer to be positioned.

### Returns

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Related Items

See the [PCDGetStream](#) object.

See the following methods:

[GetPropertyValue](#)  
[Read](#)

See the following properties:

[BytesRead](#)  
[ErrDescription](#)  
[ErrNumber](#)

# **SetChunkFactor**

## **SetChunkFactor**

This method sets the number of rows to retrieve from the server results set when a SQL call is made. The default is 10.

### **Syntax**

```
PCDLookup. SetChunkFactor( LongChunkSize )  
PCDPropertyLists. SetChunkFactor( LongChunkSize )  
PCDRecentDoc. SetChunkFactor( LongChunkSize )  
PCDSearch. SetChunkFactor( LongChunkSize )
```

### **Parameter**

`LongChunkSize` A long integer that specifies the number of rows to retrieve into the local cache at one time.

### **Returns**

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### **Usage**

You are never required to call this method—it is provided to optimize wire traffic to the server.

### **Related Items**

See the following objects:

[PCDLookup](#)  
[PCDPropertyLists](#)  
[PCDRecentDoc](#)  
[PCDSearch](#)

See the following properties:

# **SetChunkFactor**

[ErrDescription](#)  
[ErrNumber](#)

## **SetComplete**

## **SetComplete**

For languages such as JavaScript and VBScript that do not give the user explicit control over when an interface is released, use this call to release all locks and other resources used by the stream. The stream is no longer available for use after this call.

*Note:* If you have control over the release of interfaces, you should release the interface rather than issuing this call to release the interface.

### **Syntax**

[PCDGetStream](#). SetComplete()

[PCDPutStream](#). SetComplete()

### **Returns**

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### **Related Items**

See the following objects:

[PCDGetStream](#)

[PCDPutStream](#)

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# SetDST

## SetDST

Use this method to set the document security token (DST) that will be used in processing the search request or manipulating document objects.

### Syntax

```
PCDDocObj ect. SetDST( strDST )  
PCDGetDoc. SetDST( strDST )  
PCDGetForm. SetDST( strDST )  
PCDLogin. SetDST( strDST )  
PCDLookup. SetDST( strDST )  
PCDNetworkInfo. SetDST( strDST )  
PCDPropertyLists. SetDST( strDST )  
PCDPutDoc. SetDST( strDST )  
PCDRecentDoc. SetDST( strDST )  
PCDSearch. SetDST( strDST )  
PCDSQL. SetDST( strDST )
```

### Parameter

strDST            A string that contains the security token.

### Returns

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Usage

The [GetDST](#) method that is called by the [PCDLogin](#) object provides the DST that you use during your entire user session. By using the SetDST method as part of each document or database retrieval, you allow the DM library to authenticate who you are and what access privileges have

# **SetDST**

been granted to you. The DST can be used across multiple DM libraries.

## **Example**

The SetDST method is used in many examples throughout this guide. The [Fetching a DM Document Object](#) discussion in Chapter 1 provides a representative reference that shows how you can use this method.

## **Related Items**

See the following objects:

[PCDDocObject](#)  
[PCDGetDoc](#)  
[PCDGetForm](#)  
[PCDLogin](#)  
[PCDLookup](#)  
[PCDNetworkInfo](#)  
[PCDPropertyLists](#)  
[PCDPutDoc](#)  
[PCDRecentDoc](#)  
[PCDSearch](#)  
[PCDSQL](#)

See the [GetDST](#) method.

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# **SetLibrary**

## **SetLibrary**

This method sets the DM library that is to be used with the current instance of the [PCDSQL](#) object.

### **Syntax**

```
PCDSQL. SetLibrary( strMyLibrary )
```

### **Parameter**

**strMyLibrary**      A string variable that contains the name of the library to use.

### **Returns**

Returns an HRESULT value. It always returns a value that indicates SUCCESS, whether the library that is being set exists or not.

### **Usage**

If an empty string is sent with the SetLibrary method, the user's default library becomes the target of subsequent SQL statements.

### **Example**

The [Example](#) in the discussion of the [PCDSQL](#) object shows how you can use this method.

### **Related Items**

See the [PCDSQL](#) object.

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# **SetLookupId**

## **SetLookupId**

Use this method to set the lookup form that you want to process. Similar to searches, lookups are controlled by forms.

### **Syntax**

```
PCDLookup.SetLookupId( strLookupName )
```

### **Parameter**

**strLookupName**    The lookup form that you want to use. One common example is the PEOPLE form.

### **Returns**

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### **Example**

The [Example](#) included in the discussion of the [PCDLookup](#) object illustrates how you can use this method.

### **Related Items**

See the [PCDLookup](#) object.

See the following methods:

[AddSearchCriteria](#)  
[AddSearchLib](#)  
[Execute](#)  
[GetMetaPropertyValue](#)  
[GetMetaRowsFound](#)  
[GetRowsFound](#)  
[GetSearchCriteria](#)  
[NextMetaRow](#)

[NextRow](#)  
[SetChunkFactor](#)  
[SetDST](#)  
[SetMetaRow](#)  
[SetRow](#)  
[SetSearchCriteria](#)  
[SetSearchObject](#)  
[SetTargetProperty](#)

See the following properties:

# **SetLookupId**

[ErrDescription](#)  
[ErrNumber](#)

# **SetMaxRows**

## **SetMaxRows**

Use this method to set the maximum number of rows to be returned by your lookup or search.

### **Syntax**

```
PCDLookup. SetMaxRows( intMaxRows )  
PCDRecentDoc. SetMaxRows( intMaxRows )  
PCDSearch. SetMaxRows( intMaxRows )
```

### **Parameter**

`intMaxRows` An integer variable that contains the maximum number of rows.

### **Return Values**

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### **Usage**

If you intend to limit the number of rows returned by your search, you must use this method before you call the Execute method. SetMaxRows has no effect if called after the lookup or search has been executed. Passing a maximum row parameter that is set to zero or less than zero results in an unlimited number of rows being returned.

### **Related Items**

See the following objects:

[PCDLookup](#)  
[PCDRecentDoc](#)  
[PCDSearch](#)

See the following properties:

# **SetMaxRows**

[ErrDescription](#)  
[ErrNumber](#)

## **SetMetaRow**

## **SetMetaRow**

Use this method to set the pointer to the current metadata row in the results set to a specific row number.

### **Syntax**

```
PCDLookup. SetMetaRow( LongRowNbr )  
PCDRecentDoc. SetMetaRow( LongRowNbr )  
PCDSearch. SetMetaRow( LongRowNbr )
```

### **Parameter**

LongRowNbr	A long integer that indicates the row number where you want to position the result set pointer.
------------	---

### **Returns**

Returns a Boolean value. TRUE indicates the current row number has been set to the specified row number. FALSE indicates there was an error.

### **Usage**

The row count in the result set is 1-based. Thus, if you call SetMetaRow( 0 ), and then call [GetMetaPropertyValue](#), you will generate an error. Also, if you set it to a row number that is higher than the number of rows in the result set, you will get an error.

### **Example**

The [Example](#) in the discussion of [PCDLookup](#) illustrates how you can use this method.

### **Related Items**

See the following objects:

[PCDLookup](#)  
[PCDRecentDoc](#)  
[PCDSearch](#)

## **SetMetaRow**

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# **SetObjectType**

## **SetObjectType**

Use this method to set the object type. For [PCDDocObject](#) and [PCDGetForm](#) the object is a form available to the user.

### **Syntax**

```
PCDDocObj ect. SetObj ectType( strObj ectType )  
PCDGetForm. SetObj ectType( strObj ectType )  
PCDPropertyLi sts. SetObj ectType( strObj ectType )
```

### **Parameter**

`strObj ectType` A string (BSTR) variable that identifies the object type that the method is to use.

### **Returns**

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### **Usage**

This routine must be called before you call any of the following methods:

[Create](#)  
[Fetch](#)  
[Update](#)

### **Example**

The [Example](#) in the discussion of the [PCDGetStream](#) object illustrates how you can use SetObjectType in your custom applications.

### **Related Items**

See the following objects:

[PCDDocObject](#)  
[PCDGetForm](#)

# **SetObjectType**

[PCDPropertyLists](#)

See the following methods:

[Create](#)  
[Fetch](#)  
[Update](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# **SetOptions**

## **SetOptions**

This method allows you to set recursion and level options for items in property list collections.

### **Syntax**

```
PCDPropertyLists. SetOptions( LongoptionVal )
```

### **Parameters**

LongoptionVal	A long integer that contains the bit-map indicator for the option that you set.
No options set	= 0
Make the property list recursive	= 1
Identify the property's level	= 2
Both recursive and level settings	= 3

### **Returns**

The SetOptions method returns an HRESULT that indicates whether the option was set successfully. S\_OK indicates success.

### **Usage**

A [PCDPropertyList](#) object can contain almost any information contained in the SQL database about a document, record, folder, or other type of database item. Assume a PCDPropertyList object points to a folder that contains two document objects and one folder object, which also contains two documents. Enabling recursion results in actions that affect the primary object (in this example, the folder that contains everything else) propagating through to all the other objects. This allows a user to set everything in the folder “read only” by setting the first folder to “read only.” With the recursion setting enabled, the other folder and all four documents are also set to “read only.”

Setting the level indicator returns information about the level of each affected object. In the example above, the first folder is at level 0. The two documents it contains and the other folder are at level 1. The two documents in the level 1 folder are at level 2. If recursion is not enabled, only level 0 items will be affected by any Execute method.

# **SetOptions**

## **Related Items**

See the following objects:

[PCDPropertyList](#)

[PCDPropertyLists](#)

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# **SetProperties**

## **SetProperties**

Use this method to set an object with a collection of properties. Any previously set properties are deleted.

### **Syntax**

```
PCDDocObject. SetProperties( strPropList )  
PCDPropertyList. SetProperties( strPropList )  
PCDPropertyLists. SetProperties( strPropList )
```

### **Parameter**

**strPropList** A pointer to the object (often a [PCDPropertyList](#) object) from which the [PCDDocObject](#) should copy the new properties.

### **Returns**

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### **Related Items**

See the following objects:

[PCDDocObject](#)  
[PCDPropertyList](#)  
[PCDPropertyLists](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# **SetProperty**

## **SetProperty**

Use this method to set a property to be used with a [Create](#), [Execute](#), or [Update](#) method. If the property already exists in the property list, its value will be updated. If it does not exist, it will be added to the property list.

### **Syntax**

```
PCDDocObj ect. SetProperty( strPropName, vntVal )
```

```
PCDPropertyLists. SetProperty( strPropName,  
vntVal )
```

### **Parameters**

**strPropName** A string (BSTR) variable that contains the name of the property to be set.

**vntVal** A VARIANT that contains the value to which the property is to be set.

### **Returns**

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### **Example**

The [Example](#) in the discussion of the [PCDGetStream](#) object demonstrates how you can use the SetProperty method.

### **Related Items**

See the following objects:

[PCDDocObject](#)  
[PCDPropertyLists](#)

See the following properties:

# **SetProperty**

ErrDescription  
ErrNumber

# **SetReturnProperties**

## **SetReturnProperties**

Use this method to set [PCDRecentDoc](#) or [PCDSearch](#) objects with a collection of properties. Any property values that were previously set are deleted by the SetReturnProperties method.

### **Syntax**

```
PCDRecentDoc. SetReturnProperties( obj PropList )  
PCDSearch. SetReturnProperties( obj PropList )
```

### **Parameter**

**obj PropList**      A variable that points to a [PCDPropertyList](#) object that contains the return properties of the search.

### **Returns**

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### **Related Items**

See the following objects:

[PCDPropertyList](#)  
[PCDRecentDoc](#)  
[PCDSearch](#)

See the [GetReturnProperties](#) method.

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

## **SetRow**

## **SetRow**

Use this method to set the row pointer in the results set to a specific row.

### **Syntax**

```
PCDGetDoc. SetRow( LongRowNbr )  
PCDLookup. SetRow( LongRowNbr )  
PCDPutDoc. SetRow( LongRowNbr )  
PCDRecentDoc. SetRow( LongRowNbr )  
PCDSearch. SetRow( LongRowNbr )  
PCDSQL. SetRow( LongRowNbr )
```

### **Parameter**

LongRowNbr     A long integer that identifies the row number to which the row pointer is to be set.

### **Returns**

Returns a Boolean value that indicates whether or not the operation completed successfully. It is TRUE if the row number was set to a row within the current result set. A FALSE value indicates that the row number was less than 1 or greater than the number of rows in the result set.

### **Usage**

The row count in the result set is 1-based. You can use [GetColumnValue](#) or [GetPropertyValue](#) methods to retrieve data from the selected row in the result set.

### **Example**

The [Example](#) in the [PCDLookup](#) object discussion illustrates how you can use the SetRow method.

### **Related Items**

See the following objects:

# **SetRow**

[PCDGetDoc](#)  
[PCDLookup](#)  
[PCDPutDoc](#)  
[PCDRecentDoc](#)  
[PCDSearch](#)

See the following methods:

[GetColumnValue](#)  
[GetPropertyValue](#)  
[NextRow](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# **SetSearchCriteria**

## **SetSearchCriteria**

Use this method to set the search criteria to be used in the current search.

### **Syntax**

```
PCDGetDoc. SetSearchCriteria( obj PropList )  
PCDLookup. SetSearchCriteria( obj PropList )  
PCDPutDoc. SetSearchCriteria( obj PropList )  
PCDRecentDoc. SetSearchCriteria( obj PropList )  
PCDSearch. SetSearchCriteria( obj PropList )
```

### **Parameter**

**obj PropList** An object variable that points to a [PCDPropertyList](#) object.

### **Returns**

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### **Usage**

Instead of adding criteria one property pair at a time, this method lets you set a collection of search criteria using a [PCDPropertyList](#) object. Any previously existing search properties are deleted when the SetSearchCriteria method executes.

### **Related Items**

See the following objects:

[PCDGetDoc](#)  
[PCDLookup](#)  
[PCDPropertyList](#)  
[PCDPutDoc](#)  
[PCDRecentDoc](#)

# **SetSearchCriteria**

[PCDSearch](#)

See the following methods:

[AddSearchCriteria](#)  
[GetSearchCriteria](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

## **SetSearchObject**

## **SetSearchObject**

Use this method to identify the form that you want to use to process the specified operation.

### **Syntax**

```
PCDGetDoc. SetSearchObj ect( strObj Name )
PCDLookup. SetSearchObj ect( strObj Name )
PCDPutDoc. SetSearchObj ect( strObj Name )
PCDRecentDoc. SetSearchObj ect( strObj Name )
PCDSearch. SetSearchObj ect( strObj Name )
```

### **Parameter**

strObj Name	A string variable that identifies the form to be used in the operation.
-------------	---

### **Returns**

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### **Usage**

Use this method before calling [Execute](#). The FORM\_NAME column in the FORMS table contains the identifier that you use to identify the form that is to be used in an operation.

### **Example**

The [Example](#) in the discussion of [PCDLookup](#) shows how you can use this method in your custom applications.

### **Related Items**

See the following objects:

[PCDGetDoc](#)  
[PCDLookup](#)  
[PCDPutDoc](#)

## **SetSearchObject**

[PCDRecentDoc](#)  
[PCDSearch](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# **SetTargetProperty**

## **SetTargetProperty**

Use this method to set the target property of a lookup.

### **Syntax**

```
PCDLookup. SetTargetProperty( strTargetProp )
```

### **Parameter**

<b>strTargetProp</b>	A string variable that identifies the target property that is to be used for the lookup.
----------------------	--

### **Returns**

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### **Usage**

The target property can be any data object that appears on a lookup form. If your form has a data item called SUPERVISOR, it will usually be mapped to the lookup form as “SUPERVISOR”, and you select it as the target property by passing that identifier as the [SetTargetProperty](#) parameter.

*Note:* While the DM API allows you to change the name of fields on forms you create (or modify), you should only do this when absolutely necessary. You should never rename original fields on the standard forms that DM creates when you initially install it.

You can use the [AddUserFilterCriteria](#) method to further limit the search. If you want to retrieve documents written by Mary Smith, the AddUserFilterCriteria method allows you, for example, to restrict the search to author names that match “SMITH\_M”.

### **Related Items**

See the [PCDLookup](#) object.

See the [AddUserFilterCriteria](#) method.

See the following properties:

# **SetTargetProperty**

[ErrDescription](#)  
[ErrNumber](#)

## SetTrustee

## SetTrustee

Use this method to set a trustee value in a trustee list. If the trustee name and flags match an existing entry in the list, its rights will be updated. If it does not match an existing entry, the entry will be added.

### Syntax

```
PCDDocObj ect. SetTrustee( strTrusteeName, _  
    intTrusteeFlags, _  
    intTrusteeRights )
```

### Parameters

strTrusteeName	A string (BSTR) variable that identifies the name of the trustee.
intTrusteeFlags	An integer variable that sets the trustee flags. Supported values are as follows:  PCD_TRUSTEE_UNKNOWN_TYPE = 0 PCD_TRUSTEE_GROUP_TYPE = 1 PCD_TRUSTEE_PERSON_TYPE = 2
intTrusteeRights	An integer variable that identifies the rights that are to be assigned to the trustee.

### Returns

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Example

The [Example](#) in the discussion of [FetchTrustees](#) shows how you can use the SetTrustee method.

### Related Items

See the [PCDDocObject](#) object.

See the following methods:

[GetTrustee](#)  
[GetTrustees](#)

# **SetTrustee**

[SetTrustees](#)

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# **SetTrustees**

## **SetTrustees**

Use this method to copy all the trustee entries from a [PCDTrusteeList](#) object into the internal trustee list for the [PCDDocObject](#).

### **Syntax**

```
PCDDocObj ect. SetTrustees( obj TrusteeLi st )
```

### **Parameter**

**obj TrusteeLi st** A pointer to a [PCDTrusteeList](#) object from which the [PCDDocObject](#) can copy the trustee list.

### **Returns**

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### **Example**

The section that discusses [Getting and Updating Trustee Information](#) in chapter 1 provides an example of how you can use the SetTrustees method.

### **Related Items**

See the following objects:

[PCDDocObject](#)  
[PCDTrusteeList](#)

See the following methods:

[GetTrustee](#)  
[GetTrustees](#)  
[SetTrustee](#)

See the following properties:

# **SetTrustees**

[ErrDescription](#)  
[ErrNumber](#)

# **SetTrusteeRights**

## **SetTrusteeRights**

Use this method to update the Trustee rights for a trustee at a given offset in the trustee list.

### **Syntax**

```
PCDTrusteeList. SetTrusteeRights( LongNdx, _  
    intTrusteeRights )
```

### **Parameters**

LongNdx	An unsigned long integer that contains the offset in the trustee list where trustee rights are to be set.
intTrusteeRights	An integer that contains the trustee rights flags.

### **Returns**

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### **Usage**

The trustee rights flags parameter uses the same access rights settings as other security methods. These security settings are maintained in the ACCESSRIGHTS column of the SECURITY table in the SQL database.

### **Related Items**

See the [PCDTrusteeList](#) object.

See the [GetTrusteeRights](#) method.

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# Skip

## Skip

This method skips [PCDPropertyList](#) objects in the [PCDPropertyLists](#) collection. You specify how many objects are to be skipped.

### Syntax

```
PCDEnumPropertyLists.Skip( LongNmbr )
```

### Parameters

LongNmbr	A long integer that identifies the number of objects that the enumeration list pointer is to skip.
----------	--

### Returns

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Related Items

See the following objects:

[PCDEnumPropertyLists](#)  
[PCDPropertyList](#)  
[PCDPropertyLists](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# **UnitName**

## **UnitName**

Use this method to get unit name of an entry in a [PCDLogin](#) list that a [PCDNetAliasList](#) object is accessing.

### **Syntax**

```
PCDNetAl i asLi st. Uni tName( l ngNdx )
```

### **Parameter**

l ngNdx	A long integer that identifies the entry in the list.
---------	---

### **Returns**

Returns a string that identifies the unit name.

### **Usage**

The index list processed by this method is zero-based.

### **Example**

The discussion on [Providing Library Access](#) in chapter 1 shows how you can use this method in custom applications you create.

### **Related Items**

See the following objects:

[PCDLogin](#)  
[PCDNetAliasList](#)

See the following methods:

[UnitType](#)  
[UserName](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# **UnitType**

## **UnitType**

Use this method to get unit type of an entry in a [PCDLogin](#) list that a [PCDNetAliasList](#) object is accessing.

### **Syntax**

```
PCDNetAl i asLi st. Uni tType( l ongNdx )
```

### **Parameter**

l ongNdx	A long integer that identifies the entry in the list.
----------	---

### **Returns**

Returns an integer value that identifies the unit type. The type values are as follows:

Value	Unit Type
0	LIBRARY_LOGIN
1	NETWORK_BROWSER
2	NETWORK_NDS
8	MS_NETWORK

### **Usage**

The index list processed by this method is zero-based.

### **Example**

The discussion on [Providing Library Access](#) in chapter 1 shows how you can use this method in custom applications you create.

### **Related Items**

See the following objects:

[PCDLogin](#)  
[PCDNetAliasList](#)

See the following methods:

# **UnitType**

[UnitName](#)  
[UserName](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# Update

## Update

This method updates a [PCDDocObject](#) using the information that has previously been set by other methods.

### Syntax

```
PCDDocObj ect. Update()
```

### Returns

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Example

The discussion about [Getting and Updating Trustee Information](#) in chapter 1 shows how you can use the Update method.

### Related Items

See the [PCDDocObject](#) object.

See the following methods:

[Create](#)

[Delete](#)

[Fetch](#)

See the following properties:

[ErrDescription](#)

[ErrNumber](#)

# UpdateTrustees

## UpdateTrustees

Use this method to update trustee information for a [PCDDocObject](#) object.

### Syntax

```
PCDDocObj ect. UpdateTrustees()
```

### Returns

Returns an HRESULT to receive the result of the call. S\_OK indicates success. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Example

The `UpdateTrustees` method is discussed in the section titled [Getting and Updating Trustee Information](#) in chapter 1.

### Related Items

See the [PCDDocObject](#) object.

See the following methods:

- [FetchTrustees](#)
- [GetTrustee](#)
- [GetTrustees](#)
- [SetTrustee](#)
- [SetTrustees](#)

See the following properties:

- [ErrDescription](#)
- [ErrNumber](#)

# UserName

## UserName

Use this method to get user name of an entry in a [PCDLogin](#) list that a [PCDNetAliasList](#) object is accessing.

### Syntax

```
PCDNetAl i asLi st. UserName( l ngNdx )
```

### Parameter

l ngNdx	A long integer that identifies the entry in the list.
---------	---

### Returns

Returns a string value that contains a user name.

### Usage

The index list processed by this method is zero-based.

### Example

The discussion on [Providing Library Access](#) in chapter 1 shows how you can use this method in custom applications you create.

### Related Items

See the following objects:

[PCDLogin](#)  
[PCDNetAliasList](#)

See the following methods:

[UnitName](#)  
[UnitType](#)

See the following properties:

[ErrDescription](#)  
[ErrNumber](#)

# Write

## Write

Use this method to write binary data to a physical file.

### Syntax

```
PCDPutStream. Write( vntData, lngBytes )
```

### Parameters

vntData	A variant variable that contains the data to write out to the file. It should be a BSTR or a SAFEARRAY.
lngBytes	The total number of bytes of data that are written to the vntData variable.

### Returns

Returns the actual number of bytes written during the Write operation. Languages such as JavaScript, Visual Basic, and VBScript return this as a function value.

### Related Items

See the [PCDPutStream](#) object.

See the [GetPropertyValue](#) method.

See the following properties:

[BytesWritten](#)  
[ErrDescription](#)  
[ErrNumber](#)

# 5

## DM API Tokens

### In This Chapter

This chapter presents an alphabetical list of DM tokens. Each entry discusses the syntax, usage, and other information associated with each token.

## **%ADD\_ATTACHMENT**

## **%ADD\_ATTACHMENT**

This token is used with the `%VERSICONDIRCTIVE` token to add an attachment. See `%VERSICONDIRCTIVE` for further information.

### **Syntax**

```
PCDDocObj ect. SetProperty(" %VERSICONDIRCTIVE" ,  
" %ADD_ATTACHMENT" )
```

### **Parameters**

<code>%VERSICONDIRCTIVE</code>	The token that indicates that this command statement adjusts the document version settings.
<code>%ADD_ATTACHMENT</code>	The token identifier that indicates that an attachment is being added to the current document version.

### **Example**

See the `%VERSICONDIRCTIVE` example on page 500.

### **Related Items**

See the [PCDDocObj ect](#) object.

See the `SetProperty` method.

See the [%VERSICONDIRCTIVE](#) token.

## **%ATTACHMENT\_ID**

## **%ATTACHMENT\_ID**

This token is used by the `Update` method that `PCDDocObj ect` supports to add an attachment to the document.

### **Syntax**

```
PCDDocObj ect. SetProperty("%ATTACHMENT_ID", _  
    l ngAttachDocNum)
```

### **Parameters**

<code>%ATTACHMENT_ID</code>	The token identifier that indicates an attachment is being added to the current document.
<code>l ngAttachDocNum</code>	The value from the <code>DOCNUMBER</code> column of the <code>PROFILE</code> table that identifies the document that is being added as an attachment.

### **Usage**

Use `%ATTACHMENT_ID` with the `SetProperty` method that the `PCDDocObj ect` object supports.

### **Example**

```
pDocObj ect = CreateObj ect("PCDCI ent. PCDDocObj ect. 1")  
  
' Set the login security information (DST) here.  
  
pDocObj ect. SetProperty("%VERS ION_DIREC TIVE", "%ADD_ATTACHMENT")  
pDocObj ect. SetProperty("%ATTACHMENT_ID", atcl D)
```

### **Related Items**

See the `PCDDocObj ect` object.

