

Customize



*Rapidocs Data Server
Developers Guide*

Version **2**

Document Information

Printed	13 November 2001
Rapidocs Version	2.0
Author	Stuart Robinson
Copyrights	© Epoch Software Group Ltd, 2001
Contact Details:	Rapidocs (Sales) Ltd Unit 2, The Technology Park Colindeep Lane London, NW9 6BX
Telephone	+44 20 8931 3066
Web Site	http://www.rapidocs.co.uk http://www.rapidocs.com
Email	info@rapidocs.co.uk

Table of Contents

Rapidocs DataServer	7
Introduction	7
DataServer Functionality	7
RapidocsDocCOM Functionality.....	7
RapidocsX	8
Rapidocs Document Basics	8
The Rapidocs RAP File	8
E-commerce / Unrestricted Templates	9
Rapidocs Red Templates	9
Rapidocs Blue Documents	9
Installing DataServer and RapidocsDocCOM.....	9
Manually Registering the COM objects	9
De-registering DataServer and RapidocsDocCOM.....	10
Visual Basic Environment for Data Server	11
Preparing the VBA Environment	11
Defining and Opening the Rapidocs Document	11
Opening a Rapidocs Document from a File.....	12
Opening a Rapidocs Document from a Data Store.....	12
Searching for Documents According to Certain Criteria.....	12
Retrieving Answer Information from a Rapidocs Document..	13
Writing New Answers to the Document	14
Loop Controlled Answers.....	14
Illegal Values.....	15
Saving and Closing the Document	15
Deleting a Document	15
Password Protected Documents.....	16
Transferring a Document	16
Transfer Methods.....	16
Transfer Methods Summary	16

Transfer Examples	18
Visual Basic Examples.....	18
Example 1 – Listing the Answers.....	18
Example 2 – Writing New Answers.....	19
Example 3 – Searching and Manipulating Documents.....	19
Delphi Environment for Data Server	20
Preparing the Delphi Environment.....	20
Setting Up the DataServer and RapidocsDocCOM Object Variables	20
Declaring the Data Server Object Variable.....	20
Creating an Instance of DataServer	21
Declaring the RapidocsDocCOM Object Variable	21
Creating an Instance of RapidocsDocCOM.....	21
Opening a Rapidocs Document from a File.....	21
Opening a Rapidocs File from a Data Store.....	22
Searching for Documents According to Certain Criteria.....	22
Retrieving Answer Information from a Rapidocs Document.	23
Writing New Answers to the Document.....	24
Loop Controlled Answers.....	24
Illegal Values	24
Saving and Closing the Document.....	25
Deleting a Document.....	25
Password Protected Documents	25
Transferring a Document.....	26
Transfer Examples	26
Delphi Examples.....	26
Example 1 – Listing the Answers.....	26
Example 2 – Writing New Answers.....	27
Example 3 – Searching and Manipulating Documents.....	27
Header Properties	29
Using the Header Properties.....	29
Visual Basic Environment	29
Retrieving Information from the Document Header	29
Writing to the Document Header.....	30
Example Procedure.....	30
Delphi Environment.....	30
Retrieving Information from the Document Header	30
Writing to the Document Header.....	31
Example Procedure.....	31

Using RapidocsX	32
Introduction	32
Client Machine	32
Server Machine	33
Implementing RapidocsX	33
Example of Opening a Document from a Store.....	34
Example of Opening a Document from a File.....	35
Proxy Server	35
Mechanism for Selecting Documents	35
Event Handlers	36
Example	36
RapidocsX Parameters.....	37
Appendix A: Rapidocs DataServer Object Properties and Methods.....	39
Document.....	39
Answer Properties	41
Manually Setting the Licence Path	42
Appendix B: Rapidocs Document Header Object Properties....	43
Properties.....	43
Appendix C: DataServer Document Version Control.....	47
Appendix D: RapidocsDocCOM Object Properties and Methods	49
Searches Using RapidocsDocCOM.....	50
Field Names	50
AddDateCriterion	53
AddDocColorCriterion	53
AddStringCriterion.....	53
Using RapidocsDocCOM with ASP or VBScript.....	54
Error Codes	54
Appendix E: RapidocsX Client-Server Overview	56
Using RapidocsX in Netscape	58
Introduction	58