See the `SetProperty` method.

See the `%VERS ION_DIREC TIVE` token.

## **%CHECKIN\_DATE**

This token is used in the `Update` method that the `PCDDocObj` object supports when the object is being checked out or locked.

### **Syntax**

```
PCDDocObj ect. SetProperty( _  
    "%CHECKIN_DATE", dteDueDate)
```

### **Parameters**

<code>%CHECKIN_DATE</code>	The token identifier that indicates check-in data is being set.
<code>dteDueDate</code>	A date variable that contains the check-in date value.

### **Example**

```
pDocObj ect = CreateObj ect("PCDCI i ent. PCDDocObj ect. 1")  
  
' Set the expected check-in date.  
expDate = "" + getFormVal ue( "checkoutdocact", "ExpectedDate" )  
If ( expDate. Length > 0 ) Then  
    pDocObj ect. SetProperty ( "%CHECKIN_DATE", expDate )  
End If
```

### **Related Items**

See the `PCDDocObj ect` object.

See the `SetProperty` method.

See the following tokens:

```
%CHECKOUT_COMMENT  
%ELAPSED_TIME
```

## **%CHECKOUT\_COMMENT**

## **%CHECKOUT\_COMMENT**

This token is used in the `Update` method that the `PCDDocObj ect` object supports to specify the checkout comment when an object is being checked out or locked.

### **Syntax**

```
PCDDocObj ect. SetProperty( _  
    "%CHECKOUT_COMMENT", strComment)
```

### **Parameters**

<code>%CHECKOUT_COMMENT</code>	The token identifier that indicates a check-out comment is being set.
<code>strComment</code>	A string variable that contains the check-out comment.

### **Example**

```
pDocObj ect = CreateObj ect("PCDCI i ent. PCDDocObj ect. 1")  
  
' Set the comment, if any.  
If ( comment.length > 0 ) Then  
    pDocObj ect. SetProperty( "%CHECKOUT_COMMENT", comment )  
End If
```

### **Related Items**

See the `PCDDocObj ect` object.

See the `SetProperty` method.

See the following tokens:

```
%CHECKIN_DATE  
%ELAPSED_TIME
```

## %CONTENT

## %CONTENT

This token is used to retrieve a stream for the content of the document. It can also be used to update the document.

### Syntax

PCDGetDoc. GetPropertyValue("%CONTENT")

PCDPutDoc. GetPropertyValue("%CONTENT")

### Parameters

%CONTENT

Used with the PCDGetDoc object, the %CONTENT token retrieves a pointer to a PCDGetStream object that allows the program to get the content of the document.

Used with the PDCPutDoc object, the %CONTENT token retrieves a pointer to the PDCPutStream object so the content of the document can be written.

### Example

```
' Create the object.  
pGetDoc = CreateObject("PCDClient.PCDGetDoc.1")  
  
' check for errors  
checkError(pGetDoc, "ERROR_CREATEGETDOC")  
  
' add library to the search criteria  
pGetDoc.AddSearchCriteria("%TARGET_LIBRARY", Library)  
  
' add docnum the search criteria  
pGetDoc.AddSearchCriteria("%DOCUMENT_NUMBER", docnum)  
  
' add version of the doc the search criteria  
pGetDoc.AddSearchCriteria("%VERSION_ID", version_id)  
  
' run the search  
pGetDoc.Execute()  
  
' check error  
checkError(pGetDoc, "ERROR_CHECKOUTFILESIZE")
```

```
' Report the total components (or rows).
iCount = pGetDoc.GetReturnValue(%NUM_COMPONENTS)
MsgBox("The total number of rows = " & CStr(iCount))

' Get the first row.
pGetDoc.SetRow(1)

' Get the stream.
pGetStream = pGetDoc.GetPropertyVal ue("%CONTENT")

checkError(pGetDoc, "ERROR_GETCONTENT")

' Get the size of the content stream.
VerFileSize = GetStream.GetPropertyVal ue( _
    "%I STREAM_STATSTG_CBSIZE_LOWPART")
```

## Related Items

See the following objects:

[PCDGetDoc](#)  
[PCDPutDoc](#)

See the [GetPropertyVal ue](#) method.

See the [%I STREAM\\_STATSTG\\_CBSIZE\\_LOWPART](#) token.

## **%CONTENTS\_AFTER\_ITEM**

## **%CONTENTS\_AFTER\_ITEM**

This token is used in conjunction with %CONTENTS\_DIRECTORY to specify that a folder should be moved after another folder.

### **Syntax**

```
PCDDocObject SetProperty( _  
    "%CONTENTS_AFTER_ITEM",  lngAfterItem)
```

### **Parameters**

%CONTENTS_AFTER_ITEM	The token identifier that indicates that the current item is to be positioned after another item.
lngAfterItem	The value from the DOCNUMBER column of the PROFILE table that identifies the item after which the new link is to be inserted.

### **Example**

```
' Create the object.  
pDocObject = CreateObject("PCDCIent.PCDDocObject")  
  
' Check for errors.  
checkError(pDocObject, "ERROR_CREATECSI OBJECT_PCDDOCOBJECT")  
  
' Set the DM security token.  
pDocObject SetProperty(strDST)  
  
' Set object type.  
pDocObject SetProperty("ContentItem")  
  
' Set the library.  
pDocObject SetProperty("%TARGET_LIBRARY", strFolderLib)  
  
' Set the item to be moved.  
pDocObject SetProperty("%CONTENTS_ITEM", systemID)  
  
' Set the item to be moved after.  
pDocObject SetProperty("%CONTENTS_AFTER_ITEM", afterSystemID)  
  
' Specify the move_after action.  
pDocObject SetProperty("%CONTENTS_DIRECTORY", _
```

## %CONTENTS\_AFTER\_ITEM

CONTENTS\_MOVE\_AFTER")

```
' Update the document.  
pDocObj ect. Update()  
  
' Check for errors.  
checkError(pDocObj ect, "ERROR_MOVEAFTERFOLDERITEM")  
  
' Destroy the object.  
dele te pDocObj ect
```

### Related Items

See the PCDPDocObj ect object.

See the SetProperty method.

See the %CONTENTS\_DIRECTORY token.

## **%CONTENTS\_COPY\_CONTENTS**

## **%CONTENTS\_COPY\_CONTENTS**

This token is used in conjunction with %CONTENTS\_DIRECTIVE to specify that a folder's content should be copied into another folder.

### **Syntax**

```
PCDDocObj ect. SetProperty( _  
    "%CONTENTS_DIRECTIVE", _  
    "%CONTENTS_COPY_CONTENTS")
```

### **Parameters**

%CONTENTS_DIRECTIVE	The token identifier that indicates that the program will manipulate the contents of a folder.
%CONTENTS_COPY_CONTENTS	The token identifier that indicates the contents of one folder are being copied to another folder.

### **Example**

```
' create the object.  
pDocObj ect = CreateObj ect("PCDCI ent. PCDDocObj ect")  
  
' set the DM security token.  
pDocObj ect. SetDST(strDST)  
  
' Set the object type.  
pDocObj ect. SetObj ectType("ContentsCollect ion")  
  
' Set the library.  
pDocObj ect. SetProperty("%TARGET_LIBRARY", Library)  
  
' Set the source.  
pDocObj ect. SetProperty("%CONTENTS_SRC_PARENT", srcFol derNum)  
  
' Set the source version.  
pDocObj ect. SetProperty("%CONTENTS_SRC_PARENT_VERSION", _  
    srcFol derVersi on)  
  
' Set the source library  
pDocObj ect. SetProperty("%CONTENTS_SRC_PARENT_LIBRARY", _  
    srcFol derLi bl D)
```

# %CONTENTS\_COPY\_CONTENTS

```
' Set the target folder.  
pDocObj ect. SetProperty("%CONTENTS_DST_PARENT", tarFol derNum)  
  
' Set the target folder version.  
pDocObj ect. SetProperty("%CONTENTS_DST_PARENT_VERSION", _  
tarFol derVersi on)  
  
' Set the target folder library.  
pDocObj ect. SetProperty("%CONTENTS_DST_PARENT_LIBRARY", _  
tarFol derLi bl D)  
  
' Command to copy.  
pDocObj ect. SetProperty("%CONTENTS_DIRECTIVE", _  
"%CONTENTS_COPY_CONTENTS")  
  
' Perform the update.  
pDocObj ect. Update()  
  
' Check the error.  
checkError(pDocObj ect, "ERROR_COPYFOLDERCONTENTS")  
  
' Delete the object.  
del ete pDocObj ect
```

## Related Items

See the PCDDocObj ect object.

See the SetProperty method.

See the %CONTENTS\_DIRECTIVE token.

## **%CONTENTS\_DIRECTIVE**

## **%CONTENTS\_DIRECTIVE**

This token is used to manipulate the content of the folder. The following operations are supported:

- %CONTENTS\_MOVE\_DOWN
- %CONTENTS\_MOVE\_UP
- %CONTENTS\_MOVE\_TO\_TOP
- %CONTENTS\_MOVE\_AFTER
- %CONTENTS\_REORDER\_CONTENTS
- %CONTENTS\_WHERE\_USED
- %CONTENTS\_COPY\_CONTENTS

### **Syntax**

```
PCDDocObj ect. SetProperty( _  
    "%CONTENTS_DIRECTIVE", _  
    strOperati on)
```

### **Parameters**

%CONTENTS_DIRECTIVE	The token identifier that indicates that the program will manipulate the contents of a folder.
strOperati on	A string variable that contains one of the supported operations, or the name of the operation as a literal enclosed in double quotation marks.

### **Example**

See the %CONTENTS\_COPY\_CONTENTS [Example on page 310](#) or the %CONTENTS\_MOVE\_AFTER [Example on page 316](#).

### **Related Items**

See the PCDDocObj ect object.

See the SetProperty method.

## **%CONTENTS\_DIRECTIVE**

See the following tokens:

```
%CONTENTS_COPY_CONTENTS  
%CONTENTS_MOVE_AFTER  
%CONTENTS_MOVE_DOWN  
%CONTENTS_MOVE_TO_TOP  
%CONTENTS_MOVE_UP  
%CONTENTS_REORDER_CONTENTS  
%CONTENTS_WHERE_USED
```

## **%CONTENTS\_ITEM**

## **%CONTENTS\_ITEM**

This token identifies the system ID that identifies the object that is the primary subject of the current operation.

### **Syntax**

```
PCDDocObject SetProperty("%CONTENTS_ITEM",  
    LongSystemID)
```

### **Parameters**

**%CONTENTS\_ITEM**

The token identifier that indicates that this statement references the unique identifier for the profiled object involved in the current operation.

**LongSystemID**

The **SYSTEM\_ID** value from the **FOLDER\_ITEM** database table.

### **Usage**

In order to access other information about the profiled object referenced by **%CONTENTS\_ITEM**, the **FOLDER\_ITEM** table is joined to the **PROFILE** table by use of the **DOCNUMBER** value that exists in both tables.

### **Example**

The **%CONTENTS\_ITEM** token is used in several examples. See the **%CONTENTS\_MOVE\_AFTER** [Example on page 316](#), the **%CONTENTS\_MOVE\_DOWN** [Example on page 318](#), the **%CONTENTS\_MOVE\_TO\_TOP** [Example on page 320](#), or the **%CONTENTS\_MOVE\_UP** [Example on page 322](#).

### **Related Items**

See the **PCDDocObject** object.

See the **SetProperty** method.

See the following tokens:

**%CONTENTS\_DIRECTIVE**

**%CONTENTS\_MOVE\_AFTER**

## **%CONTENTS\_ITEM**

%CONTENTS\_MOVE\_DOWN  
%CONTENTS\_MOVE\_TO\_TOP  
%CONTENTS\_MOVE\_UP

## **%CONTENTS\_MOVE\_AFTER**

## **%CONTENTS\_MOVE\_AFTER**

This token is used in conjunction with %CONTENTS\_DIRECTORY to specify that a folder should be repositioned so it follows another folder in the collection.

### **Syntax**

```
PCDDocObj ect. SetProperty( _  
    "%CONTENTS_DIRECTORY", _  
    "%CONTENTS_MOVE_AFTER")
```

### **Parameters**

%CONTENTS_DIRECTORY	The token identifier that indicates that the program will manipulate the contents of a folder.
%CONTENTS_MOVE_AFTER	The token identifier that indicates that one folder is to be listed after another.

### **Example**

```
' Create the doc object.  
Dim pDocObj ect As New PCDDocObj ect  
  
' Check for errors.  
checkError(pDocObj ect, "ERROR_CREATECSI OBJECT_PCDDOCOBJECT")  
  
' Set the DM security token.  
pDocObj ect. SetDST strDST  
  
' Set the current library.  
pDocObj ect. SetProperty("%CONTENTS_SRC_PARENT_LIBRARY", _  
    strSrcFol derLi b  
  
' Set the object type.  
pDocObj ect. SetObj ectType("ContentItem")  
  
' Set the destination library.  
pDocObj ect. SetProperty("%TARGET_LIBRARY", strTgtFol derLi b)  
  
' Set the ID of the item.  
pDocObj ect. SetProperty("%CONTENTS_ITEM", systemID)
```

# %CONTENTS\_MOVE\_AFTER

```
' Specify the place.  
pDocObj ect. SetProperty("%CONTENTS_AFTER_I TEM", afterSystemID)  
  
' Set the command.  
pDocObj ect. SetProperty("%CONTENTS_DI RECTI VE", _  
    "%CONTENTS_MOVE_AFTER")  
  
' Perform the update.  
pDocObj ect. Update()  
  
' Check for errors.  
checkError(pDocObj ect, "ERROR_MOVEAFTERFOLDERI TEM")  
  
' Delete the object.  
delete pDocObj ect
```

## Related Items

See the PCDDocObj ect object.

See the SetProperty method.

See the %CONTENTS\_DI RECTI VE token.

## **%CONTENTS\_MOVE\_DOWN**

## **%CONTENTS\_MOVE\_DOWN**

Use this token in conjunction with %CONTENTS\_DIRECTORY to move a folder down by one position.

### **Syntax**

```
PCDDocObj ect. SetProperty( _  
    "%CONTENTS_DIRECTORY", _  
    "%CONTENTS_MOVE_DOWN")
```

### **Parameters**

%CONTENTS_DIRECTORY	The token identifier that indicates that the program will manipulate the contents of a folder.
%CONTENTS_MOVE_DOWN	The token identifier that the target folder is to be displayed one position further down in the folder collection.

### **Example**

```
' Get the library name.  
Dim strFol derLib As String  
strFol derLib = InputBox( "Enter the library name.")  
  
' Create the doc object.  
pDocObj ect = CreateObj ect("PCDClient.PCDDocObj ect")  
  
' Check for errors.  
checkError(pDocObj ect, "ERROR_CREATECSI OBJECT_PCDDOCOBJECT")  
  
' Set the DM security token.  
pDocObj ect. SetDST(strDST)  
  
' Set the object type.  
pDocObj ect. SetObj ectType("ContentItem")  
  
' Set the destination library.  
pDocObj ect. SetProperty("%TARGET_LIBRARY", strFol derLib)  
  
pDocObj ect. SetProperty("%CONTENTS_ITEM", systemID)  
  
' Set the action.  
pDocObj ect. SetProperty("%CONTENTS_DIRECTORY", _
```

# %CONTENTS\_MOVE\_DOWN

```
"%CONTENTS_MOVE_DOWN")  
  
' Process the update.  
pDocObj ect. Update()  
  
' Check for error(s).  
checkError(pDocObj ect, "ERROR_MOVEDOWNFOLDERITEM")  
  
' Clean-up.  
delete pDocObj ect
```

## Related Items

See the PCDDocObj ect object.

See the SetProperty method.

See the %CONTENTS\_DIRECTIVE token.

## **%CONTENTS\_MOVE\_TO\_TOP**

## **%CONTENTS\_MOVE\_TO\_TOP**

Use this token in conjunction with %CONTENTS\_DIRECTORY to move a folder to the top of the collection.

### **Syntax**

```
PCDDocObj ect. SetProperty( _  
    "%CONTENTS_DIRECTORY", _  
    "%CONTENTS_MOVE_TO_TOP")
```

### **Parameters**

%CONTENTS_DIRECTORY	The token identifier that indicates that the program will manipulate the contents of a folder.
%CONTENTS_MOVE_TO_TOP	This token indicates that the specified folder is to be listed first in the folder collection.

### **Example**

```
' Get the library name.  
Dim strFol derLib As String  
strFol derLib = InputBox( "Enter the library name.")  
  
' Create the doc object.  
pDocObj ect = CreateObj ect("PCDClient.PCDDocObj ect")  
  
' Check for errors.  
checkError(pDocObj ect, "ERROR_CREATECSI OBJECT_PCDDOCOBJECT")  
  
' Set the DM security token.  
pDocObj ect. SetDST(strDST)  
  
' Set the object type.  
pDocObj ect. SetObj ectType("ContentItem")  
  
' Set the destination library.  
pDocObj ect. SetProperty("%TARGET_LIBRARY", strFol derLib)  
  
' Identify the object to be moved.  
pDocObj ect. SetProperty("%CONTENTS_ITEM", systemD)  
  
' Set the action.
```

## %CONTENTS\_MOVE\_TO\_TOP

```
pDocObj ect. SetProperty("%CONTENTS_DIRECTI VE", _  
    "%CONTENTS_MOVE_TO_TOP")  
  
    ' Perform the update.  
    pDocObj ect. Update()  
  
    ' Check for errors.  
    checkError(pDocObj ect, "ERROR_MOVEDOWNFOLDERITEM")  
  
    ' Clean up.  
    delete pDocObj ect
```

### Related Items

See the PCDDocObj ect object.

See the SetProperty method.

See the %CONTENTS\_DIRECTI VE token.

## **%CONTENTS\_MOVE\_UP**

## **%CONTENTS\_MOVE\_UP**

Use this token in conjunction with %CONTENTS\_DIRECTORY to move a folder up by one position.

### **Syntax**

```
PCDDocObj ect. SetProperty( _  
    "%CONTENTS_DIRECTORY", _  
    "%CONTENTS_MOVE_UP")
```

### **Parameters**

%CONTENTS_DIRECTORY	The token identifier that indicates that the program will manipulate the contents of a folder.
%CONTENTS_MOVE_UP	The token identifier that the target folder is to be displayed one position earlier in the folder collection.

### **Example**

```
' Get the library name.  
Dim strFol derLib As String  
strFol derLib = InputBox( "Enter the library name.")  
  
' create the doc object  
pDocObj ect = CreateObj ect("PCDCI ent.PCDDocObj ect")  
  
' Check for errors.  
checkError(pDocObj ect, "ERROR_CREATECSI OBJECT_PCDDOCOBJECT")  
  
' Set the DM security token.  
pDocObj ect. SetDST(strDST)  
  
' Set the object type.  
pDocObj ect. SetObj ectType("ContentItem")  
  
' Set the destination library.  
pDocObj ect. SetProperty("%TARGET_LIBRARY", strFol derLib)  
  
' Identify the object to be moved.  
pDocObj ect. SetProperty("%CONTENTS_ITEM", systemD)  
  
' Set the action.
```

# %CONTENTS\_MOVE\_UP

```
pDocObj ect. SetProperty("%CONTENTS_DIRECTI VE",
                           "%CONTENTS_MOVE_UP")

' Perform the update
pDocObj ect. Update()

' Check for errors.
checkError(pDocObj ect, "ERROR_MOVEDOWNFOLDERITEM")

' Clean up.
del ete pDocObj ect
```

## Related Items

See the PCDDocObj ect object.

See the SetProperty method.

See the %CONTENTS\_DIRECTI VE token.

## **%CONTENTS\_REORDER\_CONTENTS**

## **%CONTENTS\_REORDER\_CONTENTS**

Use this token in conjunction with %CONTENTS\_DIRECTIVE to change the order of the folder collection.

### **Syntax**

```
PCDDocObj ect. SetProperty( _  
    "%CONTENTS_DIRECTIVE", _  
    "%CONTENTS_REORDER_CONTENTS")
```

### **Parameters**

%CONTENTS_DIRECTIVE	The token identifier that indicates that the program will manipulate the contents of a folder.
%CONTENTS_REORDER_CONTENTS	The token identifier that the folder collection is to be reordered according to the sequence specified in the %CONTENTS_REORDER_ARRAY command.

### **Example**

```
' Create the doc obj ects.  
pDocObj ect = CreateObj ect("PCDCI ent.PCDDocObj ect")  
  
' Check the error.  
checkError(pDocObj ect, "ERROR_CREATECSI OBJECT_PCDDOCOBJECT")  
  
' Set the DM security token.  
pDocObj ect. SetDST(strDST)  
  
' Set the object type.  
pDocObj ect. SetObj ectType("ContentsCollecti on")  
  
' Set the destination library.  
pDocObj ect. SetProperty("%TARGET_LI BRARY", strFol derLi b)  
  
' Set the content folder.  
pDocObj ect. SetProperty("%CONTENTS_PARENT", fol derNum)  
  
' Set the version.  
pDocObj ect. SetProperty("%CONTENTS_PARENT_VERSI ON", fol derVersi on)
```

# %CONTENTS\_REORDER\_CONTENTS

```
' Set the new order.  
pDocObj ect. SetProperty("%CONTENTS_REORDER_ARRAY", pOrder)  
  
' Set the action.  
pDocObj ect. SetProperty("%CONTENTS_DIRECTIVE", _  
    "%CONTENTS_REORDER_CONTENTS")  
  
' Perform the update.  
pDocObj ect. Update()  
  
' Check for error(s).  
checkError(pDocObj ect, "ERROR_REORDERFOLDERITEMS")  
  
' Clean up.  
del ete pDocObj ect
```

## Related Items

See the PCDDocObject object.

See the SetProperty method.

See the %CONTENTS\_DIRECTIVE token.

## **%CONTENTS\_SRC\_PARENT**

## **%CONTENTS\_SRC\_PARENT**

This token is used in conjunction with the `%CONTENTS_DIRECTIVE` token and the `%CONTENTS_COPY_CONTENTS` token. It identifies the document number of the folder whose contents are to be copied to another folder.

### **Syntax**

```
PCDDocObj ect. SetProperty( _  
    "%CONTENTS_SRC_PARENT", _  
    IngDocNum)
```

### **Parameters**

`%CONTENTS_SRC_PARENT`

The token identifier that indicates that the document number value in this statement references the source folder item.

`IngDocNum`

The value from the `DOCNUMBER` column of the `PROFILE` table that identifies the folder item whose contents are being copied.

### **Example**

See the `%CONTENTS_COPY_CONTENTS` [Example on page 310](#).

### **Related Items**

See the `PCDDocObj ect` object.

See the `SetProperty` method.

See the following tokens:

```
%CONTENTS_DIRECTIVE  
%CONTENTS_COPY_CONTENTS
```

## **%CONTENTS\_SRC\_PARENT\_LIBRARY**

## **%CONTENTS\_SRC\_PARENT\_LIBRARY**

This token is used in conjunction with the %CONTENTS\_DIRECTORY token and the %CONTENTS\_COPY\_CONTENTS token. It identifies the library of the source folder whose contents are being copied to another folder.

### **Syntax**

```
PCDDocObject SetProperty( _  
    "%CONTENTS_SRC_PARENT_LIBRARY", _  
    intLibraryID)
```

### **Parameters**

%CONTENTS\_SRC\_PARENT\_LIBRARY

The token identifier that indicates that this statement references the library ID number of the folder item.

intLibraryID

The value from the PARENT\_LIBRARY column of the FOLDER\_ITEM table that identifies the library of the folder item whose contents are being copied.

### **Example**

See the %CONTENTS\_COPY\_CONTENTS [Example on page 310](#).

### **Related Items**

See the PCDDocObject object.

See the SetProperty method.

See the following tokens:

```
%CONTENTS_DIRECTORY  
%CONTENTS_COPY_CONTENTS
```

## **%CONTENTS\_SRC\_PARENT\_VERSION**

## **%CONTENTS\_SRC\_PARENT\_VERSION**

This token is used in conjunction with the %CONTENTS\_DIRECTIVE token and the %CONTENTS\_COPY\_CONTENTS token. It identifies the document version of the source whose contents are to be copied to another folder.

### **Syntax**

```
PCDDocObj ect. SetProperty(  
    "%CONTENTS_SRC_PARENT_VERSION",  
    intVerNum)
```

### **Parameters**

%CONTENTS\_SRC\_PARENT\_VERSION

The token identifier that indicates that this statement references the version number of the source document.

intVerNum

The value from the VERSION column of the VERSIONS table that identifies the version of the folder whose contents are being copied.

### **Example**

See the %CONTENTS\_COPY\_CONTENTS [Example on page 310](#).

### **Related Items**

See the PCDDocObj ect object.

See the SetProperty method.

See the following tokens:

```
%CONTENTS_DIRECTIVE  
%CONTENTS_COPY_CONTENTS
```

## **%CONTENTS\_DST\_PARENT**

## **%CONTENTS\_DST\_PARENT**

This token is used in conjunction with the %CONTENTS\_DIRECTIVE token and the %CONTENTS\_COPY\_CONTENTS token. It identifies the document number of the folder that is the destination of the content operation.

### **Syntax**

```
PCDDocObject SetProperty( _  
    "%CONTENTS_DST_PARENT", _  
    LongDocNum)
```

### **Parameters**

%CONTENTS_DST_PARENT	The token identifier that indicates that the document number value in this statement references the destination folder item.
LongDocNum	The value from the DOCNUMBER column of the PROFILE table that identifies the folder that receives the copied item(s).

### **Example**

See the %CONTENTS\_COPY\_CONTENTS [Example on page 310](#).

### **Related Items**

See the PCDDocObject object.

See the SetProperty method.

See the following tokens:

%CONTENTS\_DIRECTIVE  
%CONTENTS\_COPY\_CONTENTS

## **%CONTENTS\_DST\_PARENT\_LIBRARY**

## **%CONTENTS\_DST\_PARENT\_LIBRARY**

This token is used in conjunction with the %CONTENTS\_DIRECTIVE token and the %CONTENTS\_COPY\_CONTENTS token. It identifies the library of the destination folder to a document object is being copied.

### **Syntax**

```
PCDDocObject SetProperty( _  
    "%CONTENTS_DST_PARENT_LIBRARY", _  
    intLibraryID)
```

### **Parameters**

%CONTENTS\_DST\_PARENT\_LIBRARY

The token identifier that indicates that this statement references the library ID number of the folder item.

intLibraryID

The value from the PARENT\_LIBRARY column of the FOLDER\_ITEM table that identifies the library of the folder item that is to receive the document object that is being copied.

### **Example**

See the %CONTENTS\_COPY\_CONTENTS [Example on page 310](#).

### **Related Items**

See the PCDDocObject object.

See the SetProperty method.

See the following tokens:

```
%CONTENTS_DIRECTIVE  
%CONTENTS_COPY_CONTENTS
```

## **%CONTENTS\_DST\_PARENT\_VERSION**

## **%CONTENTS\_DST\_PARENT\_VERSION**

This token is used in conjunction with the %CONTENTS\_DIRECTIVE token and the %CONTENTS\_COPY\_CONTENTS token. It identifies the document version to which document contents are being.

### **Syntax**

```
PCDDocObj ect. SetProperty( _  
    "%CONTENTS_DST_PARENT_VERSION", _  
    "intVerNum")
```

### **Parameters**

%CONTENTS_DST_PARENT_VERSION	The token identifier that indicates that this statement references the version number of the destination document.
intVerNum	The value of the VERSION column of the VERSIONS table that identifies the version of the folder that receives the copied document.

### **Example**

See the %CONTENTS\_COPY\_CONTENTS [Example on page 310](#).

### **Related Items**

See the PCDDocObj ect object.

See the SetProperty method.

See the following tokens:

%CONTENTS\_DIRECTIVE  
%CONTENTS\_COPY\_CONTENTS

## **%CONTENTS\_PARENT**

## **%CONTENTS\_PARENT**

This token is used in conjunction with the %CONTENTS\_REORDER\_ARRAY token to reorganize a folder collection.

### **Syntax**

```
PCDDocObj ect. SetProperty("%CONTENTS_PARENT", _  
    |ngDocNum)
```

### **Parameters**

%CONTENTS\_PARENT

The token identifier. (Although its name implies it references a “parent,” it references the document number of the affected folder.)

I ngDocNum

The document number retrieved through the link in the DOCNUMBER column of the FOLDER\_ITEM table.

### **Example**

See the %CONTENTS\_REORDER\_CONTENTS [Example on page 324](#).

### **Related Items**

See the PCDDocObj ect object.

See the SetProperty method.

See the following tokens:

```
%CONTENTS_PARENT_VERSION  
%CONTENTS_REORDER_CONTENTS  
%CONTENTS_REORDER_ARRAY
```

## **%CONTENTS\_PARENT\_VERSION**

## **%CONTENTS\_PARENT\_VERSION**

This token is used in conjunction with the `%CONTENTS_REORDER_ARRAY` token to reorganize a folder collection.

### **Syntax**

```
PCDDocObj ect. SetProperty( _  
    "%CONTENTS_PARENT_VERSION", _  
    I ngFol derVersi on)
```

### **Parameters**

<code>%CONTENTS_PARENT_VERSION</code>	The token identifier, indicating that version information is being provided.
<code>I ngFol derVersi on</code>	The document number retrieved through the link in the <code>VERS ION_ID</code> column of the <code>VERS IONS</code> table.

### **Example**

See the `%CONTENTS_REORDER_CONTENTS` [Example on page 324](#).

### **Related Items**

See the `PCDDocObj ect` object.

See the `SetProperty` method.

See the following tokens:

```
%CONTENTS_PARENT_VERSI ON  
%CONTENTS_REORDER_CONTENTS  
%CONTENTS_REORDER_ARRAY
```

## **%CONTENTS\_WHERE\_USED**

## **%CONTENTS\_WHERE\_USED**

Use this token in conjunction with %CONTENTS\_DIRECTORY to get the information about where the folder is being used.

### **Syntax**

```
PCDDocObj ect. SetProperty( _  
    "%CONTENTS_DIRECTORY", _  
    "%CONTENTS_WHERE_USED")
```

### **Parameters**

%CONTENTS_DIRECTORY	The token identifier that indicates that the program will manipulate the contents of a folder.
%CONTENTS_WHERE_USED	The token identifier that returns the names of any containers (other than the current container) that contain the search document.

### **Example**

```
' Create the object.  
PropLists = CreateObject("PCDClient.PCDPropertyLists")  
  
' Check for error(s).  
checkError(PropLists, "ERROR_CREATECSI OBJECT_PCDPROPERTYLISTS")  
  
' Set the DM security token.  
PropLists.SetDST(strDST)  
  
' Set the object type.  
PropLists.SetObjectType("ContentsCollection")  
  
' Set the action code.  
PropLists SetProperty("%CONTENTS_DIRECTORY", _  
    "%CONTENTS_WHERE_USED")  
  
' Set the destination library.  
PropLists SetProperty("%TARGET_LIBRARY", Library)  
  
' Set the document number.  
PropLists SetProperty("DOCNUMBER", docNum)
```

## %CONTENTS\_WHERE\_USED

```
' Set the chunk size.  
PropLists.SetChunkFactor(size)  
  
' Execute the search.  
PropLists.Execute()  
  
checkError(PropLists, "ERROR_WHEREUSED")  
While (PropLists.NextRow())  
    ' Process data returned by the search.  
Wend
```

### Related Items

See the PCDDocObject object.

See the SetProperty method.

See the %CONTENTS\_DIRECTORY token.

## **%COPYDOC**

## **%COPYDOC**

This token is used when a document is created by copying content from another document.

### **Syntax**

```
PCDDocObj ect. SetProperty("%COPYDOC", _  
    l ngDocNum)
```

### **Parameters**

<b>%COPYDOC</b>	The token identifier that indicates document content is being copied into a new document.
<b>l ngDocNum</b>	The DOCNUMBER value from the PROFILE table that uniquely identifies the document that is being copied to create the new document.

### **Example**

```
pDocObj ect. SetProperty("%COPYDOC", docnumbertocopy)
```

### **Related Items**

See the PCDDocObject object.

See the SetProperty method.

See the following tokens:

```
%CONTENTS_COPY_CONTENTS  
%COPYDOC  
%COPYDOC_LIBRARY  
%COPYDOC_VERSION
```

## **%COPYDOC\_LIBRARY**

## **%COPYDOC\_LIBRARY**

The %COPYDOC\_LIBRARY token is used when document content from one library is copied to create a new document in another library. This token identifies the library where the original document is located.

### **Syntax**

```
PCDDocObject SetProperty("%COPYDOC_LIBRARY",  
    LibraryID)
```

### **Parameters**

**%COPYDOC\_LIBRARY**

The token identifier that specifies that this statement contains the library where the original document that is to be copied is located.

**LibraryID**

The library identifier of the document that is being copied.

### **Usage**

This token is omitted if the document that is being copied currently exists in the library where the new document is being created.

### **Example**

```
pDocObject SetProperty("%COPYDOC_LIBRARY", Library)
```

### **Related Items**

See the PCDDocObject object.

See the SetProperty method.

See the following tokens:

%CONTENTS\_COPY\_CONTENTS  
%COPYDOC  
%COPYDOC\_LIBRARY  
%COPYDOC\_VERSION

## **%COPYDOC\_VERSION**

## **%COPYDOC\_VERSION**

When document content is being copied to create a new document, this token identifies the version of the document that is to be copied.

### **Syntax**

```
PCDDocObj ect. SetProperty("%COPYDOC_VERSION",  
"l ngVerNum")
```

### **Parameters**

**%COPYDOC\_VERSION**

The token identifier that specifies that this statement contains the version of the original document that is to be copied.

**l ngVerNum**

The VERSI ON value from the VERSI ONS table that identifies which version of the document is to be copied.

### **Example**

```
pDocObj ect. SetProperty("%COPYDOC_VERSION", versi on_id)
```

### **Related Items**

See the PCDDocObj ect object.

See the SetProperty method.

See the following tokens:

```
%CONTENTS_COPY_CONTENTS  
%COPYDOC  
%COPYDOC_LIBRARY  
%COPYDOC_VERSION
```

# %DATA

## %DATA

The Execute method that PCDLookup supports returns both data and metadata. The %DATA token is used to retrieve the data returned by the lookup operation.

### Syntax

```
PCDLookup.GetMetaPropertyValue("%DATA")
```

### Parameters

%DATA	The token identifier used to request the data from a lookup retrieval.
-------	--

### Example

```
' Create the object.  
pClient = CreateObject("PCDClient.PCDLookup")  
  
' Set the DM security token.  
pClient.SetDST(strDST)  
  
' Set the form.  
pClient.SetSearchObject("cyd_defprof")  
  
' Set the lookup ID.  
pClient.SetLookupID("DEPL_PACKAGES")  
  
' Set the target property.  
pClient.SetTargetProperty("PACKAGE_ID")  
  
' Add search library.  
pClient.AddSearchLib("CurrentLibrary")  
  
' Execute the search.  
pClient.Execute()  
  
' Get the data.  
strPkgID = pClient.GetMetaPropertyValue("%Data")  
  
' Report the data in whatever way is required.  
MsgBox("The name of the Deployment Package Initialization file" _  
    & " is: " & strPkgID)
```

# %DATA

## Related Items

See the `PCDLookup` object.

See the `GetMetaPropertyValue` method.

See the following tokens:

`%PROPERTYNAME`

`%PROPERTYTYPE`

`%TITLE`

`%VISIBLE`

## **%DELETE\_ALL**

## **%DELETE\_ALL**

This token is used with the `SetProperty` method that `PCDDocObject` supports. Used with the `%DELETE_OPTION` token, it allows deletion of both the profile and the content of DM objects.

### **Syntax**

```
PCDDocObject SetProperty("%DELETE_OPTION",  
                          "%DELETE_ALL")
```

### **Parameters**

<code>%DELETE_OPTION</code>	The token that indicates the content, or the content and the profile, is to be deleted from a DM object.
<code>%DELETE_ALL</code>	The token that both the profile and the content of the DM object are to be deleted.

### **Example**

See the [%DELETE\\_OPTION Example on page 343](#).

### **Related Items**

See the `PCDDocObject` object.

See the following methods:

`Delete`  
`SetProperty`

See the following tokens:

`%DELETE_EXPUNGE`  
`%DELETE_OPTION`  
`%DELETE_PHYSICAL_FILES`

## **%DELETE\_EXPUNGE**

## **%DELETE\_EXPUNGE**

This token is reserved for future use. At the current time, it is not supported for use by the DM API.

### **Related Items**

See the PCDDocObject object.

See the following methods:

`Delete`  
`SetProperty`

See the following tokens:

`%DELETE_ALL`  
`%DELETE_OPTION`  
`%DELETE_PHYSICAL_FILES`

## **%DELETE\_OPTION**

## **%DELETE\_OPTION**

This token is used with the SetProperty method that PCDDocObj ect supports. It allows deletion of either the content or both the profile and the content of DM objects.

### **Syntax**

```
PCDDocObj ect. SetProperty("%DELETE_OPTION", _  
    strDeleteCmd)
```

### **Parameters**

%DELETE_OPTION	The token that indicates the content, or the content and the profile, is to be deleted from a DM object.
strDeleteCmd	The token that indicates the type of delete operation that is to occur. This variable must resolve either to "%DELETE_PHYSICAL_FILES" or to "%DELETE_ALL".

### **Example**

```
pDel Obj ect. SetDST(strDST)  
pDel Obj ect. SetObj ectType("cyd_defprof")  
pDel Obj ect. SetProperty("%TARGET_LIBRARY", library)  
pDel Obj ect. SetProperty("%OBJECT_IDENTIFIER", docnum)  
  
If (Request("deleteContent"). count>0) _  
Or (Request("searchDeleteContent"). count>0) Then  
    pDel Obj ect. SetProperty("%DELETE_OPTION", _  
        "%DELETE_PHYSICAL_FILES")  
Else  
    pDel Obj ect. SetProperty("%DELETE_OPTION", "%DELETE_ALL")  
End If  
  
' Clean up.  
pDel Obj ect. Delete()
```

### **Related Items**

See the PCDDocObj ect object.

## **%DELETE\_OPTION**

See the following methods:

  Delete  
  SetProperty

See the following tokens:

  %DELETE\_ALL  
  %DELETE\_EXPUNGE  
  %DELETE\_PHYSICAL\_FILES

## **%DELETE\_PHYSICAL\_FILES**

## **%DELETE\_PHYSICAL\_FILES**

This token is used with the SetProperty method that PCDDocObj ect supports. It allows deletion of the content of DM objects. It does not delete the profile of DM objects whose content is deleted.

### **Syntax**

```
PCDDocObj ect. SetProperty("%DELETE_OPTI ON",  
                           "%DELETE_PHYSICAL_FILES")
```

### **Parameters**

%DELETE_OPTI ON	The token that indicates the content, or the content and the profile, is to be deleted from a DM object.
%DELETE_PHYSICAL_FILES	The token that only the content of the DM object is to be deleted. The profile of the DM object will not be deleted.

### **Example**

See the %DELETE\_OPTI ON Example on page 343.

### **Related Items**

See the PCDDocObj ect object.

See the following methods:

Del ete  
SetProperty

See the following tokens:

%DELETE\_ALL  
%DELETE\_EXPUNGE  
%DELETE\_OPTI ON  
%DELETE\_PHYSICAL\_FILES

## **%DOCS\_LIBRARY\_NAME**

## **%DOCS\_LIBRARY\_NAME**

When used with the `PCDSearch` object, this token retrieves the library name associated with the document specified in the search. When used with the `PCDPropertyLists` object, it retrieves the library name associated with the current row of the list.

### **Syntax**

```
PCDPropertyLists.GetCurrentPropertyName( _  
    "%DOCS_LIBRARY_NAME")  
  
PCDSearch.AddReturnProperty( _  
    "%DOCS_LIBRARY_NAME")
```

### **Parameters**

<code>%DOCS_LIBRARY_NAME</code>	The token identifier that indicates that the name of the library is to be extracted from the property list or search results.
---------------------------------	---

### **Example**

```
' Create a PCDSearch object.  
pRelated = CreateObject("PCDClient.PCDSearch")  
  
' Check for errors.  
checkError(pRelated, "ERROR_CREATESEARCH")  
  
' Set the search object.  
pRelated.SetSearchObject('RelatedItemsSearch')  
  
' Set the DM security token.  
pRelated.SetDST(strDST)  
  
' Add the current library.  
pRelated.AddSearchLib(strLibName)  
  
' Set the number of the document to get.  
pRelated.AddSearchCriteria("%OBJECT_IDENTIFIER", docnum)  
  
' Define the properties to be returned.  
pRelated.AddReturnProperty("DOCNUMBER")  
pRelated.AddReturnProperty("%DOCS_LIBRARY_NAME")
```

# %DOCS\_LIBRARY\_NAME

' Execute the search.  
Related. Execute

## Related Items

See the following objects:

PCDPropertyLists  
PCDSearch

See the following methods:

AddReturnProperty  
GetCurrentPropertyName

## **%DOCUMENT\_NUMBER**

## **%DOCUMENT\_NUMBER**

This token is used to specify the document number of the profiled item that the application requires.

### **Syntax**

```
PCDDocObj ect. SetProperty("%DOCUMENT_NUMBER", _  
    "strDocNum")
```

### **Parameters**

<b>%DOCUMENT_NUMBER</b>	The token identifier indicating that the document number is identified in this program statement.
<b>strDocNum</b>	The value from the DOCNUMBER column of the PROFILE table.

### **Example**

```
' Create an object to use to send the contents of the document.  
pPutDoc = CreateObj ect( "PCDClient.PCDPutDoc.1" )  
  
' Check for errors.  
checkError( pPutDoc, "ERROR_CREATEPUTDOC" )  
  
' Set the DM security token.  
pPutDoc. SetDST( strDST )  
  
' Constrain the search to the library, document number,  
' and version.  
pPutDoc. AddSearchCri teria( "%TARGET_LI BRARY", library )  
pPutDoc. AddSearchCri teria( "%DOCUMENT_NUMBER", docnum )  
pPutDoc. AddSearchCri teria( "%VERSI ON_ID", version_id )  
  
' Fi nd the document.  
pPutDoc. Execute()  
  
' Check for errors.  
checkError( pPutDoc, "ERROR_UPLOADACT_PUTDOC" )
```

### **Related Items**

See the PCDDocObj ect object.

## **%DOCUMENT\_NUMBER**

See the SetProperty method.

## **%EFFECTIVE\_RIGHTS**

## **%EFFECTIVE\_RIGHTS**

Each document can be accessed by many users, and it can be customized. Using the Access control on the profile form, an author or document administrator can grant various access permissions to other users.

### Syntax

```
PCDDocObj ect. GetReturnProperty( _  
    "%EFFECTIVE_RIGHTS")
```

### Parameters

%EFFECTIVE_RIGHTS	The token identifier that requests the application to provide the access rights that the current document allows for the current user.
-------------------	--

### Usage

The HasRight method lists the tokens that %EFFECTIVE\_RIGHTS supports. See the discussion of [Usage on page 240](#).

### Example

```
' Create a doc object.  
pDocObj ect = CreateObj ect("PCDCI ent. PCDDocObj ect. 1")  
  
' Check the errors.  
checkError(pDocObj ect, "ERROR_CREATESEARCH")  
  
' Set the user's DM security (DST).  
pDocObj ect. SetDST(strDST)  
  
' Set the object type.  
pDocObj ect. SetObj ectType("cyd_defprof")  
  
' Set the library name, document number, and version.  
pDocObj ect. SetProperty("%TARGET_LIBRARY", library)  
pDocObj ect. SetProperty("%OBJECT_IDENTER", docnumber)  
pDocObj ect. SetProperty("%VERSION_ID", versi on)  
  
' Get the requested information.  
pDocObj ect. Fetch()
```

# **%EFFECTIVE\_RIGHTS**

```
' Check for errors.  
checkError(pDocObj ect, "ERROR_DOCPROFILEDSP_RI GHTS")  
  
' Get the user's effective rights for this document object.  
Set intAccessRi ghts = pDocObj ect.GetReturnProperty(_  
    "%EFFECTIVE_RI GHTS")  
  
' Make sure user has rights to edit the profile.  
If Not (pDocObj ect.HasRight("%PR_EDIT", intAccessRi ghts)) Then  
    ' The user cannot edit this document.  
End If
```

## **Related Items**

See the PCDDocObj ect object.

See the following methods:

GrantRi ght  
HasRi ght  
RevokeRi ght

See the following tokens:

%PR\_ACCESS\_CONTROL  
%PR\_CONTENT\_COPY  
%PR\_CONTENT\_DELETE  
%PR\_CONTENT\_EDIT  
%PR\_CONTENT\_RETRIEVE  
%PR\_CONTENT\_VIEW  
%PR\_EDIT  
%PR\_VIEW  
%RI GHT8  
%RI GHT9

## **%ELAPSED\_TIME**

## **%ELAPSED\_TIME**

This token sets the `ELAPSED_TIME` column of the ACTIVI TYLOG table with the amount of time that the specified document was open for edit or new version activity.

### **Syntax**

```
PCDDocObj ect. SetProperty("%ELAPSED_TIME", _  
    i El apsedTi me)
```

### **Parameters**

`%ELAPSED_TIME`

The token identifier that indicates that this command line contains the elapsed time value.

`i El apsedTi me`

The elapsed time (in seconds) that the specified document was open for edit or new version activity.

### **Example**

```
' Create the object.  
pDocObj ect = CreateObj ect("PCDCI ent. PCDDocObj ect. 1")  
  
' Check for errors.  
checkError( pDocObj ect, "ERROR_CREATEDOCOBJECT" )  
  
' Set the login security information (DST).  
pDocObj ect. SetDST( strDST )  
  
' Use the cyd_defprof form file to set the object properties.  
pDocObj ect. SetObj ectType("cyd_defprof")  
  
' Set the library.  
pDocObj ect. SetProperty( "%TARGET_LIBRARY", Library )  
  
' Set the object number.  
pDocObj ect. SetProperty( "%OBJECT_IDENTIFIER", docnum )  
  
' Set the version number.  
pDocObj ect. SetProperty( "%VERSION_ID", version_id )  
  
' Lock the document for checkout.
```

## **%ELAPSED\_TIME**

```
pDocObject SetProperty( "%STATUS", "%LOCK_FOR_CHECKOUT" )  
  
' Set the elapsed time.  
pDocObject SetProperty( "%ELAPSED_TIME", iElapsedTime )
```

### **Related Items**

See the PCDDocObject object.

See the SetProperty method.

See the following tokens:

%CHECKIN\_DATE  
%CHECKOUT\_COMMENT

## **%ENCAPSULATION\_TYPE**

## **%ENCAPSULATION\_TYPE**

When users indicate that they want to view documents, those documents must be transferred from the server as data streams. There are 3 stream types:

- HTML streams, where documents are converted HTML streams,
- Native streams, where the MIME type of the transferred documents is used to control their display, and
- RIFF streams, where the DM viewer is used to display documents.

The %ENCAPSULATION\_TYPE token is used to convert documents into RIFF streams.

### **Syntax**

```
PCDGetDoc.AddSearchCriteria( _  
    "%ENCAPSULATION_TYPE", "RIFF")
```

### **Parameters**

%ENCAPSULATION_TYPE	The token identifier that indicates that the encapsulation type is being set.
RIFF	The keyword that indicates that the encapsulation type is being set to "RIFF".

### **Usage**

At the present time, "RIFF" is the only value that %ENCAPSULATION\_TYPE supports.

### **Example**

```
' Create the PCDGetDoc object.  
pGetDoc = CreateObject("PCDClient.PCDGetDoc.1")  
  
' Check for errors.  
checkError(pGetDoc, "ERROR_CREATESEARCH")  
  
' Set the DM security token.  
pGetDoc.SetDST(strDST)
```

# %ENCAPSULATION\_TYPE

```
' Check whether it is for the DM Viewer.  
If (rendition = "riff") Then  
    pGetDoc.AddSearchCriteria("%ENCAPSULATION_TYPE", "RIFF")  
End If  
  
' Retrieve the document objects.  
pGetDoc.Execute
```

## Related Items

See the PCDGetDoc object.

See the AddSearchCriteria method.

## **%FILTER\_DISABLED\_ROWS**

## **%FILTER\_DISABLED\_ROWS**

Some tables have a `DISABLED` column, which allows the system to specify that a row is disabled. Use this token to exclude such rows from a search result set.

### **Syntax**

```
PCDLookup.AddUserFilterCriteria( _  
    "%FILTER_DISABLED_ROWS", "Y")
```

### **Parameters**

`%FILTER_DISABLED_ROWS` The token identifier that indicates that this command line will specify whether disabled rows are to be filtered or not.

`Y` Yes, disabled rows are to be filtered out of the result set. (Because the default setting is not to exclude disabled rows, the Server ignores any non-Y value.)

### **Usage**

Because, the default setting is not to exclude disabled rows, the Server ignores any non-Y value.

### **Example**

```
' Create a look-up object.  
pSearch = CreateObject ("PCDClient.PCDLookup.1")  
  
' Tell the DM Server to exclude disabled rows.  
pSearch.AddUserFilterCriteria("%FILTER_DISABLED_ROWS", "Y")  
  
' Delete the object.  
delete pSearch
```

### **Related Items**

See the `PCDLookup` object.

See the `AddUserFilterCriteria` method.

## **%FOLDERITEM\_LIBRARY\_NAME**

## **%FOLDERITEM\_LIBRARY\_NAME**

When a new item is being linked, or added to, a folder, this token identifies the name of the home library that contains this document or folder.

### **Syntax**

```
PCDDocObj ect. SetProperty( _  
    "%FOLDERITEM_LIBRARY_NAME", strLi bName)
```

### **Parameters**

<code>%FOLDERITEM_LIBRARY_NAME</code>	The token identifier that indicates the current command line identifies the library where the specified folder item resides.
<code>strLibName</code>	The name of the library. This value is obtained from the <code>_LIBRARY_NAME</code> column of the <code>REMOTE_LIBRARIES</code> table.

### **Usage**

If this is not set, the default setting points to the home library of container folder.

### **Example**

```
' Create a doc object.  
Set pDocObj ect = CreateObj ect("PCDClient.PCDDocObj ect")  
  
' Set the DM security token.  
pDocObj ect. SetDST(strDST)  
  
' Set the object type  
pDocObj ect. SetObjectType("ContentItem")  
  
' Set the destination library.  
pDocObj ect. SetProperty("%TARGET_LIBRARY", strFolderLib)  
  
' Set the parent folder.  
pDocObj ect. SetProperty("PARENT", folderNum)
```

## **%FOLDERITEM\_LIBRARY\_NAME**

```
' Set the parent folder's version.  
pDocObject SetProperty("PARENT_VERSION", folderVersion)  
  
' Set the document number.  
pDocObject SetProperty("DOCNUMBER", docNum)  
  
' Set the library name.  
pDocObject SetProperty("%FOLDERITEM_LIBRARY_NAME", itemLib)  
  
' Create the object.  
pDocObject.Create()
```

### **Related Items**

See the PCDDocObject object.