Implementing RapidocsN.....	58
Example of Opening a Document from a Store.....	59
Event Handlers.....	61

Rapidocs DataServer

Introduction

Rapidocs DataServer is a COM object toolbox which allows external applications to access the internal information held in Rapidocs documents. It is designed as an In-Process COM server that can be used by other applications within the context of their own process. Essentially there are two components to the tool box represented by two separate DLL files: DataServer and RapidocsDocCOM. A third COM object is also documented here, RapidocsX, which is often used in conjunction with the DataServer COM objects in order to display Rapidocs documents.

DataServer Functionality

Applications can be used to open 'red' or 'blue' Rapidocs documents, initialize or modify answers in the document, and save 'blue' documents. Typically documents will be saved and stored in a Rapidocs document store database.

External applications can be programming languages (Delphi, C, C++, Visual Basic, Java) or applications (Databases such as Access or Paradox, spreadsheets such as Excel or Quattro, word processors such as Microsoft Word or WordPerfect). Any Windows environment that supports ActiveX can host DataServer.

RapidocsDocCOM Functionality

This COM object complements DataServer in that it enables applications to search a Rapidocs document store and retrieve a list of documents according to a given set of criteria (for example all documents created between two dates). In most applications, both DataServer and RapidocsDocCOM will need to be used in parallel.

RapidocsDocCOM can be implemented in a similar way to DataServer with the same applications or programming languages listed above for DataServer.

RapidocsX

This COM object is an ActiveX control that enables a Rapidocs document to be opened and assembled within a web-browser (usually Internet Explorer) or any programming environment that can accept ActiveX controls, eg Visual Basic, Visual C++, Delphi etc. The document can then be saved back to the document store from where it came or a different store if required.

Before using these COM objects, it is useful to have some basic understanding of document types in Rapidocs as well as the software architecture.

Rapidocs Document Basics

The Rapidocs RAP File

The Rapidocs RAP File format is an unpublished proprietary format.

For reasons of security every RAP file is encrypted using a 128-bit key that can be further enhanced with user-specific passwords if required.

RAP files are highly compressed, with even complex contracts producing file sizes of as little as 15Kb.

Each RAP file is a self-contained object containing everything needed for the document. Every permutation of text used, the logic that controls the assembly process, questions and supporting information, user answers and annotations, and every modification made to the text of the document is stored in this file. The file can even contain workflow information to allow users to forward or return documents using a single-key process, and to control what Rapidocs functionality can be used when working with the document.

This single-file format makes file management simpler, and reduces version incompatibilities that could otherwise occur. Being small it can be transferred across the Internet quickly and efficiently, minimising online time where that is an issue.

As a proprietary format containing no procedural code, Rapidocs files are immune from damaging viruses.

E-commerce / Unrestricted Templates

Rapidocs templates created using Rapidocs Professional contain an originating site code that restricts their use. Before any template can be used with the free Classic version they must be published for 'Unrestricted' use or unlocked from an e-commerce site.

Rapidocs templates distributed via web sites such as Desktop Lawyer or Law Assure are unlocked at the time they are downloaded from the site. These templates can be used with any Rapidocs client.

The ability to publish 'Unrestricted' templates directly is presently limited to use by Epoch Software only.

Rapidocs Red Templates

A 'red' document icon in Rapidocs represents a published template. There is no distinction between templates that have been published for an enterprise or those that are published for e-commerce use. As described above, Rapidocs Classic can only open e-commerce published templates.

Opening a 'red' template immediately generates a 'blue' document instance. Thus DataServer cannot modify and resave a red template.