See the SetProperty method.

## **%FORM\_APPLICATION**

## **%FORM\_APPLICATION**

When used with the `PCDPropertyList` object, this token retrieves the application ID associated with the form. When used with the `PCDDocObject` object, this token can be used to set the application ID that is associated with a form.

### **Syntax**

```
PCDPropertyList.GetCurrentPropertyValue( _  
    "%FORM_APPLICATION")  
  
PCDDocObject SetProperty("%FORM_APPLICATION", _  
    appId)
```

### **Parameters**

<code>%FORM_APPLICATION</code>	The token identifier that indicates the application ID for the specified form either is being set or retrieved.
<code>appId</code>	The application ID for this form.

### **Example**

```
' Create a document object.  
Set pDocObject = CreateObject("PCDCIent.PCDDocObject.1")  
  
' Check for errors.  
checkError(pDocObject, "ERROR_CREATESEARCH")  
  
' Set the DM security token.  
pDocObject SetProperty(strDST)  
  
' Set the object type.  
pDocObject SetProperty("DocsForm")  
  
' Set the destination library.  
pDocObject SetProperty("%TARGET_LIBRARY", "CurrentLibrary")  
  
' Set the form name.  
pDocObject SetProperty("%FORM_NAME", frmName)  
  
' Set the app ID for the form.  
pDocObject SetProperty("%FORM_APPLICATION", appId)
```

## **%FORM\_APPLICATION**

' Retrieve the requested information.  
pDocObj ect. Fetch()

### **Related Items**

See the following objects:

PCDDocObj ect  
PCDPropertyLi st

See the following methods:

GetCurrentPropertyValue  
SetProperty

See the %FORM\_NAME token.

## **%FORM\_DEFAULT\_PRIMARY**

## **%FORM\_DEFAULT\_PRIMARY**

Use this token to identify the default profile form.

### **Syntax**

```
PCDPropertyList.GetPropertyValue( _  
    "%FORM_DEFAULT_PRIMARY")
```

### **Parameters**

**%FORM\_DEFAULT\_PRIMARY** The token identifier that indicates the default profile form is to be returned.

### **Usage**

Returns “Y” if the form in the current return properties data set is the default profile form. Otherwise, “N” is returned.

### **Example**

```
' Create a doc object and a property list object.  
Dim pDocObject As PCDClient.PCDDocObject  
Dim pFormProperties As PCDClient.PCDPropertyList  
  
' Set the DM security token.  
pDocObject.SetDST(strDST)  
  
' Check for errors.  
checkError(pDocObject, "ERROR_CREATESEARCH")  
  
' Set the object type.  
pDocObject.SetObjectType("DocsFormsList")  
  
' Set the destination library.  
pDocObject SetProperty("%TARGET_LIBRARY", Library)  
  
' Set the search type.  
pDocObject SetProperty("%FORM_LIST_TYPE", "%LIST")  
  
' Get the results.  
pDocObject.Fetch()  
  
' Check for errors.  
checkError(pDocObject, "ERROR_SEARCH")
```

## **%FORM\_DEFAULT\_PRIMARY**

```
' Get all the properties.  
Set pFormProperties = pDocObject.GetReturnProperties()  
  
Dim intNumRows As Integer  
Set intNumRows = pFormProperties.RowCount()  
Dim pProperties As New pFormProperties  
Dim strFormName As String  
Dim intCounter As Integer, strCounter As String  
  
for (intCounter = 1 To intNumRows)  
    Set pProperties = pFormProperties.GetCurrentPropertyValue()  
    If (pProperties.GetPropertyVal ue( _  
        "%FORM_DEFAULT_PRIMARY") = "Y") Then  
        Set strFormName = pProperties.GetPropertyVal ue("%FORM_NAME")  
        Set strCounter = CStr(intCounter)  
        MsgBox(strFormName & " is the default form. It is at " _  
            & "row number " & StrCounter & ".")  
    End If  
Next intCounter
```

### **Related Items**

See the PCDPropertyList object.

See the GetPropertyValue method.

## **%FORM\_LIST\_TYPE**

## **%FORM\_LIST\_TYPE**

This token is used to perform a search of a form. Returns a list properties describing the form.

### **Syntax**

```
PCDDocObj ect. SetProperty("%FORM_LIST_TYPE", _  
    strFormType)
```

### **Parameters**

<b>%FORM_LIST_TYPE</b>	The token that indicates the current command line identifies the type of form that the list operation is to retrieve.
<b>strFormType</b>	A string variable or quoted string that contains the form type that the list is to contain.

### **Usage**

The **%FORM\_LIST\_TYPE** token can take any of the following form types as its second parameter:

- **%HIGHLIGHT**
- **%PROFILE**
- **%SEARCH**

### **Example**

```
'Create a doc object.  
Set pDMObj = CreateObject("PCDCIent.PCDDocObject.1")  
  
' Set the DM security token.  
pDMObj .SetDST(strDST)  
  
' Set the object type.  
pDMObj .SetObjectType("DocsFormsList")  
  
' Set the destination library.  
pDMObj .SetProperty("%TARGET_LIBRARY", lib)  
  
' Search for the profile forms.  
pDMObj .SetProperty("%FORM_LIST_TYPE", "%PROFILE")
```

## **%FORM\_LIST\_TYPE**

```
' Run the search.  
pDMObj .Fetch()
```

### **Related Items**

See the PCDDocObject object.

See the SetProperty method.

## **%FORM\_NAME**

## **%FORM\_NAME**

Use this token to set the name of a form that a search operation uses. After a search has been performed, this token can also be used to retrieve the name of the form used in the search.

### **Syntax**

```
PCDDocObj ect. SetProperty("%FORM_NAME", _  
strFormName)
```

```
PCDPropertyList. GetPropertyValue("%FORM_NAME")
```

### **Parameters**

<code>%FORM_NAME</code>	The token identifier that indicates a form name is either being specified or requested.
<code>strFormName</code>	The name of the form that is to be used for the search.

### **Example**

```
' Create a doc object.  
Set pClient = CreateObject("PCDClient.PCDDocObject.1")  
  
' Check for errors.  
checkError(pClient, "ERROR_CREATESEARCH")  
  
' Set the DM security token.  
pClient.SetDST(strDST)  
  
' Set the search object.  
pClient.SetObjectType("DocsForm")  
  
' Set the library that is to be searched.  
pClient SetProperty("%TARGET_LIBRARY", library)  
  
' Set the form ID.  
pClient SetProperty("%OBJECT_IDENTIFIER", formID)  
  
' Fetch the search results.  
pClient.Fetch()  
  
' Check for errors.
```

# **%FORM\_NAME**

```
checkError(pClient, "ERROR_DOCPROFILEDSP_PROFILEINFOEXECUTE")  
  
' Get the returned properties.  
Set pProperties = pClient.GetReturnProperties()  
  
' Get the form name.  
formName = pProperties.GetPropertyValues("%FORM_NAME")
```

## **Related Items**

See the following objects:

[PCDDocObject](#)  
[PCDPropertyList](#)

See the following methods:

[GetPropertyVal ue](#)  
[SetProperty](#)

See the `%FORM_APPLICATION` token.

## **%FORM\_PROFILE\_DEFAULTS**

## **%FORM\_PROFILE\_DEFAULTS**

Use this token to retrieve the default settings for a form.

### **Syntax**

```
PCDPropertyList.GetPropertyValue( _  
    "%FORM_PROFILE_DEFAULTS")
```

### **Parameters**

<code>%FORM_PROFILE_DEFAULTS</code>	The token identifier that returns the default settings for a form.
-------------------------------------	--

### **Usage**

This token returns a two-dimension safe array of property name and property value pairs. Use the `GetCurrentPropertyName` and `GetCurrentPropertyValue` to iterate through the returned data set containing the default settings.

### **Example**

```
'create the doc object  
Set pDocObj = CreateObject("PCDCIent.PCDDocObject")  
  
'Check for errors.  
If Not (checkError = 0) Then  
    'Process the error.  
End If  
  
'Set the DM security token.  
pDocObj.SetDST(strDST)  
  
'Set the search form.  
pDocObj.SetObjectType("MySearchForm")  
  
'Set the library.  
pDocObj SetProperty("%TARGET_LIBRARY", "MyLib")  
  
'Set the ID number of the document that is to be fetched.  
pDocObj SetProperty("%OBJECT_IDENTIFIER", 1234)  
  
'Get the document.  
pDocObj.Fetch()
```

# %FORM\_PROFILE\_DEFAULTS

```
' Instantiate a PCDPropertyList object.  
pPropList = pDocObj.GetReturnProperties()  
  
' Populate a property list with the profile defaults.  
pProfDefaults = pPropList.GetPropertyVal ue(  
    "%FORM_PROFILE_DEFAULTS")  
  
' Check for errors.  
If Not( pPropList.ErrNumber = 0) Then  
    ' Process the error.  
End If  
  
' Iterate through the default properties.  
pProfDefaults.BeginIter()  
  
while pProfDefaults = 0  
  
    strPropertyName = pProfDefaults.GetCurrentPropertyName()  
    strPropValue = pProfDefaults.GetCurrentPropertyValue()  
  
    ' Process the current property.  
  
    ' Set the pointer to process the next property value.  
    pProfDefaults.NextProperty()  
wend
```

## Related Items

See the `PCDPropertyList` object.

See the `GetPropertyVal ue` method.

## **%FORM\_TITLE**

## **%FORM\_TITLE**

This token is used to get the form title from the search result set. It is usually used when searching for a particular form or a collection of forms.

### **Syntax**

```
PCDPropertyList.GetPropertyValue("%FORM_TITLE")
```

### **Parameters**

**%FORM\_TITLE**

The token that indicates the title of the form is to be returned by the fetch operation.

### **Example**

```
' Create a doc object.  
Set pClient = CreateObject("PCDClient.PCDDocObject.1")  
  
' Check for errors.  
checkError(pClient, "ERROR_CREATESEARCH")  
  
' Set the DM security token.  
pClient.SetDST(strDST)  
  
' Set the search object.  
pClient.SetObjectType("DocsForm")  
  
' Set the library that is to be searched.  
pClient SetProperty("%TARGET_LIBRARY", library)  
  
' Set the form ID.  
pClient SetProperty("%OBJECT_IDENTIFIER", formID)  
  
' Retrieve the search values.  
pClient.Fetch()  
  
' Get the returned properties.  
Set pProperties = pClient.GetReturnProperties()  
  
' Get the form name.  
formName = pProperties.GetPropertyValues("%FORM_TITLE")
```

# %FORM\_TITLE

## Related Items

See the `PCDPropertyList` object.

See the `GetPropertyVal ue` method.

## **%FT\_CHARACTER\_SET**

## **%FT\_CHARACTER\_SET**

This token is used to specify a character set for documents rendered in HTML format.

### **Syntax**

```
PCDGetDoc.AddSearchCriteria( _  
    "%FT_CHARACTER_SET", intCharSetID)
```

### **Parameters**

<code>%FT_CHARACTER_SET</code>	The token identifier that indicates this command statement contains a character set value.
<code>intCharSetID</code>	The number that identifies the character set that is to control how HTML documents are rendered.

### **Usage**

Because of the way the bits are set, the character set ID number is usually shown as a hexadecimal number. For example, to render HTML in Japanese, set the character-set ID to 0x13A40000. The setting for Korean is 0x13B50000.

### **Example**

```
' Create a doc object.  
pGetDoc = CreateObject("PCDCient.PCDGetDoc.1")  
  
' Check for errors.  
checkErrorPortal( pGetDoc )  
  
' Set the DM security token.  
pGetDoc.SetDST(strDST)  
  
' Specify the character set.  
pGetDoc.AddSearchCriteria("%FT_CHARACTER_SET", code)  
  
' Execute the search.  
pGetDoc.Execute()
```

## **%FT\_CHARACTER\_SET**

### **Related Items**

See the `PCDGetDoc` object.

See the `AddSearchCriteria` method.

## **%FT\_CONFIDENCE**

## **%FT\_CONFIDENCE**

This token is used when a full text search is performed. It expresses the relevance of a document returned by the search to the search criteria. The relevance value is expressed as a value from 1 to 5, with 1 representing greater relevance and 5 representing lesser relevance.

### **Syntax**

```
PCDSearch.AddReturnProperty("%FT_CONFIDENCE")
```

### **Parameters**

**%FT\_CONFIDENCE**

The token identifier that indicates that a relevance value is to be returned as part of the search process.

### **Example**

```
' Create a search object.  
Set pClient = CreateObject("PCDClient.PCDSearch.1")  
  
' Add the confidence.  
pClient.AddReturnProperty("%FT_CONFIDENCE")  
  
' Execute the search.  
pClient.Execute()  
  
' Create a variable to hold the retrieved search value.  
Dim vValue As Variant  
  
' Retrieve the data.  
vValue = pClient.GetPropertyVal ue("%FT_CONFIDENCE")  
  
' Process the data.
```

### **Related Items**

See the [PCDSearch object](#).

See the [AddReturnProperty method](#).

See the following tokens:

[%FT\\_FORMAT](#)

## **%FT\_CONFIDENCE**

%FT\_MARKER\_LIST  
%FT\_SCORE  
%FT\_TIMESTAMP  
%FT\_VCC\_LIST  
%FT\_VCC\_RULES  
%SCORE\_GRAPHIC  
%SCORE\_PERCENT

## **%FT\_FORMAT**

## **%FT\_FORMAT**

This token is used when a full text search is performed. This token specifies the document format. It is required by DM Viewer so search documents can be rendered properly.

### **Syntax**

```
PCDSearch.AddReturnProperty("%FT_FORMAT")
```

### **Parameters**

**%FT\_FORMAT**

A token that specifies the document format so that the DM Viewer can render it properly.

### **Example**

```
' Create a search object.  
Set pClient = CreateObject("PCDClient.PCDSearch.1")  
  
' Document format identifier used by DM Viewer.  
pClient.AddReturnProperty("%FT_FORMAT")  
  
' Execute the search.  
pClient.Execute()  
  
' Create a variable to hold the retrieved search value.  
Dim vValue As Variant  
  
' Retrieve the data.  
vValue = pClient.GetPropertyVal ue("%FT_FORMAT")  
  
' Process the data.
```

### **Related Items**

See the PCDSearch object.

See the AddReturnProperty method.

See the following tokens:

%FT\_CONFIDENCE  
%FT\_MARKER\_LIST

## **%FT\_FORMAT**

```
%FT_SCORE  
%FT_TIMESTAMP  
%FT_VCC_LIST  
%FT_VCC_RULES  
%SCORE_GRAPHIC  
%SCORE_PERCENT
```

## **%FT\_MARKER\_LIST**

## **%FT\_MARKER\_LIST**

This token is used when a full text search is performed. It allows the SearchServer™ to mark search terms in PDF files.

### **Syntax**

```
PCDSearch.AddReturnProperty("%FT_MARKER_LIST")
```

### **Parameters**

<b>%FT_MARKER_LIST</b>	The token allows the SearchServer to mark search text in PDF files.
------------------------	---

### **Example**

```
' Create a search object.  
Set pClient = CreateObject("PCDClient.PCDSearch.1")  
  
' This is used by the SearchServer when searching PDF files.  
pClient.AddReturnProperty("%FT_MARKER_LIST")  
  
' Execute the search.  
pClient.Execute()  
  
' Create a variable to hold the retrieved search value.  
Dim vValue As Variant  
  
' Retrieve the data.  
vValue = pClient.GetPropertyVal ue("%FT_MARKER_LIST")  
  
' Process the data.
```

### **Related Items**

See the PCDSearch object.

See the AddReturnProperty method.

See the following tokens:

%FT\_CONFIDENCE  
%FT\_FORMAT  
%FT\_SCORE  
%FT\_TIMESTAMP

## **%FT\_MARKER\_LIST**

%FT\_VCC\_LIST  
%FT\_VCC\_RULES  
%SCORE\_GRAPHIC  
%SCORE\_PERCENT

## **%FT\_SCORE**

## **%FT\_SCORE**

This token is used when a full text search is performed. It expresses the relevance of the search criteria to the document. The relevance score is expressed as a percentage.

### **Syntax**

```
PCDSearch.AddReturnProperty("%FT_SCORE")
```

### **Parameters**

**%FT\_SCORE**

This token returns the relevance score of the search results as a percentage value.

### **Example**

```
' Create a search object.  
Set pClient = CreateObject("PCDClient.PCDSearch.1")  
  
' Return a percentage-based relevance score.  
pClient.AddReturnProperty("%FT_SCORE")  
  
Execute the search.  
pClient.Execute()  
  
' Create a variable to hold the retrieved search value.  
Dim vValue As Variant  
  
' Retrieve the data.  
vValue = pClient.GetPropertyVal ue("%FT_SCORE")  
  
' Process the data.
```

### **Related Items**

See the [PCDSearch object](#).

See the [AddReturnProperty method](#).

See the following tokens:

[%FT\\_CONFIDENCE](#)  
[%FT\\_FORMAT](#)

## %FT\_SCORE

```
%FT_MARKER_LIST  
%FT_TIMESTAMP  
%FT_VCC_LIST  
%FT_VCC_RULES  
%SCORE_GRAPHIC  
%SCORE_PERCENT
```

## **%FT\_SMART\_DOCUMENT**

## **%FT\_SMART\_DOCUMENT**

This token is used to convert the output stream into a MIME-encapsulated, aggregate HTML (MHTML) stream. With Internet Explorer, version 4 or later, the DM Server can send in a single stream both the HTML document and all referenced images.

### **Syntax**

```
PCDGetDoc.AddSearchCriteria( _  
    "FT_SMART_DOCUMENT", "1")
```

### **Parameters**

<code>%FT_SMART_DOCUMENT</code>	The token identifier that instructs the DM Server to send referenced images in the same output stream as it uses for the HTML document.
<code>1</code>	The setting value that activates this token.

### **Usage**

Setting `%FT_SMART_DOCUMENT` equal to `"1"` causes the server to send referenced images with the HTML document. Any other value is ignored.

### **Example**

```
' Create a doc object.  
pGetDoc = CreateObject("PCDClient.PCDGetDoc.1")  
  
' Check for errors.  
checkErrorPortal ( pGetDoc )  
  
' Set the DM security token.  
pGetDoc.SetDST(strDST)  
  
' Tell server to send referenced images with the HTML document.  
pGetDoc.AddSearchCriteria("%FT_SMART_DOCUMENT", "1")  
  
' Get the document.  
pGetDoc.Execute ()
```

## **%FT\_SMART\_DOCUMENT**

### **Related Items**

See the `PCDGetDoc` object.

See the `AddSearchCriteria` method.

## **%FT\_TIMESTAMP**

## **%FT\_TIMESTAMP**

This token is used when a full text search is performed. It returns the last time the object was modified.

### **Syntax**

```
PCDSearch. AddReturnProperty("%FT_TIMESTAMP")
```

### **Parameters**

<code>%FT_TIMESTAMP</code>	Return timestamp information about the search object.
----------------------------	---

### **Usage**

The search returns timestamp information expressed in Unix format, showing the number of seconds since January 1, 1970 at 12:00 A.M.

### **Example**

```
' Create a search object.  
Set pClient = CreateObject("PCDClient.PCDSearch.1")  
  
' Return timestamp information.  
pClient.AddReturnProperty("%FT_TIMESTAMP")  
  
' Execute the search.  
pClient.Execute()  
  
' Create a variable to hold the retrieved search value.  
Dim vValue As Variant  
  
' Retrieve the data.  
vValue = pClient.GetPropertyVal ue("%FT_TIMESTAMP")  
  
' Process the data.
```

### **Related Items**

See the `PCDSearch` object.

See the `AddReturnProperty` method.

See the following tokens:

## **%FT\_TIMESTAMP**

%FT\_CONFIDENCE  
%FT\_FORMAT  
%FT\_MARKER\_LIST  
%FT\_SCORE  
%FT\_VCC\_LIST  
%FT\_VCC\_RULES  
%SCORE\_GRAPHIC  
%SCORE\_PERCENT

## **%FT\_VCC\_LIST**

## **%FT\_VCC\_LIST**

This token is used when a full-text search is performed. It highlights the search term in any documents returned by the search operation.

### **Syntax**

```
PCDSearch.AddReturnProperty("%FT_VCC_LIST")
```

### **Parameters**

<b>%FT_VCC_LIST</b>	Highlight the search terms in the search results.
---------------------	---

### **Example**

```
' Create a search object.  
Set pClient = CreateObject("PCDClient.PCDSearch.1")  
  
' Highlight the search term in the results.  
pClient.AddReturnProperty("%FT_VCC_LIST")  
  
' Execute the search.  
pClient.Execute()  
  
' Create a variable to hold the retrieved search value.  
Dim vValue As Variant  
  
' Retrieve the data.  
vValue = pClient.GetPropertyVal ue("%FT_VCC_LIST")  
  
' Process the data.
```

### **Related Items**

See the PCDSearch object.

See the AddReturnProperty method.

See the following tokens:

%FT\_CONFIDENCE  
%FT\_FORMAT  
%FT\_MARKER\_LIST  
%FT\_SCORE

## **%FT\_VCC\_LIST**

```
%FT_TI MESTAMP  
%FT_VCC_RULES  
%SCORE_GRAPHIC  
%SCORE_PERCENT
```

## **%FT\_VCC\_RULES**

## **%FT\_VCC\_RULES**

This token is used when a full-text search is performed. It is used by the DM Viewer to determine how characters should be counted.

### **Syntax**

```
PCDSearch.AddReturnProperty("%FT_VCC_RULES")
```

### **Parameters**

<b>%FT_VCC_RULES</b>	The token identifier that specifies rules the DM Viewer uses to count characters.
----------------------	---

### **Example**

```
' Create a search object.  
Set pClient = CreateObject("PCDClient.PCDSearch.1")  
  
' Set viewer rules for counting characters.  
pClient.AddReturnProperty("%FT_VCC_RULES")  
  
' Execute the search.  
pClient.Execute()  
  
' Create a variable to hold the retrieved search value.  
Dim vValue As Variant  
  
' Retrieve the data.  
vValue = pClient.GetPropertyVal ue("%FT_VCC_RULES")  
  
' Process the data.
```

### **Related Items**

See the PCDSearch object.

See the AddReturnProperty method.

See the following tokens:

%FT\_CONFIDENCE  
%FT\_FORMAT  
%FT\_MARKER\_LIST  
%FT\_SCORE

## **%FT\_VCC\_RULES**

```
%FT_TI MESTAMP  
%FT_VCC_LI ST  
%SCORE_GRAPHIC  
%SCORE_PERCENT
```

## **%GET\_ALLRELATED**

## **%GET\_ALLRELATED**

Use this token to get all items that relate to the search item, whether they are located in the same library as the search item or in remote libraries.

### **Syntax**

```
PCDSearch.AddSearchCriteria( _  
    "%GET_RELATED_ITEMS", _  
    "%GET_ALLRELATED")
```

### **Parameters**

%GET\_RELATEDITEMS

This token identifier indicates that a search parameter is to be set that indicates whether local or remote libraries are to be searched.

%GET\_ALLRELATED

This token identifier indicates that both library that contains the search object and remote libraries are to be searched for related items.

### **Example**

```
' Create a search object.  
pRelated = CreateObject("PCDCIent.PCDSearch")  
  
' Check for errors.  
checkError(pRelated, "ERROR_CREATESEARCH")  
  
' Set the search type.  
pRelated.SetSearchObject('RelatedItemsSearch')  
  
' Set the DM security token  
pRelated.SetDST(strDST)  
  
' Current display library  
pRelated.AddSearchLibrary()  
  
' Search for items related to this search object.  
pRelated.AddSearchCriteria("%OBJECT_IDENTER", _  
    strDocNum)
```

## **%GET\_ALLRELATED**

```
' Set the related token.  
pRelated.AddSearchCriteria("%GETRELATEDITEMS", _  
    "%GETALLRELATED")
```

### **Related Items**

See the PCDSearch object.

See the AddSearchCriteria method.

See the following tokens:

```
%GET_LOCAL_RELATED  
%GETRELATEDITEMS  
%GETREMOTE_RELATED
```

## **%GET\_LOCAL RELATED**

## **%GET\_LOCAL RELATED**

Use this token to get all the documents related to the search item that are in the same library as the search object.

### **Syntax**

```
PCDDocObject SetProperty( _  
    "%GET RELATED ITEMS", _  
    "%GET_LOCAL RELATED")
```

### **Parameters**

%GET RELATED ITEMS

This token identifier indicates that a search parameter is to be set that indicates whether local or remote libraries are to be searched.

%GET\_LOCAL RELATED

This token identifier indicates that only the library that contains the search object is to be searched for related items.

### **Example**

```
' Create a search object.  
pRelated = CreateObject("PCDClient.PCDSearch")  
  
' Check for errors.  
checkError(pRelated, "ERROR_CREATESEARCH")  
  
' Set the search type.  
pRelated.SetSearchObject('RelatedItemsSearch')  
  
' Set the DM security token.  
pRelated.SetDST(strDST)  
  
' Current display library.  
pRelated.AddSearchLibrary()  
  
' Search for items related to this search object.  
pRelated.AddSearchCriteria("%OBJECT_IDENTIFIER", _  
    strDocNum)  
  
' Set the related token.
```

## **%GET\_LOCAL RELATED**

```
pRelated.AddSearchCriteria("%GET_RELATED_ITEMS",  
                           "%GET_LOCAL RELATED")
```

### **Related Items**

See the `PCDSearch` object.

See the `AddSearchCriteria` method.

See the following tokens:

```
%GET_ALL_RELATED  
%GET_RELATED_ITEMS  
%GET_REMOTE_RELATED
```

## **%GET\_RELATED\_ITEMS**

## **%GET\_RELATED\_ITEMS**

This token indicates that the search should return related items.