Rapidocs Blue Documents

A 'blue' document icon in Rapidocs represents a specific document instance. Each instance contains a unique identification code, and Rapidocs clients will only recognize one document instance with the same code. This effectively limits the user to 'one-time' use of such documents.

However, these blue documents can be distributed for use with any Rapidocs client. The functionality available to users when they work with these documents may be limited by the distributor of the document, as well as by the functionality available within the Rapidocs client being used.

Installing DataServer and RapidocsDocCOM

To install Dataserver and RapidocsDocCOM, it is simply a case of running each installer program. Both COM objects are licence controlled, so it's vital that the host machine has a "RUM" licence installed somewhere on it's hard-drive.

Manually Registering the COM objects

Generally speaking the installers will register the COM objects and set the licence registry key automatically, however if a user

ever wishes to manually register the DLL files containing the COM objects, it can be done as follows:

Open a DOS command line, and change the folder such that the path points to the folder containing **DataServer.dll** and **RapidocsDocCOM.dll** (assuming they are in the same folder).

Then type:

```
Regsvr32 DataServer.dll  
Regsvr32 RapidocsDocCOM.dll
```

Finally, type:

```
Regedit
```

This will open the registry editor to enable you to update the following registry key:

```
HKEY_CURRENT_USER\Software\Epoch\RapidocsDocumentClient\User  
Data
```

With the pathname and filename of the Rapidocs RUM licence. Use the string: `RUMPath="Pathname\LicenceName.rum"`

Once accomplished, the Rapidocs COM interfaces will be available for use in your own applications. You will be able to open Rapidocs documents from within your applications, and read or write to the document as well as searching the document database.

The process of using the COM objects will vary from one application to another, but this is covered in detail for the VBA and Delphi environments in this manual.

De-registering DataServer and RapidocsDocCOM

If there is a need to de-register either or both of these DLLs, for example if an updated version of one of them is to be installed, then this is done as follows:

```
Regsvr32/u DataServer.dll  
Regsvr32/u RapidocsDocCOM.dll
```

Visual Basic Environment for Data Server

This is a guide to the possibilities offered by Rapidocs Data Server in conjunction with Visual Basic for Applications. In the samples of code below, the underlined text designates an arbitrarily named variable, which should be declared before use as the type specified in Appendix A.

Preparing the VBA Environment

- Load the application which supports Visual Basic for Applications (e.g. Excel)
- Start the Visual Basic Editor from your application:
Tools | Macros | Visual Basic Editor
- Enable the Rapidocs DataServer reference:
Tools | References | DataServer Library
- Similarly for RapidocsDocCOM:
Tools | References | RapidocsDocCOM Library

Defining and Opening the Rapidocs Document

- To use Data Server within the client, new server object variables must be declared as follows:
Dim TheDocument As New Document {For DataServer}
Dim TheSearch As New DataStore {For RapidocsDocCOM}
- To create a new procedure:
Sub TheProcedure
Dim TheDocument As New Document
Dim TheSearch As New DataStore

```
`` Do something with the document
`` As Document goes out of scope it is automatically freed
End Sub
```

Opening a Rapidocs Document from a File

- To specify the name and location of the document:

```
TheDocument.DosName = "C:\PATHNAME\DOCUMENTNAME.RAP"
```

- To open the document from a file:

```
TheDocument.Open
```

Opening a Rapidocs Document from a Data Store

- Each document contains a unique reference number called the GUID (Global Unique Identifier), which is used to locate a document within a document store.
- The GUID used to call a document from a store is equivalent to the InstanceGUID in the Header Properties (see Appendices B and C).
- Specify the IP address and port number of the document store, and the GUID of the document to load.

```
TheDocument.IPAddress = "127.0.0.1"
```

```
TheDocument.PortNumber = "9000"
```

```
TheDocument.GUID="2F5CC2B3-E753-4E68-BFBE-1CD87221ECE6"
```

- For testing purposes, sample GUID values can be obtained by clicking on the document 'properties' for a Rapidocs document in Rapidocs Pro.
- To open the document from the store.

```
TheDocument.Open
```

Note In either case, opening from a data store or opening from a file, it is only possible to write to the answers within the document if it is blue. If the document to be opened is a red template, Data Server converts it upon opening to a blue read/write document with a new *InstanceGUID* and *DocumentGUID* (See Appendix C).