### **Syntax**

```
PCDSearch.AddSearchCriteria( _  
    "%GET_RELATED_ITEMS", _  
    strTypeOfRelatedSearch)
```

### **Parameters**

<code>%GET_RELATED_ITEMS</code>	This token identifier indicates that a search parameter is to be set that indicates whether local or remote libraries are to be searched.
<code>strTypeOfRelatedSearch</code>	A string variable or quoted literal that resolves to one of the search types that the <code>%GET_RELATED_ITEMS</code> token supports.

### **Usage**

The `%GET_RELATED_ITEMS` token supports the following search types:

- Searches across both local and remote libraries, using the `%GET_ALL_RELATED` token.
- Searches only the library only, using the `%GET_LOCAL_RELATED` token.
- Searches only libraries that are remote to the current document object, using the `%GET_REMOTE_RELATED` token.

### **Example**

See the [%GET\\_ALL\\_RELATED Example on page 389](#) or the [%GET\\_LOCAL\\_RELATED Example on page 391](#).

### **Related Items**

See the `PCDSearch` object.

## **%GET\_RELATED\_ITEMS**

See the `AddSearchCriteria` method.

See the following tokens:

```
%GET_ALL_RELATED  
%GET_LOCAL_RELATED  
%GET_REMOTE_RELATED
```

## **%GET\_REMOTE RELATED**

## **%GET\_REMOTE RELATED**

This token indicates that the search should only return related items that are located in document libraries that are remote to the library that contains search document.

### **Syntax**

```
PCDSearch.AddSearchCriteria( _  
    "%GET RELATED ITEMS", _  
    "%GET_REMOTE RELATED")
```

### **Parameters**

%GET RELATED ITEMS	This token identifier indicates that a search parameter is to be set that indicates whether local or remote libraries are to be searched.
%GET_REMOTE RELATED	This token identifier indicates that the search is to retrieve related documents only from libraries that are remote to the library that contains the search object.

### **Example**

```
' Create a search object.  
pRelated = CreateObject("PCDCIent.PCDSearch")  
  
' Check for errors.  
checkError(pRelated, "ERROR_CREATESEARCH")  
  
' Set the search type.  
pRelated.SetSearchObject("RelatedItemsSearch")  
  
' Set the DM security token.  
pRelated.SetDST(strDST)  
  
' Current display library.  
pRelated.AddSearchLibrary()  
  
' Search for items related to this search object.  
pRelated.AddSearchCriteria("%OBJECT_IDENTIFIER", _  
    strDocNum)
```

## **%GET\_REMOTE RELATED**

```
' Set the related token.  
pRelated.AddSearchCriteria("%GET RELATED ITEMS", _  
    "%GET_REMOTE RELATED")
```

### **Related Items**

See the PCDSearch object.

See the AddSearchCriteria method.

See the following tokens:

```
%GET_ALL_RELATED  
%GET_LOCAL_RELATED  
%GET RELATED ITEMS
```

## **%HAS\_SUBFOLDERS**

## **%HAS\_SUBFOLDERS**

This token is used if a folder has subfolders.

### **Syntax**

```
PCDPropertyLists. GetPropertyValue( _  
    "%HAS_SUBFOLDERS" )
```

### **Parameters**

<code>%HAS_SUBFOLDERS</code>	The token that identifies whether or not the current folder has subfolders within it.
------------------------------	---

### **Usage**

If the current folder has subfolders of its own, the `%HAS_SUBFOLDERS` token returns the value of "Y".

### **Example**

```
' Create a property list object.  
Set PropLists = CreateObject("PCDClient.PCDPropertyLists")  
  
' Set the DM security token.  
PropLists.SetDST(strDST)  
  
' Set the search type.  
PropLists.SetObjectType("ContentsCollection")  
  
' Execute the search.  
PropLists.Execute()  
  
' Initialize the row pointer.  
PropLists.SetRow(0)  
  
' Check results.  
While ( PropLists.NextRow())  
    hasSubFolder = PropLists.GetPropertyValue("%HAS_SUBFOLDERS")  
    If( hasSubFolder ) Then  
        MsgBox( "This property list object has subfolders." )  
    Else  
        MsgBox("This property list object has no subfolders." )  
    End If  
    PropLists.NextRow
```

# **%HAS\_SUBFOLDERS**

Wend

## **Related Items**

See the `PCDPropertyLists` object.

See the `GetPropertyVal ue` method.

## %HITLIST

## %HITLIST

This token is used with the %FORM\_LIST\_TYPE token to retrieve data items referenced on a HITLIST form.

### Syntax

```
PCDDocObject SetProperty("%FORM_LIST_TYPE",  
                         "%HITLIST")
```

### Parameters

%FORM_LIST_TYPE	The token identifier that indicates the current command line identifies the type of form that the list operation is to retrieve.
%HITLIST	The token identifier that indicates a HITLIST search is to take place.

### Example

```
' Create an object to get property info about forms.  
Set pDocObject = CreateObject("PCDCIent.PCDDocObject.1")  
  
' Set the DM security token.  
pDocObject SetProperty("DM_SECURITY", "1")  
  
' Check for errors.  
checkError(pDocObject, "ERROR_CREATESEARCH")  
  
' Identify the form to search.  
pDocObject SetProperty("FORM_TYPE", "HITLIST")  
  
' Set the destination library.  
pDocObject SetProperty("%TARGET_LIBRARY", "Library")  
  
' Set the form type.  
pDocObject SetProperty("%FORM_LIST_TYPE", "%HITLIST")  
  
' Run the search.  
pDocObject Fetch()
```

## **%HITLIST**

### **Related Items**

See the `PCDDocObject` object.

See the `SetProperty` method.

See the following tokens:

`%FORM_LIST_TYPE`  
`%PROFILE`  
`%SEARCH`

## **%I STREAM\_STATSTG\_CBSIZE\_LOWPART**

## **%I STREAM\_STATSTG\_CBSIZE\_LOWPART**

This token returns the size of a stream.

### **Syntax**

```
PCDGetStream.GetPropertyVal ue( _  
    "%I STREAM_STATSTG_CBSIZE_LOWPART")
```

### **Parameters**

<b>%I STREAM_STATSTG_CBSIZE_LOWPART</b>	The token that returns the size of the content stream in bytes.
---	---

### **Example**

```
' Get the file stream.  
Set pGetStream = pGetDoc.GetPropertyVal ue("%CONTENT")  
  
' Get the size.  
Dim VerFi leSi ze As Long  
Set VerFi leSi ze = pGetStream.GetPropertyValue( _  
    "%I STREAM_STATSTG_CBSIZE_LOWPART")
```

### **Related Items**

See the `PCDGetStream` object.

See the `GetPropertyVal ue` method.

See the `%CONTENT` token.

## **%LOCK**

## **%LOCK**

This token is used in conjunction with the `%STATUS` token to lock a document. See `%STATUS` for further information.

### Syntax

```
PCDDocObj ect. SetProperty("%STATUS", "%LOCK")
```

### Parameters

<code>%STATUS</code>	The token that indicates that this command statement adjusts the status of the document.
<code>%LOCK</code>	The token identifier that indicates that the DM system should lock the document.

### Example

See the `%STATUS` [example on page 472](#).

### Related Items

See the `PCDDocObj ect` object.

See the `SetProperty` method.

See the following tokens:

```
%LOCK_FOR_CHECKOUT  
%MAKE_READ_ONLY  
%REMOVE_READ_ONLY  
%STATUS  
%UNLOCK
```

## **%LOCK\_FOR\_CHECKOUT**

## **%LOCK\_FOR\_CHECKOUT**

This token is used in conjunction with the %STATUS token to lock and check out a document. See %STATUS for further information.

### **Syntax**

```
PCDDocObj ect. SetProperty("%STATUS",  
                           "%LOCK_FOR_CHECKOUT")
```

### **Parameters**

%STATUS	The token that indicates that this command statement adjusts the status of the document.
%LOCK_FOR_CHECKOUT	The token identifier that indicates that the DM system should lock the document and check it out to the specified user.

### **Example**

See the %STATUS example on page 472.

### **Related Items**

See the PCDDocObj ect object.

See the SetProperty method.

See the following tokens:

```
%LOCK  
%MAKE_READ_ONLY  
%REMOVE_READ_ONLY  
%STATUS  
%UNLOCK
```

## **%LOOKUP\_ID**

## **%LOOKUP\_ID**

This token is used to set the ID of the hit-list form that controls the data that will be returned by the look-up operation.

### **Syntax**

```
PCDDocObj ect. SetProperty("%LOOKUP_ID", _  
    strHi tI istName)
```

### **Parameters**

<b>%LOOKUP_ID</b>	The token identifier that informs the DM Server of the name of the hit-list form it will use in the current look-up operation.
<b>strHi tI istName</b>	A string variable or literal string in double quotes that resolves to the name of the hit-list form that is to be used in the look-up operation.

### **Example**

```
' Create a doc object.  
Set pHi tI ist = CreateObj ect("PCDClient. PCDDocObj ect. 1")  
  
' Set the DM security token.  
pHi tI ist. SetDST(strDST)  
  
' Set the object type equal to the hit-list form that  
' controls this look-up operation.  
pHi tI ist. SetObjectType("Hi tI ist")  
  
' Set the target library.  
pHi tI ist. SetProperty("%TARGET_LI BRARY", Library)  
  
' The system ID of the hit-list form.  
pHi tI ist. SetProperty("%LOOKUP_ID", hi tI istName)  
  
' Run the search.  
pHi tI ist. Fetch()
```

### **Related Items**

See the **PCDDocObj ect** object.

## **%LOOKUP\_ID**

See the `SetProperty` method.

## **%MAKE\_READ\_ONLY**

## **%MAKE\_READ\_ONLY**

This token is used in conjunction with the `%STATUS` token to set a document so it cannot be altered. See `%STATUS` for further information.

This token is also used with the `%VERSION_DIRECTIVE` token to set a version so it cannot be altered. See `%VERSION_DIRECTIVE` for further information.

### **Syntax**

```
PCDDocObj ect. SetProperty("%STATUS", _  
    "%MAKE_READ_ONLY")
```

```
PCDDocObj ect. SetProperty("%VERSION_DIRECTIVE", _  
    "%MAKE_READ_ONLY")
```

### **Parameters**

<code>%STATUS</code>	The token that indicates that this command statement adjusts the status of the document.
----------------------	--

<code>%VERSION_DIRECTIVE</code>	The token that indicates that this command statement adjusts the document version settings.
---------------------------------	---

<code>%MAKE_READ_ONLY</code>	The token identifier that indicates that the document or version should be set to read only.
------------------------------	--

### **Usage**

Do the following to make a document read only:

- 1** Create a PCDCI i ent. PCDDocObj ect.
- 2** Set the object type to whatever value is appropriate.
- 3** Set the DM security token (DST).
- 4** Set the `%TARGET_LIBRARY` token equal to the library name.

## **%MAKE\_READ\_ONLY**

- 5** Set the %OBJECT\_IDENTIFIER token equal to the document number of the item that is to be made read only.
- 6** If working with a document, set the %STATUS token equal to %MAKE\_READ\_ONLY, indicating that you want to make this document read only.

If working with a document version, set the %VERSION\_DIRECTIVE token equal to %MAKE\_READ\_ONLY, indicating that you want to make this version read only.

- 7** Execute the [Update](#) method.

The %MakeReadOnly token returns SUCCESS if the document was made read only. If the user does not have sufficient security rights to make the document or version read only, then a PCD\_ERR\_INSUFFICIENT RIGHTS error is returned.

### **Example**

See the [%STATUS example on page 472](#).

See the [%VERSION\\_DIRECTIVE example on page 500](#).

### **Related Items**

See the [PCDDocObject](#) object.

See the [SetProperty](#) method.

See the following tokens:

[%STATUS](#)  
[%REMOVE\\_READ\\_ONLY](#)  
[%VERSION\\_DIRECTIVE](#)

## %MAXDAYS

## %MAXDAYS

If there are many entries in the ACTIVETY table, a search for recently edited documents (RED) can return many records. The %MAXDAYS token can be used to limit the number of days that a RED search examines.

### Syntax

```
PCDRecentDoc.AddSearchCriteria("%MAXDAYS", "30")  
PCDSearch.AddSearchCriteria(_  
    "%MAXDAYS", strNumDays)
```

### Parameters

%MAXDAYS	The token identifier that indicates that the maximum number of days is being set for the current search.
"30"	Specifies that the search is to retrieve documents edited during the previous 30 days.
strNumDays	A string variable that contains the number of days that the RED search is to examine.

### Usage

If the %MAXDAYS token is not set as one of the search criteria, the default is to search for documents edited during the previous 90 days.

### Example

See the AddSearchCriteria [example on page 142](#).

### Related Items

See the following objects:

PCDRecentDoc  
PCDSearch

See the AddSearchCriteria method.

## **%NUM\_COMPONENTS**

## **%NUM\_COMPONENTS**

Where the document is comprised of multiple files (for example, as is the case with some CAD/CAM engineering software), use this token to get the number of components that comprise the current document. For documents that are not comprised of multiple files, the %NUM\_COMPONENTS token returns the number of properties returned in the search result set. (This same value is returned by use of the GetRowsFound method.)

### **Syntax**

```
PCDGetDoc. GetPropertyValue("%NUM_COMPONENTS")
```

### **Parameters**

%NUM\_COMPONENTS

The token identifier that returns either the number of components (for a multi-file document) or the number of rows in the search result data set.

### **Usage**

The value that %NUM\_COMPONENTS returns is only available after the search is executed until the SetRow or NextRow methods are called.

### **Example**

See the %CONTENTS [example on page 306](#).

### **Related Items**

See the PCDGetDoc object.

See the GetPropertyValue method.

## **%OBJECT\_IDENTIFIER**

## **%OBJECT\_IDENTIFIER**

The %OBJECT\_IDENTIFIER token is used to set the document ID number of a document or to retrieve a search object's identifier.

### **Syntax**

```
PCDDocObject. SetProperty("%OBJECT_IDENTIFIER", _  
    I DocID)  
  
PCDDocObject. GetPropertyValue(_  
    "%OBJECT_IDENTIFIER")  
  
PCDSearch. AddSearchCriteria(_  
    "%OBJECT_IDENTIFIER", I DocID)  
  
PCDSearch. GetPropertyValue(_  
    "%OBJECT_IDENTIFIER")
```

### **Parameters**

%OBJECT_IDENTIFIER	The token identifier that indicates that this command line involves a document ID number.
I DocID	If the document ID number is being set, this is the value that is to be used.

### **Example**

```
' Create a search object.  
Set pSearch = CreateObject("PCDCIent.PCDSearch")  
  
' Check for errors.  
checkError(pSearch, "ERROR_CREATECSI OBJECT_PCDSearch")  
  
' Set the DM security token.  
pSearch. SetDST(strDST)  
  
' Add the search library.  
pSearch. AddSearchLib(library)  
  
' Identify the search form.  
pSearch. SetSearchObject("VersionsSearch")  
  
' Add the system ID.  
pSearch. AddSearchCriteria("%OBJECT_IDENTIFIER", I DocNum)
```

# **%OBJECT\_IDENTIFIER**

```
' Execute the search.  
pSearch.Execute()
```

## **Related Items**

See the following objects:

PCDDocObject  
PCDSearch

See the following methods:

AddSearchCriteria  
GetPropertyValue

## **%OBJECT\_TYPE\_ID**

## **%OBJECT\_TYPE\_ID**

Use this token to specify the form that is to be used in the operation.

### **Syntax**

```
PCDDocObject SetProperty("%OBJECT_TYPE_ID", _  
    strFormName)
```

### **Parameters**

<b>%OBJECT_TYPE_ID</b>	The token identifier that indicates that this statement specifies the name of the form that is to be used.
<b>strFormName</b>	A string variable that contains the name of the form that is to be used for this operation.

### **Example**

```
' Create a doc object.  
Set pDocObject = CreateObject("PCDCIent.PCDDocObject")  
  
' Check for errors.  
checkError(pDocObject, "ERROR_CREATECSI OBJECT")  
  
' Set the DM security token.  
pDocObject.SetDST(strDST)  
  
' Set the object type.  
pDocObject SetProperty("Profile")  
  
' Set the form name.  
pDocObject SetProperty("%OBJECT_TYPE_ID", formName)
```

### **Related Items**

See the PCDDocObject object.

See the SetProperty method.

## **%ORDER\_BY**

### **%ORDER\_BY**

This token is used to specify the field on the form that controls the sort order.

#### **Syntax**

```
PCDPropertyLists SetProperty("%ORDER_BY", _  
    strDisplayname)
```

#### **Parameters**

<b>%ORDER_BY</b>	The token identifier that this command statement identifies the field on the form that controls the sort order.
<b>strDisplayname</b>	The name of the field as shown on the form, either as a string variable or enclosed within double quote marks.

#### **Example**

```
' Create a property lists object.  
PropLists = CreateObject("PCDCIent.PCDPropertyLists")  
  
' Set the DM security token.  
PropLists.SetDST(strDST)  
  
' Set the object type.  
PropLists.SetObjectType("RootObjectsCollection")  
  
' Set the target library.  
PropLists SetProperty("%TARGET_LIBRARY", Library)  
  
' Set the order.  
PropLists SetProperty("%ORDER_BY", "DISPLAYNAME")  
  
' Retrieve the property list.  
PropLists.Execute()
```

#### **Related Items**

See the PCDPropertyLists object.

See the SetProperty method.

## **%PCD\_DELETEVERSION**

## **%PCD\_DELETEVERSION**

This token is used with the `%VERSION_DIRECTIVE` token to delete the specified version of the current document.

### **Syntax**

```
PCDDocObj ect. SetProperty("VERSION_DIRECTIVE",  
    "%PCD_DELETEVERSION")
```

### **Parameters**

<code>%VERSION_DIRECTIVE</code>	The token that indicates that this command statement adjusts the document version settings.
<code>%PCD_DELETEVERSION</code>	The token identifier that indicates that the specified version of the current document is being deleted.

### **Example**

See the `%VERSION_DIRECTIVE` [example on page 500](#).

### **Related Items**

See the [PCDDocObj ect](#) object.

See the `SetProperty` method.

See the [%VERSION\\_DIRECTIVE](#) token.

## **%PCD\_NEW\_VERSION**

## **%PCD\_NEW\_VERSION**

This token is used with the `%VERSION_DOCUMENT` token to create a new document version. See `%VERSION_DOCUMENT` for further information.

### **Syntax**

```
PCDDocObj ect. SetProperty("%VERSION_DOCUMENT",  
                           "%PCD_NEW_VERSION")
```

### **Parameters**

`%VERSION_DOCUMENT`

The token that indicates that this command statement adjusts the document version settings.

`%PCD_NEW_VERSION`

The token identifier that indicates that a new version is being created for the specified document version.

### **Example**

See the `%VERSION_DOCUMENT` [example on page 500](#).

### **Related Items**

See the [PCDDocObj ect](#) object.

See the `SetProperty` method.

See the `%VERSION_DOCUMENT` token.

## **%PCD\_NEWSUBVERSION**

## **%PCD\_NEWSUBVERSION**

This token is used with the `%VERSICON_DIRECTIVE` token to create a new document sub-version. See `%VERSICON_DIRECTIVE` for further information.

### **Syntax**

```
PCDDocObj ect. SetProperty("%VERSICON_DIRECTIVE",  
                           "%PCD_NEWSUBVERSION")
```

### **Parameters**

<code>%VERSICON_DIRECTIVE</code>	The token that indicates that this command statement adjusts the document version settings.
----------------------------------	---

<code>%PCD_NEWSUBVERSION</code>	The token identifier that indicates that a new sub-version is being created for the specified document version.
---------------------------------	---

### **Example**

See the `%VERSICON_DIRECTIVE` [example on page 500](#).

### **Related Items**

See the [PCDDocObj ect](#) object.

See the [SetProperty](#) method.

See the [%VERSICON\\_DIRECTIVE](#) token.

## **%PCD\_PARM\_HTML\_RENDERING**

## **%PCD\_PARM\_HTML\_RENDERING**

A document can be rendered in HTML, RIFF or native format to the browser. To render it in HTML, a license should be installed on DM Server. Use this tokenverifies that the appropriate license has been installed.

### **Syntax**

```
PCDDocObj ect. GetPropertyVal ue( _  
    "%PCD_PARM_HTML_RENDERING")
```

### **Parameters**

**%PCD\_PARM\_HTML\_RENDERING**

The token identifier that indicates that the DM Server is to be checked for a valid license that allows documents to be rendered in HTML.

### **Example**

```
' Create a doc object.  
pObj = CreateObj ect("PCDClient. PCDDocObj ect")  
  
' Set the DM security token.  
pObj . SetDST(strDST)  
  
' Set the object type.  
pObj . SetObj ectType(strFormName)  
  
' Set the library.  
pObj . SetProperty("%TARGET_LIBRARY", strLi bName)  
  
' Initialize the token in question.  
pObj . SetProperty("%PCD_PARM_HTML_RENDERING", "")  
  
' Get the results.  
pObj . Fetch()  
  
' Get it  
pProps = pObj . GetReturnProperties()  
  
' Check out the property in question.  
StrHTMLEnabled = pProps. GetPropertyVal ue( _  
    "%PCD_PARM_HTML_RENDERING")
```

## **%PCD\_PARM\_HTML\_RENDERING**

```
If (strHTMLEnabled = "Y") Then  
    MsgBox("A license has been installed on this DM Server.")  
Else  
    MsgBox("No license has been installed on this DM Server.")  
End If
```

### **Related Items**

See the [PCDDocObject](#) object.

See the [GetPropertyVal ue](#) method.

## **%PCD\_PARM\_LIB\_SETTINGS**

## **%PCD\_PARM\_LIB\_SETTINGS**

This token retrieves DM Webtop library configuration parameters that are stored in the SYSTEMPARAMETERS. INC file.

### **Syntax**

```
PCDPropertyList. SetProperty( _  
    "%PCD_PARM_LIB_SETTINGS", _  
    strParmName)
```

### **Parameters**

<code>%PCD_PARM_LIB_SETTINGS</code>	The token identifier that requests the specified library configuration parameter.
<code>strParmName</code>	The name of the configuration parameter that is being requested.

### **Example**

```
' Create a document object.  
Set p0bj = CreateObject("PCDClient.PCDPropertyList")  
  
' Set the DM security token.  
p0bj.SetDST(strDST)  
  
' Specify the form used for this retrieval.  
p0bj.SetObjectType(DocObjectType)  
  
' Set the library.  
p0bj SetProperty("%TARGET_LIBRARY", this.library)  
  
' Set the property name to retrieve.  
p0bj SetProperty("%PCD_PARM_LIB_SETTINGS", paramName)
```

### **Related Items**

See the PCDPropertyList object.

See the SetProperty method.

## **%PCD\_UPDATE\_VERSION**

## **%PCD\_UPDATE\_VERSION**

This token is used with the `%VERSION_DIRECTIVE` token to update create a document version to reflect modifications that have been made to it. See `%VERSION_DIRECTIVE` for further information.

### **Syntax**

```
PCDDocObject SetProperty("%VERSION_DIRECTIVE",  
                         "%PCD_UPDATE_VERSION")
```

### **Parameters**

`%VERSION_DIRECTIVE` The token that indicates that this command statement adjusts the document version settings.

`%PCD_UPDATE_VERSION` The token identifier that indicates that a document version is being updated to reflect modifications made to it.

### **Example**

See the `%VERSION_DIRECTIVE` [example on page 500](#).

### **Related Items**

See the [PCDDocObject](#) object.

See the [SetProperty](#) method.

See the [%VERSION\\_DIRECTIVE](#) token.

## **%PR\_ACCESS\_CONTROL**

## **%PR\_ACCESS\_CONTROL**

This token identifies whether or not a user has authority to control the access of other users to the current document. If the HasRight method indicates that the current user's rights include access control, then the functionality supported by the GrantRight and RevokeRight methods are enabled for this user.

### **Syntax**

```
PCDDocObj ect. HasRight("%PR_ACCESS_CONTROL", _  
    intAccessRights)
```

### **Parameters**

<code>%PR_ACCESS_CONTROL</code>	The token that indicates the application should report whether or not the user has sufficient rights to control access to this document.
<code>intAccessRights</code>	The user's access rights. The rights mask is an unsigned 32-bit integer.

### **Usage**

The HasRight method describes the access rights setting for this token. See its discussion of [Usage on page 240](#).

### **Example**

```
' Create a doc object.  
pDocObj ect = CreateObj ect("PCDCI ent. PCDDocObj ect. 1")  
  
' Check the errors.  
checkError(pDocObj ect, "ERROR_CREATESEARCH")  
  
' Set the user's access security (DST), and the FRM file.  
pDocObj ect. SetDST(strDST)  
  
' Get the object type (form).  
pDocObj ect. SetObj ectType("cyd_defprof")  
  
' Set the library name and document number/version.  
pDocObj ect. SetProperty("%TARGET_LIBRARY", library)  
pDocObj ect. SetProperty("%OBJECT_IDENTIFIER", docnumber)
```

## %PR\_ACCESS\_CONTROL

```
pDocObject SetProperty("%VERSION_ID", version)

' Get the requested information.
pDocObject.Fetch()

' Check for errors.
checkError(pDocObject, "ERROR_DOCPROFILEDSPRIGHTS")

' Get the doc's effective rights for this user.
Set intAccessRights = _
    pDocObject.GetReturnProperty("%EFFECTIVERIGHTS")

' Make sure user has rights to edit the profile.
If Not (pDocObject.HasRight("%PR_ACCESS_CONTROL", _
    intAccessRights)) Then
    ' This user cannot control access to this document.
End If
```

### Related Items

See the PCDDocObject object.

See the following methods:

GrantRight  
HasRight  
RevokeRight

See the following tokens:

%EFFECTIVERIGHTS  
%PR\_ACCESS\_CONTROL  
%PR\_CONTENT\_COPY  
%PR\_CONTENT\_DELETE  
%PR\_CONTENT\_EDIT  
%PR\_CONTENT\_RETRIEVE  
%PR\_CONTENT\_VIEW  
%PR\_EDIT  
%PR\_VIEW  
%RIGHT8  
%RIGHT9

## %PR\_CONTENT\_COPY

## %PR\_CONTENT\_COPY

This token identifies whether or not a user has authority to copy the contents of the current document.

### Syntax

```
PCDDocObj ect. HasRi ght("%PR_CONTENT_COPY", _  
    i ntAccessRi ghts)
```

### Parameters

%PR_CONTENT_COPY	The token that indicates the application should report whether or not the user has sufficient rights to copy the content of the document.
i ntAccessRi ghts	The user's access rights. The rights mask is an unsigned 32-bit integer.

### Usage

The HasRi ght method describes the access rights setting for this token. See its discussion of [Usage on page 240](#).

### Example

```
' Create a doc obj ect.  
pDocObj ect = CreateObj ect("PCDCI i ent. PCDDocObj ect. 1")  
  
' Check the errors.  
checkError(pDocObj ect, "ERROR_CREATESEARCH")  
  
' Set the user's access securi ty (DST), and the FRM fi le.  
pDocObj ect. SetDST(strDST)  
  
' Set the object type (form).  
pDocObj ect. SetObj ectType("cyd_defprof")  
  
' Set the library name and and document number/version.  
pDocObj ect. SetProperty("%TARGET_LI BRARY", library)  
pDocObj ect. SetProperty("%OBJECT_IDEN TI FIER", docnumber)  
pDocObj ect. SetProperty("%VERS ION_ID", versi on)  
  
' Get the requested information.  
pDocObj ect. Fetch()
```

## **%PR\_CONTENT\_COPY**

```
' Check for errors.  
checkError(pDocObj ect, "ERROR_DOCPROFI LEDSP_RI GHTS")  
  
' Get the doc's effective rights for this user.  
Set intAccessRights = _  
    pDocObj ect.GetReturnProperty("%EFFECTIVE_RI GHTS")  
  
' Make sure user has rights to edit the profile.  
If NOT (pDocObj ect.HasRight("%PR_CONTENT_COPY", _  
    intAccessRights)) Then  
    ' Process the error condition.  
End If
```

### **Related Items**

See the PCDDocObj ect object.

See the following methods:

GrantRi ght  
HasRi ght  
RevokeRi ght

See the following tokens:

%EFFECTIVE\_RI GHTS  
%PR\_ACCESS\_CONTROL  
%PR\_CONTENT\_DELETE  
%PR\_CONTENT\_EDIT  
%PR\_CONTENT\_RETRIEVE  
%PR\_CONTENT\_VIEW  
%PR\_EDIT  
%PR\_VIEW  
%RI GHT8  
%RI GHT9

## %PR\_CONTENT\_DELETE

## %PR\_CONTENT\_DELETE

This token identifies whether or not a user has authority to delete the contents of the current document.

### Syntax

```
PCDDocObj ect. HasRi ght("%PR_CONTENT_DELETE", _  
    i ntAccessRi ghts)
```

### Parameters

%PR_CONTENT_DELETE	The token that indicates the application should report whether or not the user has sufficient rights to delete the content of the document.
i ntAccessRi ghts	The user's access rights. The rights mask is an unsigned 32-bit integer.

### Usage

The HasRi ght method describes the access rights setting for this token. See its discussion of [Usage on page 240](#).

### Example

```
' Create a doc obj ect.  
pDocObj ect = CreateObj ect("PCDCI i ent. PCDDocObj ect. 1")  
  
' Check the errors.  
checkError(pDocObj ect, "ERROR_CREATESEARCH")  
  
' Set the user's access securi ty (DST), and the FRM fi le.  
pDocObj ect. SetDST(strDST)  
  
' Set the object type.  
pDocObj ect. SetObj ectType("cyd_defprof")  
  
' Set the library name and and document number/version.  
pDocObj ect. SetProperty("%TARGET_LI BRARY", library)  
pDocObj ect. SetProperty("%OBJECT_IDEN TI FIER", docnumber)  
pDocObj ect. SetProperty("%VERSI ON_ID", versi on)  
  
' Get the requested i nformati on.  
pDocObj ect. Fetch()
```

## **%PR\_CONTENT\_DELETE**

```
' Check for errors.  
checkError(pDocObj ect, "ERROR_DOCPROFILEDSP_RI GHTS")  
  
' Get the document's effective rights for this user.  
Set intAccessRights = _  
    pDocObj ect.GetReturnProperty("%EFFECTIVE_RI GHTS")  
  
' Make sure user has rights to edit the profile.  
If NOT (pDocObj ect.HasRight("%PR_CONTENT_DELETE", _  
    intAccessRights)) Then  
    ' Process the error condition.  
End If
```

### **Related Items**

See the PCDDocObj ect object.

See the following methods:

GrantRi ght  
HasRi ght  
RevokeRi ght

See the following tokens:

%EFFECTIVE\_RI GHTS  
%PR\_ACCESS\_CONTROL  
%PR\_CONTENT\_COPY  
%PR\_CONTENT\_EDIT  
%PR\_CONTENT\_RETRIEVE  
%PR\_CONTENT\_VIEW  
%PR\_EDIT  
%PR\_VIEW  
%RI GHT8  
%RI GHT9

## %PR\_CONTENT\_EDIT

## %PR\_CONTENT\_EDIT

This token identifies whether or not a user has authority to edit the contents of the current document.

### Syntax

```
PCDDocObj ect. HasRi ght("%PR_CONTENT_EDI T", _  
    i ntAccessRi ghts)
```

### Parameters

%PR_CONTENT_EDIT	The token that indicates the application should report whether or not the user has sufficient rights to edit the content of the document.
i ntAccessRi ghts	The user's access rights. The rights mask is an unsigned 32-bit integer.

### Usage

The HasRi ght method describes the access rights setting for this token. See its discussion of [Usage on page 240](#).

### Example

```
' Create a doc obj ect.  
pDocObj ect = CreateObj ect("PCDCI i ent. PCDDocObj ect. 1")  
  
' Check the errors.  
checkError(pDocObj ect, "ERROR_CREATESEARCH")  
  
' Set the user's access securi ty (DST), and the FRM fi le.  
pDocObj ect. SetDST(strDST)  
  
' Set the object type.  
pDocObj ect. SetObj ectType("cyd_defprof")  
  
' Set the library name and and document number/version.  
pDocObj ect. SetProperty("%TARGET_LI BRARY", library)  
pDocObj ect. SetProperty("%OBJECT_IDEN TI FIER", docnumber)  
pDocObj ect. SetProperty("%VERS ION_ID", versi on)  
  
' Get the requested i nformati on.  
pDocObj ect. Fetch()
```

# %PR\_CONTENT\_EDIT

```
' Check for errors.  
checkError(pDocObj ect, "ERROR_DOCPROFI LEDSP_RI GHTS")  
  
' Get the doc's effective rights for this user.  
Set intAccessRights = _  
    pDocObj ect.GetReturnProperty("%EFFECTIVE_RI GHTS")  
  
' Make sure user has rights to edit the profile.  
If Not (pDocObj ect.HasRight("%PR_CONTENT_EDIT", _  
    intAccessRights)) Then  
    ' Process the error condition.  
End If
```

## Related Items

See the PCDDocObj ect object.

See the following methods:

GrantRi ght  
HasRi ght  
RevokeRi ght

See the following tokens:

%EFFECTIVE\_RI GHTS  
%PR\_ACCESS\_CONTROL  
%PR\_CONTENT\_COPY  
%PR\_CONTENT\_DELETE  
%PR\_CONTENT\_RETRIEVE  
%PR\_CONTENT\_VIEW  
%PR\_EDIT  
%PR\_VIEW  
%RI GHT8  
%RI GHT9

## **%PR\_CONTENT\_RETRIEVE**

## **%PR\_CONTENT\_RETRIEVE**

This token identifies whether or not a user has authority to retrieve the content of the current document.

### **Syntax**

```
PCDDocObj ect. HasRi ght("%PR_CONTENT_RETRI EVE", _  
AccessRi ghts)
```

### **Parameters**

<code>%PR_CONTENT_RETRI EVE</code>	The token that indicates the application should report whether or not the user has sufficient rights to retrieve the content of the document.
<code>intAccessRi ghts</code>	The user's access rights. The rights mask is an unsigned 32-bit integer.

### **Usage**

The `HasRi ght` method describes the access rights setting for this token. See its discussion of [Usage on page 240](#).

### **Example**

```
' Create a doc obj ect.  
pDocObj ect = CreateObj ect("PCDCI i ent. PCDDocObj ect. 1")  
  
' Check for errors.  
checkError(pDocObj ect, "ERROR_CREATESEARCH")  
  
' Set the user's access securi ty (DST), and the FRM fi le.  
pDocObj ect. SetDST(strDST)  
  
' Set the object type.  
pDocObj ect. SetObj ectType("cyd_defprof")  
  
' Set the library name and and document number/version.  
pDocObj ect. SetProperty("%TARGET_LI BRARY", library)  
pDocObj ect. SetProperty("%OBJECT_IDEN TI FIER", docnumber)  
pDocObj ect. SetProperty("%VERSI ON_ID", versi on)  
  
' Get the requested information.  
pDocObj ect. Fetch()
```

## **%PR\_CONTENT\_RETRIEVE**

```
' Check for errors.  
checkError(pDocObj ect, "ERROR_DOCPROFILEDSP_RI GHTS")  
  
' Get the user's effective rights for this document.  
Set intAccessRights = _  
    pDocObj ect.GetReturnProperty("%EFFECTIVE_RI GHTS")  
  
' Make sure user has rights to edit the profile.  
If Not (pDocObj ect.HasRight("%PR_CONTENT_RETRIEVE", _  
    intAccessRights)) Then  
    ' The current user does not have sufficient authority to  
    ' retrieve the contents of the current document.  
End If
```

### **Related Items**

See the PCDDocObj ect object.

See the following methods:

GrantRi ght  
HasRi ght  
RevokeRi ght

See the following tokens:

%EFFECTIVE\_RI GHTS  
%PR\_ACCESS\_CONTROL  
%PR\_CONTENT\_COPY  
%PR\_CONTENT\_DELETE  
%PR\_CONTENT\_EDIT  
%PR\_CONTENT\_VIEW  
%PR\_EDIT  
%PR\_VIEW  
%RI GHT8  
%RI GHT9

## %PR\_CONTENT\_VIEW

## %PR\_CONTENT\_VIEW

This token identifies whether or not a user has authority to view the content of the current document.

### Syntax

```
PCDDocObj ect. HasRi ght("%PR_CONTENT_VI EW", _  
    i ntAccessRi ghts)
```

### Parameters

%PR_CONTENT_VI EW	The token that indicates the application should report whether or not the user has sufficient rights to view the content of the document.
i ntAccessRi ghts	The user's access rights. The rights mask is an unsigned 32-bit integer.

### Usage

The HasRi ght method describes the access rights setting for this token. See its discussion of [Usage on page 240](#).

### Example

```
' Create a doc obj ect.  
pDocObj ect = CreateObj ect("PCDCI i ent. PCDDocObj ect. 1")  
  
' Check the errors.  
checkError(pDocObj ect, "ERROR_CREATESEARCH")  
  
' Set the user's access securi ty (DST), and the FRM fi le.  
pDocObj ect. SetDST(strDST)  
  
' Set the object type.  
pDocObj ect. SetObj ectType("cyd_defprof")  
  
' Set the library name and and document number/version.  
pDocObj ect. SetProperty("%TARGET_LI BRARY", library)  
pDocObj ect. SetProperty("%OBJECT_IDEN TI FIER", docnumber)  
pDocObj ect. SetProperty("%VERSI ON_ID", versi on)  
  
' Get the requested i nformation.  
pDocObj ect. Fetch()
```

# **%PR\_CONTENT\_VIEW**

```
' Check for errors.  
checkError(pDocObject, "ERROR_DOCPROFILEDSP_RIGHTS")  
  
' Get the doc's effective rights for this user.  
Set intAccessRights = _  
    pDocObject.GetReturnProperty("%EFFECTIVE RIGHTS")  
  
' Make sure user has rights to edit the profile.  
If Not (pDocObject.HasRight("%PR_CONTENTVIEW", _  
    intAccessRights)) Then  
    ' The user does not have sufficient rights to view the  
    ' content of this document.  
End If
```

## **Related Items**

See the PCDDocObject object.

See the following methods:

GrantRight  
HasRight  
RevokeRight

See the following tokens:

%EFFECTIVE RIGHTS  
%PR\_ACCESS\_CONTROL  
%PR\_CONTENT\_COPY  
%PR\_CONTENT\_DELETE  
%PR\_CONTENT\_EDIT  
%PR\_CONTENT\_RETRIEVE  
%PR\_EDIT  
%PR\_VIEW  
%RIGHT8  
%RIGHT9

## **%PR\_EDIT**

## **%PR\_EDIT**

This token identifies whether or not a user has authority to edit the current document.

### **Syntax**

```
PCDDocObj ect. HasRi ght("%PR_CONTENT_EDI T", _  
    i ntAccessRi ghts)
```

### **Parameters**

<code>%PR_CONTENT_EDI T</code>	The token that indicates the application should report whether or not the user has sufficient rights to edit the content of the document.
<code>i ntAccessRi ghts</code>	The user's access rights. The rights mask is an unsigned 32-bit integer..

### **Usage**

The `HasRi ght` method describes the access rights setting for this token. See its discussion of [Usage on page 240](#).

### **Example**

```
' Create a doc obj ect.  
pDocObj ect = CreateObj ect("PCDCI i ent. PCDDocObj ect. 1")  
  
' Check the errors.  
checkError(pDocObj ect, "ERROR_CREATESEARCH")  
  
' Set the user's access securi ty (DST), and the FRM fi le.  
pDocObj ect. SetDST(strDST)  
  
' Set the object type.  
pDocObj ect. SetObj ectType("cyd_defprof")  
  
' Set the library name and and document number/version.  
pDocObj ect. SetProperty("%TARGET_LI BRARY", library)  
pDocObj ect. SetProperty("%OBJECT_IDEN TI FIER", docnumber)  
pDocObj ect. SetProperty("%VERSI ON_ID", versi on)  
  
' Get the requested i nformati on.  
pDocObj ect. Fetch()
```

## **%PR\_EDIT**

```
' Check for errors.  
checkError(pDocObject, "ERROR_DOCPROFILEDSP_RIGHTS")  
  
' Get the doc's effective rights for this user.  
Set intAccessRights = _  
    pDocObject.GetReturnProperty("%EFFECTIVE RIGHTS")  
  
' Make sure user has rights to edit the profile.  
If Not (pDocObject.HasRight("%PR_EDIT", intAccessRights)) Then  
    ' The user is not authorized to edit this document.  
End If
```

### **Related Items**

See the PCDDocObject object.

See the following methods:

GrantRight  
HasRight  
RevokeRight

See the following tokens:

%EFFECTIVE RIGHTS  
%PR\_ACCESS\_CONTROL  
%PR\_CONTENT\_COPY  
%PR\_CONTENT\_DELETE  
%PR\_CONTENT\_EDIT  
%PR\_CONTENT\_RETRIEVE  
%PR\_CONTENT\_VIEW  
%PR\_VIEW  
%RIGHT8  
%RIGHT9

## **%PR\_VIEW**

### **%PR\_VIEW**

This token identifies whether or not a user has authority to view the profile of the current document.

#### **Syntax**

```
PCDDocObj ect. HasRi ght("%PR_VI EW",  
                           _  
                           i ntAccessRi ghts)
```

#### **Parameters**

<code>%PR_VI EW</code>	The token that indicates the application should report whether or not the user has sufficient rights to view the content of the document.
<code>i ntAccessRi ghts</code>	The user's access rights. The rights mask is an unsigned 32-bit integer.

#### **Usage**

The `HasRi ght` method describes the access rights setting for this token. See its discussion of [Usage on page 240](#).

#### **Example**

```
' Create a doc obj ect.  
pDocObj ect = CreateObj ect("PCDCI i ent. PCDDocObj ect. 1")  
  
' Check the errors.  
checkError(pDocObj ect, "ERROR_CREATESEARCH")  
  
' Set the user's access securi ty (DST), and the FRM fi le.  
pDocObj ect. SetDST(strDST)  
  
' Set the object type.  
pDocObj ect. SetObj ectType("cyd_defprof")  
  
' Set the library name and and document number/version.  
pDocObj ect. SetProperty("%TARGET_LI BRARY", library)  
pDocObj ect. SetProperty("%OBJECT_IDEN TI FIER", docnumber)  
pDocObj ect. SetProperty("%VERSI ON_ID", versi on)  
  
' Get the requested i nformati on.  
pDocObj ect. Fetch()
```

# **%PR\_VIEW**

```
' Check for errors.  
checkError(pDocObj ect, "ERROR_DOCPROFILEDSP_RI GHTS")  
  
' Get the user's effective rights for this document.  
Set intAccessRights = _  
    pDocObj ect.GetReturnProperty("%EFFECTIVE_RI GHTS")  
  
' Make sure user has rights to edit the profile.  
If Not (pDocObj ect.HasRi ght("%PR_VI EW", intAccessRights)) Then  
    ' The user does not have sufficient access rights to view  
    ' the profile of the current document.  
End If
```

## **Related Items**

See the PCDDocObj ect object.

See the following methods:

GrantRi ght  
HasRi ght  
RevokeRi ght

See the following tokens:

%EFFECTIVE\_RI GHTS  
%PR\_ACCESS\_CONTROL  
%PR\_CONTENT\_COPY  
%PR\_CONTENT\_DELETE  
%PR\_CONTENT\_EDIT  
%PR\_CONTENT\_RETRIEVE  
%PR\_CONTENT\_VI EW  
%PR\_EDIT  
%RI GHT8  
%RI GHT9

## **%PRIMARY\_KEY**

## **%PRIMARY\_KEY**

This token allows the application to get the SYSTEM\_ID column.

### **Syntax**

```
PCDSearch.AddSearchCriteria("%PRIMARY_KEY",  
                           strDocID)
```

### **Parameters**

<code>%PRIMARY_KEY</code>	The token identifier that indicates the primary key is to be returned.
<code>strDocID</code>	The document ID number of the object for which the system ID number is to be returned.

### **Example**

```
' Create a search object.  
Set pDocNumber = CreateObject("PCDClient.PCDSearch.1")  
  
' Check for errors.  
checkError(pDocNumber, "ERROR_CREATESEARCH")  
  
' Set the DM security token, the library, and the form to be used.  
pDocNumber.SetDST(strDST)  
pDocNumber SetProperty("%TARGET_LIBRARY", strLibName)  
pDocNumber SetObjectType("DEF_PROF")  
  
' Constrain the search to the specified docnumber.  
pDocNumber.AddSearchCriteria("%PRIMARY_KEY", system_id)  
  
' Execute the search.  
pDocNumber.Execute
```

### **Related Items**

See the PCDSearch Object.

See the AddSearchCriteria method.

# %PROFILE

## %PROFILE

This token is used to return information about the default profile form for the user's primary group.

### Syntax

```
PCDDocObj ect. SetProperty("%FORM_LIST_TYPE", _  
    "%PROFILE")
```

### Parameters

%FORM_LIST_TYPE	The token that indicates the current command line identifies the type of form that the list operation is to retrieve.
%PROFILE	The token identifier that indicates information about the default Profile form is to be retrieved.

### Example

```
' Create a doc object.  
Set pDocObj ect = CreateObj ect("PCDCI ent. PCDDocObj ect. 1")  
  
' Check for errors.  
checkError(pDocObj ect, "ERROR_CREATESEARCH")  
  
' Set the DM security token.  
pDocObj ect. SetDST(strDST)  
  
' Set the object type.  
pDocObj ect. SetObj ectType("DocsFormsList")  
  
' Set the library.  
pDocObj ect. SetProperty("%TARGET_LIBRARY", strLibName)  
  
' Get the default profile form for the current group.  
pDocObj ect. SetProperty("%FORM_LIST_TYPE", "%PROFILE")  
  
' Retrieve the specified information.  
pDocObj ect. Fetch()
```

# %PROFILE

## Related Items

See the `PCDDocObject` object.

See the `SetProperty` method.

See the `%FORM_LIST_TYPE` token.

## %PROPERTYNAME

## %PROPERTYNAME

The Execute method that PCDLookup supports returns both data and metadata. The %PROPERTYNAME token is used to retrieve the name of the property as shown on the base form. Usually, this is the name of the column in the SQL database.

### Syntax

```
PCDLookup.GetMetaPropertyValue("%PROPERTYNAME")
```

### Parameters

%PROPERTYNAME	The token identifier used to request the name of the property.
---------------	--

### Example

```
' Create the object.  
Set pClient = CreateObject("PCDClient.PCDLookup")  
  
' Set the DM security token.  
pClient.SetDST(strDST)  
  
' Set the form.  
pClient.SetSearchObject("cyd_defprof")  
  
' Set the lookup ID.  
pClient.SetLookupId("DEPL_PACKAGES")  
  
' Set the target property.  
pClient.SetTargetProperty("PACKAGE_ID")  
  
' Add the search library.  
pClient.AddSearchLibrary(strLibName)  
  
' Execute the search.  
pClient.Execute()  
  
' Get the name.  
Set strPropertyName = pClient.GetMetaPropertyValue("%PROPERTYNAME")
```

# %PROPERTYNAME

## Related Items

See the `PCDLookup` object.

See the `GetMetaPropertyValue` method.

See the following tokens:

`%DATA`

`%PROPERTYTYPE`

`%TITLE`

`%VISIBILITY`

## %PROPERTYTYPE

## %PROPERTYTYPE

The Execute method that PCDLookup supports returns both data and metadata. The %PROPERTYTYPE token is used to retrieve the data type of the property as shown on the base form.

### Syntax

```
PCDLookup.GetMetaPropertyValue("%PROPERTYTYPE")
```

### Parameters

%PROPERTYTYPE

The token identifier used to request the data type of the property.

### Example

```
' Create the object.  
Set pClient = CreateObject("PCDClient.PCDLookup")  
  
' Set the DM security token.  
pClient.SetDST(strDST)  
  
' Set the form.  
pClient.SetSearchObject("cyd_defprof")  
  
' Set the lookup ID to a SQL table.  
pClient.SetLookupID("DEPL_PACKAGES")  
  
' Set the target property to a column in the SQL table.  
pClient.SetTargetProperty("PACKAGE_ID")  
  
' Specify the search library.  
pClient.AddSearchLib(strLibName)  
  
' Execute the search.  
pClient.Execute()  
  
' Get the name of the column.  
Set strColumnName = pClient.GetMetaPropertyValue("%PROPERTYTYPE")
```

### Related Items

See the PCDLookup object.

## %PROPERTYTYPE

See the `GetMetaPropertyValue` method.

See the following tokens:

%DATA  
%PROPERTYNAME  
%TITLE  
%VISIBLE

## **%PUBLISH\_VERSION**

## **%PUBLISH\_VERSION**

Use this token to publish a document version.

### **Returns**

The %PUBLISH\_VERSION token returns SUCCESS if the document version was published without error. If the user does not have sufficient security rights to make the document version read only, then a PCD\_ERR\_INSUFFICIENT RIGHTS error will be returned.

### **Usage**

To publish a version, follow the following steps:

- 1** Create a PCDCI intent. PCDDocObj ect.
- 2** Set the object type to whatever value is appropriate.
- 3** Set the DM security token (DST).
- 4** Set the %TARGET\_LIBRARY token equal to the library name.
- 5** Set the %OBJECT\_IDENTIFIER token equal to the document number of the document version that is to be published.
- 6** Set the %VERSION\_ID token equal to the version number to which this publish action applies.
- 7** Set the %VERSION\_DIRECTIVE token equal to %PUBLISH\_VERSION, indicating that you want to make this document version read only.
- 8** Execute the [Update](#) method.

### **Related Items**

See the [PCDDocObject](#) object.

See the [%UNPUBLISH\\_VERSION](#) method.

See the following tokens:

[%ADD\\_ATTACHMENT](#)

[%MAKE\\_READONLY](#)

## %PUBLISH\_VERSION

%PCD\_DELETEVERS ION  
%PCD\_NEW\_VERSI ON  
%PCD\_NEWSUBVERS ION  
%PCD\_UPDATE\_VERSI ON  
%REMOVE\_READ\_ONLY  
%UNPUBLI SH\_VERSI ON  
%VERS ION\_DIRECTI VE

## **%QS\_DELETE**

## **%QS\_DELETE**

This token identifies whether or not a user has authority to delete a Quick Search.

### **Syntax**

```
PCDDocObject. HasRight("%QS_DELETE",  
    intAccessRights)
```

### **Parameters**

<b>%QS_DELETE</b>	The token that indicates the application should report whether or not the user has sufficient rights to delete the specified Quick Search.
<b>intAccessRights</b>	The user's access rights. The rights mask is an unsigned 32-bit integer.

### **Usage**

The `HasRight` method describes the access rights setting for this token. See its discussion of [Usage on page 240](#).

### **Example**

```
' Create a doc object.  
pDocObject = CreateObject("PCDCIent.PCDDocObject.1")  
  
' Check for errors.  
checkError(pDocObject, "ERROR_CREATESEARCH")  
  
' Set the user's access security (DST).  
pDocObject.SetDST(strDST)  
  
' Set the name of the form object.  
pDocObject.SetObjectType("cyd_qbeprof")  
  
' Set the library name and document number/version.  
pDocObject SetProperty("%TARGET_LIBRARY", Library)  
pDocObject SetProperty("%OBJECT_IDENTER", docnumber)  
pDocObject SetProperty("%VERSION_ID", version)  
  
' Get the requested information.  
pDocObject.Fetch()
```

## %QS\_DELETE

```
' Check for errors.  
checkError(pDocObject, "ERROR_DOCPROFILE_DSP_RIGHTS")  
  
' Get the user's effective rights for this document.  
Set intAccessRights = _  
    pDocObject.GetReturnProperty("%EFFECTIVE RIGHTS")  
  
' Make sure user has rights to edit the profile.  
If Not (pDocObject.HasRight("%QS_DELETE", intAccessRights)) Then  
    ' The user does not have sufficient access rights to  
    ' delete the specified Quick Search.  
End If
```

### Related Items

See the PCDDocObject object.

See the following methods:

GrantRight  
HasRight  
RevokeRight

See the following tokens:

%QS\_EDIT  
%QS\_DELETE

## **%QS\_EDIT**

## **%QS\_EDIT**

This token identifies whether or not a user has authority to edit the specified Quick Search.

### **Syntax**

```
PCDDocObject HasRight("%QS_EDIT",  
                      intAccessRights)
```

### **Parameters**

<b>%QS_EDIT</b>	The token that indicates the application should report whether or not the user has sufficient rights to edit the specified Quick Search.
<b>intAccessRights</b>	The user's access rights. The rights mask is an unsigned 32-bit integer.

### **Usage**

The HasRight method describes the access rights setting for this token. See its discussion of [Usage on page 240](#).

### **Example**

```
' Create a doc object.  
pDocObject = CreateObject("PCDCIent.PCDDocObject.1")  
  
' Check for errors.  
checkError(pDocObject, "ERROR_CREATESEARCH")  
  
' Set the user's access security (DST).  
pDocObject.SetDST(strDST)  
  
' Set the form object.  
pDocObject.SetObjectType("cyd_qbeprof")  
  
' Set the library name and document number/version.  
pDocObject SetProperty("%TARGET_LIBRARY", Library)  
pDocObject SetProperty("%OBJECT_IDENTER", docnumber)  
pDocObject SetProperty("%VERSION_ID", version)  
  
' Get the requested information.  
pDocObject.Fetch()
```

# %QS\_EDIT

```
' Check for errors.  
checkError(pDocObject, "ERROR_DOCPROFILEDSP_RIGHTS")  
  
' Get the user's effective rights for this document.  
Set intAccessRights = _  
    pDocObject.GetReturnProperty("%EFFECTIVE RIGHTS")  
  
' Make sure the user has rights to edit the profile.  
If Not (pDocObject.HasRight("%QS_EDIT", intAccessRights)) Then  
    ' The user does not have sufficient access rights to edit  
    ' the specified Quick Search.  
End If
```

## Related Items

See the PCDDocObject object.

See the following methods:

GrantRight  
HasRight  
RevokeRight

See the following tokens:

[%QS\\_DELETE](#)  
[%QS\\_VIEW](#)

## **%QS\_VIEW**

## **%QS\_VIEW**

This token identifies whether or not a user has authority to view the specified Quick Search.

### **Syntax**

```
PCDDocObject. HasRight("%QS_VIEW",  
    intAccessRights)
```

### **Parameters**

<code>%PR_VIEW</code>	The token that indicates the application should report whether or not the user has sufficient rights to view the Quick Search.
<code>intAccessRights</code>	The user's access rights. The rights mask is an unsigned 32-bit integer..

### **Usage**

The `HasRight` method describes the access rights setting for this token. See its discussion of [Usage on page 240](#).

### **Example**

```
' Create a doc object.  
pDocObject = CreateObject("PCDCIent.PCDDocObject.1")  
  
' Check for errors.  
checkError(pDocObject, "ERROR_CREATESEARCH")  
  
' Set the DM security token.  
pDocObject.SetDST(strDST)  
  
' Set the form object.  
pDocObject.SetObjectType("cyd_qbeprof")  
  
' Set the library name and document number/version.  
pDocObject SetProperty("%TARGET_LIBRARY", Library)  
pDocObject SetProperty("%OBJECT_IDENTIFIER", docnumber)  
pDocObject SetProperty("%VERSION_ID", version)  
  
' Get the requested information.  
pDocObject.Fetch()
```

## %QS\_VIEW

```
' Check for errors.  
checkError(pDocObject, "ERROR_DOCPROFILEDSP_RIGHTS")  
  
' Get the doc's effective rights for this user.  
Set intAccessRights = _  
    pDocObject.GetReturnProperty("%EFFECTIVE RIGHTS")  
  
' Make sure user has rights to view and execute the search form.  
If Not (pDocObject.HasRight("%QS_VIEW", intAccessRights)) Then  
    ' The user is a trustee for this Quick Search. The user  
    ' cannot execute this Quick Search.  
End If
```

### Related Items

See the PCDDocObject object.

See the following methods:

GrantRight  
HasRight  
RevokeRight

See the following tokens:

[%QS\\_DELETE](#)  
[%QS\\_EDIT](#)

## **%RECENTACTIVITYDATE**

## **%RECENTACTIVITYDATE**

This token allows to sort the items returned by a search according to the time they were most recently modified.

### **Syntax**

```
PCDRecentDoc. AddOrderByProperty( _  
    "%RECENTACTIVITYDATE", _  
    bl nTrueFa l se)
```

### **Parameters**

<code>%RECENTACTIVITYDATE</code>	The token identifier that indicates that the retrieved data should be sorted in order by the time each was created.
<code>bl nTrueFa l se</code>	A Boolean value that indicates whether the data should be sorted in ascending or descending order. A False value results in descending order. Anything else (including no value) results in ascending order.

### **Example**

```
' Create a recent doc object.  
Set pClient = CreateObject("PCDClient.PCDRecentDoc. 1")  
  
' Identify the form to use.  
pClient. SetSearchObject strFormName  
  
' Identify what data is to be returned.  
pClient. AddReturnProperty "DOCNAME"  
pClient. AddReturnProperty "AUTHOR"  
  
' Sort by modification date in ascending order.  
pClient. AddOrderByProperty("%RECENTACTIVITYDATE", Fa l se)  
  
' Execute the retrieval.  
pClient. Execute  
  
' Identify the number of items returned.  
IntRowCount = pClient. GetRowsFound
```

# %RECENTACTIVITYDATE

```
' Show data for the retrieved documents.  
While (lNgRow < lNgRowCount)  
    strDocName = pClient.GetReturnValue("DOCNAME")  
    strAuthor = pClient.GetReturnValue("AUTHOR")  
    strMessage = strAuthor & " wrote this document: " & strDocName  
    MsgBox(strMessage)  
Wend
```

## Related Items

See the PCDRecentDoc object.

See the AddOrderByProperty method.

See the [%RECENTACTIVITYDATE](#) token.

## %RECENTACTIVITYTIME

## %RECENTACTIVITYTIME

This token allows to sort the items returned by a search according to the time they were most recently modified.

### Syntax

```
PCDRecentDoc.AddOrderByProperty( _  
    "%RECENTACTIVITYTIME", _  
    blnTrueFalse)
```

### Parameters

%RECENTACTIVITYTIME	The token identifier that indicates that the retrieved data should be sorted in order by the time each was created.
blnTrueFalse	A Boolean value that indicates whether the data should be sorted in ascending or descending order. A False value results in descending order. Anything else (including no value) results in ascending order.

### Example

```
' Create a recent doc object.  
Set pClient = CreateObject("PCDClient.PCDRecentDoc.1")  
  
' Sort by modification date, most recent first.  
pClient.AddOrderByProperty("%RECENTACTIVITYDATE", False)  
  
' ... And then by modification time, earliest first.  
pClient.AddOrderByProperty("%RECENTACTIVITYTIME")
```

### Related Items

See the PCDRecentDoc object.

See the AddOrderByProperty method.

See the [%RECENTACTIVITYDATE](#) token.

## **%RELATED\_REMOTE\_LIBS**

## **%RELATED\_REMOTE\_LIBS**

This token can be used to specify that a search for related documents should retrieve from remote libraries only.

### **Syntax**

```
PCDSearch.AddSearchCriteria( _  
    "%RELATED_REMOTE_LIBS", _  
    strLibraryList)
```

### **Parameters**

<code>%RELATED_REMOTE_LIBS</code>	The token identifier that directs a search to retrieve only from remote libraries.
<code>strLibraryList</code>	A comma-delimited list of remote libraries that are to be included in the search.

### **Example**

```
' Create a search object.  
pRelated = CreateObject("PCDClient.PCDSearch")  
  
' Set the search type.  
pRelated.SetSearchObject("RelatedItemsSearch")  
  
' Set the DM security token.  
pRelated.SetDST(strDST)  
  
' Search in the following remote libraries.  
pRelated.AddSearchCriteria("%RELATED_REMOTE_LIBS", LibraryList)
```

### **Related Items**

See the PCDSearch object.

See the AddSearchCriteria method.

## **%REMOVE\_READ\_ONLY**

## **%REMOVE\_READ\_ONLY**

This token is used in conjunction with the %STATUS and %VERSION\_DOCUMENT tokens to remove the read-only setting from a document or a document version. See the %STATUS and %VERSION\_DOCUMENT token descriptions for further information.

### **Syntax**

```
PCDDocObj ect. SetProperty("%STATUS",  
                           "%REMOVE_READ_ONLY")
```

```
PCDDocObj ect. SetProperty("%VERSION_DOCUMENT",  
                           "%REMOVE_READ_ONLY")
```

### **Parameters**

%STATUS	The token that indicates that this command statement adjusts the status of the document.
%VERSION_DOCUMENT	The token that indicates that this command statement adjusts the document version settings.
%REMOVE_READ_ONLY	The token identifier that indicates that the DM system should remove the read-only setting from the specified document or document version.

### **Returns**

The RemoveReadonly method returns SUCCESS if the document was set so it is not read only. This allows the document to be modified. If the user does not have sufficient security rights to remove any read-only setting, then a PCD\_ERR\_INSUFFICIENT RIGHTS error will be returned.

### **Usage**

To make a document or document version read only, follow the following steps:

- 1 Create a PCDCI i ent. PCDDocObj ect.

## **%REMOVE\_READ\_ONLY**

- 2** Set the object type (form name) to whatever value is appropriate.
- 3** Set the DM security token (DST).
- 4** Set the %TARGET\_LIBRARY token equal to the library name.
- 5** Set the %OBJECT\_IDENTIFIER token equal to the document number of the document or document version that is to have its read-only setting removed.
- 6** Set the %STATUS token equal to `%REMOVE_READ_ONLY`, indicating that you want to remove the read-only setting for this document.
- 7** Execute the [Update](#) method.

### **Example**

See the [%STATUS example on page 472](#).

See the [%VERSION\\_DIRECTIVE example on page 500](#).

### **Related Items**

See the `PCDDocObject` object.

See the `SetProperty` method.

See the following tokens:

`%MAKE_READ_ONLY`  
`%STATUS`  
`%VERSION_DIRECTIVE`

# %REMOVE\_READ\_ONLY

## RemoveReadOnly Document Token

Use the RemoveReadOnly Document token to remove the read-only setting from a document. PH Note: Merge this text into the main discussion of the %REMOVE\_READ\_ONLY token.

### Returns

The RemoveReadOnly Document method returns SUCCESS if the document was set so it is not read only. This allows the document to be modified. If the user does not have sufficient security rights to remove any read-only setting, then a PCD\_ERR\_INSUFFICIENT\_PERMISSIONS error will be returned.

### Usage

To make a document version read only, follow the following steps:

- 1 Create a PCDCI object. PCDDocObject ect.
- 2 Set the object type to whatever value is appropriate.
- 3 Set the DM security token (DST).
- 4 Set the %TARGET\_LIBRARY token equal to the library name.
- 5 Set the %OBJECT\_IDENTERIFIER token equal to the document number of the document version that is to have its read-only setting removed.
- 6 Set the %STATUS token equal to `%REMOVE_READ_ONLY`, indicating that you want to remove the read-only setting for this document.
- 7 Execute the [Update](#) method.

### Related Items

See the [PCDDocObject](#) object.

See the following methods:

[%MAKE\\_READ\\_ONLY](#)  
[%REMOVE\\_READ\\_ONLY](#)

## **%RENDITION\_TYPE**

## **%RENDITION\_TYPE**

This token is used to check whether the DM Server can deliver documents in BINDER mode.

### **Syntax**

```
PCDGetDoc.AddSearchCriteria( _  
    "%RENDITION_TYPE", "BINDER")
```

### **Parameters**

<b>%RENDITION_TYPE</b>	The token identifier that queries to determine whether or not the DM Server can deliver documents in the specified mode.
<b>BINDER</b>	The key word that indicates BINDER mode is being specified.

### **Example**

```
' Create a PCDGetDoc object.  
pGetDoc = CreateObject("PCDCIent.PCDGetDoc.1")  
  
' Set the DM security token.  
pGetDoc.SetDST(strDST)  
  
' Set the search criteria to check for binder docs.  
pGetDoc.AddSearchCriteria("%RENDITION_TYPE", "BINDER")  
  
' Execute the search.  
pGetDoc.Execute  
  
' If there was an error, the DM Server will not be able to  
' deliver binder documents.  
If (pGetDoc.ErrNumber = -2147220814) Then binderrendition = False  
  
' Display the search results.  
If (binderrendition) Then  
    MsgBox("Documents can be delivered in BINDER mode.")  
Else  
    MsgBox("Documents cannot be delivered in BINDER mode.")  
End If  
  
' Release the memory used for the result set.
```

## **%RENDITION\_TYPE**

pGetDoc. ReleaseResults

### **Related Items**

See the `PCDGetDoc` object.

See the `AddSearchCriteria` method.

## %RIGHT8

## %RIGHT8

This token identifies whether or not a user has authority to assign a document or record to a file.

### Syntax

```
PCDDocObj ect. HasRi ght("%RI GHT9",  
                           _  
                           i ntAccessRi ghts)
```

### Parameters

%RI GHT9	The token that indicates the application should report whether or not the user has sufficient rights to assign a Document or Record to a File.
i ntAccessRi ghts	The user's access rights. The rights mask is an unsigned 32-bit integer.

### Usage

The HasRi ght method describes the access rights setting for this token. See its discussion of [Usage on page 240](#).

### Example

```
' Create a doc obj ect.  
pDocObj ect = CreateObj ect("PCDCI i ent. PCDDocObj ect. 1")  
  
' Check for errors.  
checkError(pDocObj ect, "ERROR_CREATESEARCH")  
  
' Set the user's access securi ty (DST), and the FRM fi le.  
pDocObj ect. SetDST(strDST)  
  
' Set the object type.  
pDocObj ect. SetObj ectType("cyd_defprof")  
  
' Set the library name and and document number/version.  
pDocObj ect. SetProperty("%TARGET_LI BRARY", library)  
pDocObj ect. SetProperty("%OBJECT_IDEN TI FIER", docnumber)  
pDocObj ect. SetProperty("%VERSI ON_ID", versi on)  
  
' Get the requested i nformation.  
pDocObj ect. Fetch()
```

# %RIGHT8

```
' Check for errors.  
checkError(pDocObj ect, "ERROR_DOCPROFILEDSP_RI GHTS")  
  
' Get the user's effective rights for this document object.  
Set intAccessRights = _  
    pDocObj ect.GetReturnProperty("%EFFECTIVE_RI GHTS")  
  
' Make sure user has rights to assign a document or record  
' to a File.  
If Not (pDocObj ect.HasRight("%RIGHT8", intAccessRights)) Then  
    ' The user does not have sufficient access rights to  
    ' assign a document or record to a file.  
End If
```

## Related Items

See the PCDDocObj ect object.

See the following methods:

GrantRi ght  
HasRi ght  
RevokeRi ght

See the following tokens:

%EFFECTIVE\_RI GHTS  
%PR\_ACCESS\_CONTROL  
%PR\_CONTENT\_COPY  
%PR\_CONTENT\_DELETE  
%PR\_CONTENT\_EDIT  
%PR\_CONTENT\_RETRIEVE  
%PR\_CONTENT\_VIEW  
%PR\_EDIT  
%RIGHT9

## %RIGHT9

## %RIGHT9

This token identifies whether or not a user has authority to assign a document or record to a file.

### Syntax

```
PCDDocObj ect. HasRi ght("%RI GHT9",  
    _  
    intAccessRi ghts)
```

### Parameters

%RI GHT9	The token that indicates the application should report whether or not the user has sufficient rights to assign a Document or Record to a File.
intAccessRi ghts	The user's access rights. The rights mask is an unsigned 32-bit integer.

### Usage

The HasRi ght method describes the access rights setting for this token. See its discussion of [Usage on page 240](#).

### Example

```
' Create a doc obj ect.  
pDocObj ect = CreateObj ect("PCDCI i ent. PCDDocObj ect. 1")  
  
' Check for errors.  
checkError(pDocObj ect, "ERROR_CREATESEARCH")  
  
' Set the DM securi ty token.  
pDocObj ect. SetDST(strDST)  
  
' Set the object type.  
pDocObj ect. SetObj ectType("cyd_defprof")  
  
' Set the library name and and document number/version.  
pDocObj ect. SetProperty("%TARGET_LI BRARY", library)  
pDocObj ect. SetProperty("%OBJECT_IDEN TI FIER", docnumber)  
pDocObj ect. SetProperty("%VERSI ON_ID", versi on)  
  
' Get the requested i nformati on.  
pDocObj ect. Fetch()
```

# %RIGHT9

```
' Check for errors.  
checkError(pDocObj ect, "ERROR_DOCPROFILEDSP_RI GHTS")  
  
' Get the user's effective rights for this document.  
Set intAccessRights =  
    pDocObj ect.GetReturnProperty("%EFFECTIVE_RI GHTS")  
  
' Make sure user has rights to view only published documents  
' and records.  
If Not (pDocObj ect.HasRight("%RIGHT9", intAccessRights)) Then  
    ' The user does not have sufficient access rights to view  
    ' only published documents.  
End If
```

## Related Items

See the PCDDocObj ect object.

See the following methods:

GrantRi ght  
HasRi ght  
RevokeRi ght

See the following tokens:

%EFFECTIVE\_RI GHTS  
%PR\_ACCESS\_CONTROL  
%PR\_CONTENT\_COPY  
%PR\_CONTENT\_DELETE  
%PR\_CONTENT\_EDIT  
%PR\_CONTENT\_RETRIEVE  
%PR\_CONTENT\_VIEW  
%PR\_EDIT  
%RIGHT8

# **%SCORE\_GRAPHIC**

## **%SCORE\_GRAPHIC**

This token is used when a full text search is performed. It expresses the calculated relevance of each returned item as a value from 1 to 5. Lower values have greater relevance.

### **Syntax**

```
PCDSearch.AddReturnProperty("%SCORE_GRAPHIC")
```

### **Parameters**

<code>%SCORE_GRAPHIC</code>	Return the search relevance as a numeric value that ranges from 1 to 5.
-----------------------------	---

### **Example**

```
' Create a search object.  
Set pClient = CreateObject("PCDClient.PCDSearch.1")  
  
' Return the search relevance as a value from 1 to 5  
pClient.AddReturnProperty("%SCORE_GRAPHIC")  
  
' Execute the search  
pClient.Execute()  
  
' Create a variable to hold the retrieved search value.  
Dim vValue As Variant  
  
' Retrieve the data.  
Set vValue = pClient.GetPropertyVal ue("%SCORE_GRAPHIC")  
  
' Process the relevance data.
```

### **Related Items**

See the `PCDSearch` object.

See the `AddReturnProperty` method.

See the following tokens:

```
%FT_CONFIDENCE  
%FT_FORMAT  
%FT_MARKER_LIST
```

## **%SCORE\_GRAPHIC**

%FT\_SCORE  
%FT\_TIMESTAMP  
%FT\_VCC\_LIST  
%FT\_VCC\_RULES  
%SCORE\_PERCENT

## **%SCORE\_PERCENT**

## **%SCORE\_PERCENT**

This token is used when a full text search is performed. It expresses the relevance of the search criteria to the document. The relevance score is expressed as a percentage.

### **Syntax**

```
PCDSearch.AddReturnProperty("%SCORE_PERCENT")
```

### **Parameters**

<code>%SCORE_PERCENT</code>	Express the relevance value returned by the search process as a percent.
-----------------------------	--

### **Example**

```
' Create a search object.  
Set pClient = CreateObject("PCDClient.PCDSearch.1")  
  
' Return the search relevance as a percentage.  
pClient.AddReturnProperty("%SCORE_PERCENT")  
  
' Execute the search.  
pClient.Execute()  
  
' Create a variable to hold the retrieved search value.  
Dim vValue As Variant  
  
' Retrieve the data.  
Set vValue = pClient.GetPropertyVal ue("%SCORE_PERCENT")  
  
' Process the data.
```

### **Related Items**

See the `PCDSearch` object.

See the `AddReturnProperty` method.

See the following tokens:

```
%FT_CONFIDENCE  
%FT_FORMAT  
%FT_MARKER_LIST
```

## **%SCORE\_PERCENT**

```
%FT_SCORE  
%FT_TIMESTAMP  
%FT_VCC_LIST  
%FT_VCC_RULES  
%SCORE_GRAPHIC
```

## %SEARCH

## %SEARCH

This token is used in conjunction with %FORM\_LIST\_TYPE to return all search forms available to the user.

### Syntax

```
PCDDocObj ect. SetProperty("%FORM_LIST_TYPE", _  
    "%SEARCH")
```

### Parameters

%FORM_LIST_TYPE	The token that indicates the current command line identifies the type of form that the list operation is to retrieve.
%SEARCH	The token identifier that specifies that the list of search forms available to the user should be returned.

### Example

```
' Create a doc object.  
Set pDocObj ect = CreateObject("PCDClient.PCDDocObject.1")  
  
' Set the DM security token.  
pDocObj ect. SetDST(strDST)  
  
' Set the object (form) type.  
pDocObj ect. SetObjectType("FormsList")  
  
' Set the library.  
pDocObj ect. SetProperty("%TARGET_LIBRARY", strMyLib)  
  
' Specify the type of forms that this search should list.  
pDocObj ect. SetProperty("%FORM_LIST_TYPE", "%SEARCH")  
  
' Fetch the list of forms.  
pDocObj ect. Fetch()
```

### Related Items

See the PCDDocObject object.

See the SetProperty method.

## %SEARCH

See the %FORM\_LIST\_TYPE token.

## %SECURITY

## %SECURITY

This token is used to retrieve the user's access rights for the specified document(s). Each user may be granted different privileges (for example, allowed to view, not allowed to copy, etc.)

### Syntax

```
PCDSearch. AddReturnProperty("%SECURITY")
```

### Parameters

%SECURITY

The token identifier that the current user's security mask (access privileges) are to be returned for the specified document(s).

### Example

```
' Create a search object.  
Set pDoc = CreateObject("PCDClient.PCDSearch.1")  
  
' Set the DM Security Token.  
pDoc.SetDST(DST)  
  
' Indicate that the search should return security properties.  
pDoc.AddReturnProperty("%SECURITY")  
  
' Execute the search.  
pDoc.Execute
```

### Related Items

See the PCDSearch object.

See the AddReturnProperty method.

## %STATUS

## %STATUS

This token is used to change the status of a document. The document status can be set to one of the following:

- Lock the document, using the %LOCK token.
- Lock the document and check it out to the specified user, by use of the %LOCK\_FOR\_CHECKOUT token.
- Make the document read only using the %MAKE\_READ\_ONLY token.
- Remove the read-only setting from the document, using the %REMOVE\_READ\_ONLY token.
- Remove the read only status from the document, using the %UNLOCK token.

### Syntax

```
PCDDocObj ect. SetProperty("%STATUS", strSetting)
```

### Parameters d

%STATUS	The token that indicates that this command statement adjusts the status of the document.
strSetting	A variable that resolves to one of the settings that the %STATUS token supports.

### Example

```
' Create a doc object.  
pDocObj ect = CreateObj ect("PCDCI ent. PCDDocObj ect. 1")  
  
' Set the DM security token.  
pDocObj ect. SetDST( myDST )  
  
' Set the object (Form) type.  
pDocObj ect. SetObj ectType("cyd_defprof")  
  
' Set the object identifier.
```

# %STATUS

```
pDocObj ect. SetProperty( "%OBJECT_IDENTIFIER", strDocNum )  
  
' Set the status.  
pDocObj ect. SetProperty( "%STATUS", "%REMOVE_READ_ONLY" )  
  
' Perform the update.  
pDocObj ect. Update
```

## Related Items

See the PCDDocObj ect object.

See the SetProperty method.

See the following tokens:

%LOCK  
%LOCK\_FOR\_CHECKOUT  
%MAKE\_READ\_ONLY  
%REMOVE\_READ\_ONLY  
%UNLOCK

## **%TARGET\_LIBRARY**

## **%TARGET\_LIBRARY**

This token specifies the library to use for various actions (such as searching or creating documents).

### **Syntax**

```
PCDDocObj ect. SetProperty("%TARGET_LIBRARY", _  
    strLi bName)
```

### **Parameters**

%TARGET_LIBRARY	The token identifier that indicates that the target library is being set.
strLi bName	The name of the library.

### **Usage**

If no target library is specified, the default value is the current library.

### **Example**

```
' Create a doc object.  
pDocObj ect = CreateObj ect("PCDCI i ent. PCDDocObj ect. 1")  
  
' Set the DM Security Token.  
pDocObj ect. SetDST( strDST )  
  
' Use the cyd_defprof form to set properties.  
pDocObj ect. SetObj ectType("cyd_defprof")  
  
' Set the target library.  
pDocObj ect. SetProperty("%TARGET_LIBRARY", library)
```

### **Related Items**

See the `PCDDocObj ect` object.

See the `SetProperty` method.

# %TITLE

## %TITLE

The Execute method that PCDLookup supports returns both data and metadata about the object specified in the lookup operation. The %TITLE token is used to retrieve the title in the lookup list box for this column.

### Syntax

```
PCDLookup.GetMetaPropertyValue("%TITLE")
```

### Parameters

%TITLE	The token identifier used to request the title in the lookup list box for this property.
--------	--

### Example

```
' Create the object.  
Set pClient = CreateObject("PCDClient.PCDLookup")  
  
' Set the DM security token.  
pClient.SetDST(myDST)  
  
' Set the form.  
pClient.SetSearchObject("cyd_defprof")  
  
' Set the lookup ID.  
pClient.SetLookupID("DEPL_PACKAGES")  
  
' Set the target property.  
pClient.SetTargetProperty("PACKAGE_ID")  
  
' Add the search library.  
pClient.AddSearchLib(strMyLib)  
  
' Execute the search.  
pClient.Execute  
  
' Retrieve the title.  
Set strTitle = pClient.GetMetaPropertyValue("%TITLE")
```

# %TITLE

## Related Items

See the `PCDLookup` object.

See the `GetMetaPropertyValue` method.

See the following tokens:

`%DATA`  
`%PROPERTYNAME`  
`%PROPERTYTYPE`  
`%VISIBLE`

## **%TRUSTEE\_ID**

## **%TRUSTEE\_ID**

The %TRUSTEE\_ID token, together with the %TRUSTEE\_RIGHTS and %TRUSTEE\_TYPE tokens, allows trustee settings for a document to be set or modified. Users who can modify a document object are called trustees.

### Syntax

```
PCDPropertyList.AddProperty("%TRUSTEE_ID", _  
    vntUserName)
```

### Parameters

%TRUSTEE_ID	The token identifier used to indicate that the name of the trustee is being specified.
vntUserName	The name of the Trustee, as shown in the USER_ID column of the PEOPLE table.

### Example

```
' Create a property list object/  
Set pPropList = CreateObject("PCDClient.PCDPropertyList")  
  
' Request the trustee.  
pPropList.AddProperty("%TRUSTEE_ID", vntName)  
  
' Request the type of trustee.  
pPropList.AddProperty("%TRUSTEE_TYPE", vntType)  
  
' Request the trustee permissions.  
pPropList.AddProperty("%TRUSTEE_RIGHTS", vntRights)
```

### Related Items

See the PCDPropertyList object.

See the AddProperty method.

See the following tokens:

[%TRUSTEE\\_RIGHTS](#)

**%TRUSTEE\_ID**

**%TRUSTEE\_TYPE**

## **%TRUSTEE RIGHTS**

## **%TRUSTEE RIGHTS**

The %TRUSTEE RIGHTS token, together with the %TRUSTEE\_ID and %TRUSTEE\_TYPE tokens, allows trustee settings for a document to be set or modified. Users who can modify a document object are called trustees.

### **Syntax**

```
PCDPropertyList.AddProperty("%TRUSTEE RIGHTS", _  
    vntRights)
```

### **Parameters**

%TRUSTEE RIGHTS	The token identifier used to indicate that the rights of the trustee are being identified.
vntRights	The rights of this user for this document object, as shown in the SECURITY table.

### **Usage**

The rights specified by this setting can apply to a person, a group, or an ORGANIZATION unit, as shown in the ORGANIZATION database table.

### **Example**

```
' Create a property list object.  
Set pPropList = CreateObject("PCDClient.PCDPropertyList")  
  
' Request trustee information.  
pPropList.AddProperty("%TRUSTEE_ID", vntName)  
  
' Request the trustee type.  
pPropList.AddProperty("%TRUSTEE_TYPE", vntType)  
  
' Request the trustee permissions.  
pPropList.AddProperty("%TRUSTEE RIGHTS", vntRights)
```

### **Related Items**

See the PCDPropertyList object.

## **%TRUSTEE\_RIGHTS**

See the `AddProperty` method.

See the following tokens:

`%TRUSTEE_ID`  
`%TRUSTEE_TYPE`

## **%TRUSTEE\_TYPE**

## **%TRUSTEE\_TYPE**

The `%TRUSTEE_TYPE` token, together with the `%TRUSTEE_ID` and `%TRUSTEE RIGHTS` tokens, allows trustee settings for a document to be set or modified. Users who can modify a document object are called trustees.

### **Syntax**

```
PCDPropertyList.AddProperty("%TRUSTEE_TYPE",  
    vntUserType)
```

### **Parameters**

<code>%TRUSTEE_TYPE</code>	The token identifier used to indicate that the type of trustee is being identified.
<code>vntUserType</code>	The type of Trustee.

### **Usage**

The user type can be set to any of the following values:

0	Unknown User Type
1	Group
2	Person

### **Example**

```
' Create a property list object.  
Set pPropList = CreateObject("PCDCIent.PCDPropertyList")  
  
' Identify the trustee.  
pPropList.AddProperty("%TRUSTEE_ID", vntName)  
  
' Identify the trustee type.  
pPropList.AddProperty("%TRUSTEE_TYPE", vntType)  
  
' Identify the trustee permissions.  
pPropList.AddProperty("%TRUSTEE RIGHTS", vntRights)
```

## **%TRUSTEE\_TYPE**

### **Related Items**

See the `PCDPropertyList` object.

See the `AddProperty` method.

See the following tokens:

`%TRUSTEE_ID`  
`%TRUSTEE_RIGHTS`

## **%TRUSTEES\_ADD**

## **%TRUSTEES\_ADD**

Use this token to add new entities (people, groups, etc.) to the current trustee list.

### **Syntax**

```
PCDDocObj ect. SetProperty("%TRUSTEES_UPDATE",  
                           "%TRUSTEES_ADD")
```

### **Parameters**

%TRUSTEES_UPDATE	The token identifier that indicates trustee information is about to be updated.
%TRUSTEES_ADD	The token identifier that indicates one or more trustees is to be added to the current list of trustees.

### **Example**

```
' Create a document object.  
Set pDocObj ect = CreateObject("PCDClient.PCDDocObj ect.1")  
  
' Set the DM security token.  
pDocObj ect. SetDST(strDST)  
  
' Set the object type. Data on the MassProfile Update form shows  
' the updated trustee settings.  
pDocObj ect. SetObjectType("MassProfileUpdate")  
  
' Set the update action so that the specified entities will be  
' added to the current trustee list.  
pDocObj ect. SetProperty("%TRUSTEES_UPDATE", "%TRUSTEES_ADD")  
  
' Perform the update.  
pDocObj ect. Update
```

### **Related Items**

See the PCDDocObj ect object.

See the SetProperty method.

See the following tokens:

## **%TRUSTEES\_ADD**

**%TRUSTEES\_REMOVE  
%TRUSTEES\_SET  
%TRUSTEES\_UPDATE**

## **%TRUSTEES\_REMOVE**

## **%TRUSTEES\_REMOVE**

Use this token to delete one or more trustees from the current list of trustees for the specified object.

### **Syntax**

```
PCDDocObj ect. SetProperty("%TRUSTEES_UPDATE",  
                           "%TRUSTEES_REMOVE")
```

### **Parameters**

%TRUSTEES_UPDATE	The token identifier that indicates trustee information is about to be updated.
%TRUSTEES_REMOVE	The token identifier that indicates one or more of the trustees is to be removed from the trustee list.

### **Example**

```
' Create a document object.  
Set pDocObj ect = CreateObject("PCDClient.PCDDocObj ect.1")  
  
' Set the DM security token.  
pDocObj ect. SetDST(strDST)  
  
' Set the object type. Data on the MassProfile Update form shows  
' the updated trustee settings.  
pDocObj ect. SetObjectType("MassProfileUpdate")  
  
' Set the update to remove the specified trustees.  
pDocObj ect. SetProperty("%TRUSTEES_UPDATE", "%TRUSTEES_REMOVE")  
  
' Perform the update.  
pDocObj ect. Update
```

### **Related Items**

See the PCDDocObj ect object.

See the SetProperty method.

See the following tokens:

## **%TRUSTEES\_REMOVE**

`%TRUSTEES_ADD  
%TRUSTEES_SET  
%TRUSTEES_UPDATE`

## **%TRUSTEES\_SET**

## **%TRUSTEES\_SET**

Use this token to set trustee information.

### **Syntax**

```
PCDDocObj ect. SetProperty("%TRUSTEES_UPDATE",  
                           "%TRUSTEES_SET")
```

### **Parameters**

%TRUSTEES_UPDATE	The token identifier that indicates trustee information is about to be updated.
%TRUSTEES_SET	The token identifier that indicates the rights of one or more of the trustees has changed and should be saved.

### **Example**

```
' Create a document object.  
Set pDocObj ect = CreateObj ect("PCDClient.PCDDocObj ect.1")  
  
' Set the DM security token.  
pDocObj ect. SetDST(strDST)  
  
' Set the object type. Data on the MassProfile Update form shows  
' the updated trustee settings.  
pDocObj ect. SetObjectType("MassProfileUpdate")  
  
' Set the update action.  
pDocObj ect. SetProperty("%TRUSTEES_UPDATE", "%TRUSTEES_SET")  
  
' Perform the update.  
pDocObj ect. Update
```

### **Related Items**

See the PCDDocObj ect object.

See the SetProperty method.

See the following tokens:

[%TRUSTEES\\_ADD](#)  
[%TRUSTEES\\_REMOVE](#)

**%TRUSTEES\_SET**

**%TRUSTEES\_UPDATE**

## **%TRUSTEES\_UPDATE**

## **%TRUSTEES\_UPDATE**

Use this token to set trustee information.

### **Syntax**

```
PCDDocObj ect. SetProperty("%TRUSTEES_UPDATE", _  
    vntAction)
```

### **Parameters**

<code>%TRUSTEES_UPDATE</code>	The token identifier that indicates trustee information is about to be updated.
<code>vntAction</code>	One of the three trustee actions supported by the <code>%TRUSTEES_UPDATE</code> token.

### **Usage**

The `%TRUSTEES_UPDATE` token supports the following actions:

<code>%TRUSTEES_ADD</code>	Add one or more trustees to the current list of trustees.
<code>%TRUSTEES_REMOVE</code>	Remove one or more trustees from the current list of trustees.
<code>%TRUSTEES_SET</code>	Adjust the access rights of one or more of the current trustees.

### **Example**

```
' Create a document object.  
Set pDocObject = CreateObject("PCDClient.PCDDocObject.1")  
  
' Set the DM security token.  
pDocObject.SetDST(strDST)  
  
' Set the object type. Data on the MassProfile Update form shows  
' the updated trustee settings.  
pDocObject.SetObjectType("MassProfileUpdate")  
  
' Set the update action.  
pDocObject SetProperty("%TRUSTEES_UPDATE", "%TRUSTEES_SET")
```

## **%TRUSTEES\_UPDATE**

' Perform the update.  
pDocObj ect. Update

### **Related Items**

See the PCDDocObj ect object.

See the SetProperty method.

See the following tokens:

[%TRUSTEES\\_ADD](#)  
[%TRUSTEES\\_REMOVE](#)  
[%TRUSTEES\\_SET](#)

## %UNLOCK

## %UNLOCK

This token is used in conjunction with the %STATUS token to unlock a document. See the “%STATUS” token on page 472 for further information.

### Syntax

```
PCDDocObj ect. SetProperty("%STATUS", "%UNLOCK")
```

### Parameters

%STATUS	The token that indicates that this command statement adjusts the status of the document.
%UNLOCK	The token identifier that indicates that the DM system should unlock the document.

### Example

See the %STATUS example on page 472.

### Related Items

See the PCDDocObj ect object.

See the SetProperty method.

See the following tokens:

%LOCK  
%LOCK\_FOR\_CHECKOUT  
%MAKE\_READ\_ONLY  
%REMOVE\_READ\_ONLY  
%STATUS

## **%UNPUBLISH\_VERSION**

## **%UNPUBLISH\_VERSION**

This token is used with the `%VERSION_DIRECTIVE` token to reset a previously published document version so it is unpublished. See the “[%VERSION\\_DIRECTIVE](#)” token on page 500 for further information.

### **Syntax**

```
PCDDocObj ect. SetProperty(“%VERSION_DIRECTIVE”,  
“%UNPUBLISH_VERSION”)
```

### **Parameters**

<code>%VERSION_DIRECTIVE</code>	The token that indicates that this command statement adjusts the document version settings.
<code>%UNPUBLISH_VERSION</code>	The token identifier that indicates that a document version is being reset so that it no longer indicates that it has been published.

### **Returns**

The UnpublishVersion token returns `SUCCESS` if the document version had its “published” status removed. If the user does not have sufficient security rights to remove the published status of a document version, then a `PCD_ERR_INSUFFICIENT RIGHTS` error will be returned.

### **Usage**

To remove the published status of a document version, follow the following steps:

- 1** Create a `PCDCI i ent. PCDDocObj ect.`
- 2** Set the object type to whatever value is appropriate.
- 3** Set the DM security token (DST).
- 4** Set the `%TARGET_LIBRARY` token equal to the library name.

## %UNPUBLISH\_VERSION

- 5 Set the %OBJECT\_IDENTIFIER token equal to the document number of the document version that is to have its published status removed.
- 6 Set the %VERSION\_ID token equal to the version number of the document version that is to have its published status removed.
- 7 Set the %VERSION\_DIRECTIVE token equal to %UNPUBLISH\_VERSION, indicating that you want to remove the published status of the specified document version.
- 8 Execute the [Update](#) method.

### Example

See the %VERSION\_DIRECTIVE [example on page 500](#).

### Related Items

See the [PCDDocObject](#) object.

See the [SetProperty](#) method.

See the following tokens:

%PUBLISH\_VERSION  
%VERSION\_DIRECTIVE

## **%USER\_ID**

## **%USER\_ID**

This token is used to set the user ID property for various actions.

### **Syntax**

```
PCDDocObj ect. SetProperty("%USER_ID",  
                           "vntUserName")
```

### **Parameters**

<b>%USER_ID</b>	The token identifier that indicates the user name is being identified.
<b>vntUserName</b>	The name of the user.

### **Example**

```
' Create a doc object.  
Set p0bj = CreateObj ect("PCDCI ent. PCDDocObj ect")  
  
' Set the DM security token.  
p0bj . SetDST(strDST)  
  
' Set the object (form) type.  
p0bj . SetObj ectType("ImptDocRetri eval Form")  
  
' Set the user ID property.  
p0bj . SetProperty("%USER_ID", vntGuest)  
  
' Fetch documents authored by the user specified on the form.  
p0bj . Fetch
```

### **Related Items**

See the **PCDDocObject** object.

See the **SetProperty** method.

## **%VERIFY\_ONLY**

## **%VERIFY\_ONLY**

This token allows you to verify that the attributes are correct for the document. This is used primarily when a document is first created. You might want to do this to make sure no errors are returned.

### **Syntax**

```
PCDDocObj ect. SetProperty("%VERIFY_ONLY", _  
    strYesNo)
```

### **Parameters**

<code>%VERIFY_ONLY</code>	The token identifier that indicates a check of document attributes is going to take place.
<code>strYesNo</code>	A string that resolves to either <code>%YES</code> or <code>%NO</code> .

### **Usage**

The `%VERIFY_ONLY` token supports the following tokens:

<code>%YES</code>	Yes, verify the parameters only.
<code>%NO</code>	No, do not <i>just</i> verify parameters. Actually create the object in addition to verifying the parameters.

### **Example**

```
' Create a document object.  
Set p0bj = CreateObject("PCDCIent.PCDDocObj ect")  
  
' Set the DM security token.  
p0bj . SetDST(strDST)  
  
' Validate the parameters, but do not create the object.  
p0bj . SetProperty("%VERIFY_ONLY", "%YES")  
  
' Do the verification now.  
p0bj . Create
```

## **%VERIFY\_ONLY**

### **Related Items**

See the PCDDocObject object.

See the SetProperty method.

## **%VERSION\_AUTHOR**

## **%VERSION\_AUTHOR**

This token allows to specify the author of the document version.

### **Syntax**

```
PCDDocObject SetProperty("%VERSION_AUTHOR", _  
    vntAuthorName)
```

### **Parameters**

<code>%VERSION_AUTHOR</code>	The token identifier that indicates the author is being specified.
<code>vntAuthorName</code>	The USER_ID of the document version author.

### **Example**

```
' Create a document object.  
Set p0bj = CreateObject("PCDCObject.PCDDocObject")  
  
' Set the DM security token.  
p0bj.SetDST(strDST)  
  
' Set the author name.  
pDocObject SetProperty("%VERSION_AUTHOR", "ROBERT_X")  
  
' Set the object (form) type.  
pDocObject.SetObjectType("DEF_PROF")  
  
' Fetch the data.  
pDocObject.Fetch
```

### **Related Items**

See the PCDDocObject object.

See the SetProperty method.

## **%VERSION\_COMMENT**

## **%VERSION\_COMMENT**

This token allows you to specify the comment for the document version.

### **Syntax**

```
PCDDocObject SetProperty("%VERSION_COMMENT", _  
    strCommentText)
```

### **Parameters**

<code>%VERSION_COMMENT</code>	The token that indicates the comment for this version is being set.
<code>strCommentText</code>	A string variable (or literal text in double quotes) that contains the comment for the specified document version.

### **Usage**

The version comment text is inserted into the `COMMENTS` column of the `VERSIONS` table for the specified document version.

### **Example**

```
' Create a document object.  
Set pObj = CreateObject("PCDClient.PCDDocObject")  
  
' Set the DM security token.  
pObj.SetDST(strDST)  
  
' Set the object (form) type.  
pObj.SetObjectType("NewDocProfileForm")  
  
' Set the comment for this document version.  
pObj SetProperty("%VERSION_COMMENT", _  
    "Operating Plan, Final Version")  
  
' Perform the update.  
pObj.Update
```

### **Related Items**

See the `PCDDocObject` object.

## **%VERSION\_COMMENT**

See the `SetProperty` method.

## **%VERSION\_DIRECTIVE**

## **%VERSION\_DIRECTIVE**

This token indicates that any of several supported actions is to affect the specified document version.

### **Syntax**

```
PCDDocObj ect. SetProperty("%VERSION_DIRECTIVE",  
                           vntVersionID)
```

### **Parameters**

<code>%VERSION_DIRECTIVE</code>	The token identifier that indicates a version directive is being set.
<code>strVersionAction</code>	Any of the actions that the <code>%VERSION_DIRECTIVE</code> token supports.

### **Usage**

The `%VERSION_DIRECTIVE` token supports any of the following:

<b>Token Identifier</b>	<b>Description</b>
<code>%ADD_ATTACHMENT</code>	Creates an attachment.
<code>%MAKE_READ_ONLY</code>	Make this version read only.
<code>%PCD_DELETEVERSION</code>	Delete the specified version.
<code>%PCD_NEW_VERSION</code>	Create a new version.
<code>%PCD_NEWSUBVERSION</code>	Create a new sub-version.
<code>%PCD_UPDATE_VERSION</code>	Update modifications made to a version.
<code>%PUBLISH_VERSION</code>	Publish a version.
<code>%REMOVE_READ_ONLY</code>	Set the version so it is not read only.
<code>%UNPUBLISH_VERSION</code>	Set the version so it is not published.

### **Example**

```
' Create a document object.  
pDelObj ect = CreateObj ect("PCDCI ent. PCDDocObj ect. 1")
```

```
' Set the DM security token.
```

# **%VERSION\_DIRECTIVE**

```
pDelObj ect. SetDST( strDST )  
  
' Set the version ID  
pDelObj ect. SetProperty( "%VERS ION_ID", versi on)  
  
' Delete the document version.  
pDelObj ect. SetProperty( "%VERS ION_DIRECTI VE", _  
"%PCD_DELETE_VERSI ON"
```

## **Related Items**

See the `PCDDocObj ect` object.

See the `SetProperty` method.

## **%VERSION\_ID**

## **%VERSION\_ID**

This token allows a specific version ID to be set for the document.

### **Syntax**

```
PCDDocObj ect. GetReturnProperty("%VERS I ON_I D")  
PCDDocObj ect. SetProperty("%VERS I ON_I D", _  
                           strVersi onI D)  
PCDDocObj ect. SetProperty("%VERS I ON_I D", _  
                           "%VERS I ON_TO_I NDEX")  
PCDGetDoc. AddSearchCri teria("%VERS I ON_I D", _  
                           strVersi onI D)  
PCDPutDoc. AddSearchCri teria("%VERS I ON_I D", _  
                           strVersi onI D)
```

### **Parameters**

<code>%VERS I ON_I D</code>	The token identifier that indicates an action is to involve a version.
<code>strVersi onI D</code>	The version identifier that is to be used in the search or update.
<code>%VERS I ON_TO_I NDEX</code>	After a document is returned by a search operation, this token specifies the version of a document that is to be referenced in a subsequent content search.

### **Example**

```
' Create a search object.  
Set pVer = CreateObj ect("PCDCI ent. PCDSearch. 1")  
  
' Set the DM security token.  
pVer. SetDST(strDST)  
  
' Set the object (form) type.  
pVer. SetObj ectType("DEF_PROF")  
  
' Set the version identifier.  
pVer. AddSearchCri teria("%VERS I ON_I D", versi on[i])
```

# %VERSION\_ID

```
' Execute the search.  
pVer.Execute
```

## Related Items

See the following objects:

- PCDDocObject
- PCDGetDoc
- PCDPutDoc

See the following methods:

- AddSearchCriteria
- GetProperty
- SetProperty

See the %VERSION\_TO\_INDEX token.

## **%VERSION\_LABEL**

## **%VERSION\_LABEL**

This token is used to retrieve the preview content.

### Syntax

```
PCDGetDoc.AddSearchCriteria("%VERSION_LABEL",  
                            "PR1")
```

### Parameters

%VERSION_LABEL	The token identifier that specifies that the version's preview content is to be returned.
PR1	A required parameter that is used in the search.

### Example

```
' Create a PCDGetDoc object.  
pGetDoc = CreateObject("PCDCClient.PCDGetDoc.1")  
  
' Set the DM security token.  
pGetDoc.SetDST(strDST)  
  
' Get the preview content.  
pGetDoc.AddSearchCriteria("%VERSION_LABEL", "PR1")  
  
' Retrieve the document version.  
pGetDoc.Execute()
```

### Related Items

See the `PCDGetDoc` object.

See the `AddSearchCriteria` method.

See the `%CONTENT` token.

## **%VERSION\_TO\_INDEX**

## **%VERSION\_TO\_INDEX**

This token identifies a version identified in a search whose content is now to be retrieved. In this circumstance, it can be less complex to identify the version by referencing its location in the data set returned by the initial search operation than in trying to identify it by its document ID number.

### **Syntax**

```
PCDDocObj ect. SetProperty("%VERS I ON_I D",  
                           "%VERS I ON_TO_I NDEX")
```

### **Parameters**

`%VERS I ON_I D`

The token identifier that indicates an action is to involve a version.

`%VERS I ON_TO_I NDEX`

After a document is returned by a search operation, this token specifies the version of a document that is to be referenced in a subsequent content search.

### **Related Items**

See the `PCDDocObj ect` object.

See the `SetProperty` method.

See the `%VERS I ON_I D` token.

## **%VERSION\_TYPIST**

## **%VERSION\_TYPIST**

This token is used to identify the typist for the specified document version.

### **Syntax**

```
PCDDocObj ect. SetProperty("%VERSION_TYPIST",  
                           strTypistName)
```

### **Parameters**

**%VERSION\_TYPIST**

This token indicates that the typist associated with the specified document version is being specified.

**strTypistName**

The name of the typist, as shown in the **USER\_ID** column of the **PEOPLE** table in the DM database.

### **Example**

```
' Create a doc object.  
Set pObj = CreateObj ect("PCDClient.PCDDocObj ect")  
  
' Set the DM security token.  
pObj .SetDST(strDST)  
  
' Set the typist.  
pDocObj ect SetProperty("%VERSION_TYPIST", "j_SMI TH")
```

### **Related Items**

See the **PCDDocObj ect** object.

See the **SetProperty** method.

## %VISIBLE

## %VISIBLE

The Execute method that PCDLookup supports returns both data and metadata. The %VISIBLE token returns a Boolean value that indicates whether or not this lookup column should be displayed to the user.

### Syntax

```
PCDLookup. GetMetaPropertyValue("%VISIBLE")
```

### Parameters

%VISIBLE

The token identifier used to request the flag setting that indicates whether or not this lookup column should be displayed to the user.

### Example

```
' Create the object.  
Set pClient = CreateObject("PCDClient.PCDLookup")  
  
' Set the DM security token.  
pClient.SetDST(strDST)  
  
' Set the form  
pClient.SetSearchObject("cyd_defprof")  
  
' Set the lookup ID.  
pClient.SetLookupID("DEPL_PACKAGES")  
  
' Set the target property.  
pClient.SetTargetProperty("PACKAGE_ID")  
  
' Add the search library.  
pClient.AddSearchLib(strLibName)  
  
' Execute the search.  
pClient.Execute()  
  
' Get the visible indicator.  
Set strIndicator = pClient.GetMetaPropertyValue("%VISIBLE")
```

# %VISIBLE

## Related Items

See the PCDLookup object.

See the GetMetaPropertyValue method.

See the following tokens:

%DATA  
%PROPERTYNAME  
%PROPERTYTYPE  
%TITLE