Searching for Documents According to Certain Criteria

- In many cases the GUID of the document is unknown (unlike in the above example).
- If this is the case, use RapidocsDocCOM together with a set of search criteria to retrieve the GUID(s) of the document(s) required.

- Setup initial variables

```
Dim ResultCount As Long
Dim N As Integer
Dim docGUID As String
```

- Point the COM variable at the data store to search.

```
TheSearch.IPAddress = "127.0.0.1"
TheSearch.PortNumber = "9000"
```

- Free any previous searches, and initialize search.

```
TheSearch.BeginSearch
```

- Give the Search Criteria required (in this case all documents of type "rap"). There are three search methods here: AddStringCriterion, AddDateCriterion and AddColorCriterion. Any number of these criteria can be included in one search. For more information on this see Appendix D.

```
Call TheSearch.AddStringCriterion("doctype", ssEqual,
"rap")
```

- Execute the search, with ResultCount output as the number of results in the search. Then loop through each result and retrieve the GUID using GetResultGUID. Finally, output to the screen the value of each GUID (Optional).

```
Call TheSearch.ExecuteSearch(ResultCount)
For N = 0 To ResultCount - 1
    Call TheSearch.GetResultGuid(N, docGUID)
    MsgBox (docGUID)
Next N
```

Retrieving Answer Information from a Rapidocs Document

- Find out the number of answers in the document:

```
NumberOfAnswers = TheDocument.AnswerCount
```

Note AnswerCount is 0 unless at least one of the questions has been answered in the document. If a document is opened from a red template, then AnswerCount is always 0 initially.

- Refer to answers by index (ranging from 0 to AnswerCount - 1)

```
TheDocument.Answers(AnswerIndex)
```

- Retrieve answer properties:

```
With TheDocument.Answer(AnswerIndex)
    .AsString
    .Asked
```

```
.Blank
.Name
.ID
End With
```

(For more properties see Appendix A)

- If a document has been conventionally assembled and you are accessing the document to retrieve those answers, use:

```
For I = 0 To TheDocument.AnswerCount - 1
    Answer = TheDocument.Answers(I)
    Value = Answer.AsString
Next I
```

Note All answer types (string, numeric, date, multi, and flag) support AsString. Additionally, NumericAnswers support AsNumeric, DateAnswers support AsDateTime and AsNull, and Flag answers support AsBoolean.

Writing New Answers to the Document

If you are accessing the document to write answers to the document, then a different strategy is required as AnswerCount will initially be zero.

Use TheDocument.AnswerByName("Name") to access an existing answer or to create a new one. Note that AnswerCount will change its value if a new answer is created. Note also that Answers(I) is not guaranteed to retain the same value if answers are added.

```
TheDocument.Answers(I).AsString="Epoch Software"
```

Or if the Answers are initially empty:

```
TheDocument.AnswerByName("TheCompanyName").AsString= "Epoch Software"
```

Loop Controlled Answers

Before you can input a loop controlled answer, you must set the loop control to the correct value. If the source document has a loop MyLoop containing two questions MyName and MyAge then begin by setting the Loop to the correct value (two, say):

```
TheDocument.AnswerByName("MyLoop").AsNumeric = 2
```

And then set the answers, with the loop index in square brackets

```
TheDocument.AnswerByName("MyName[1]").AsString := "Oliver"
```

```
TheDocument.AnswerByName("MyAge[1]").AsNumeric := 32  
TheDocument.AnswerByName("MyName[2]").AsString := "Joshua"  
TheDocument.AnswerByName("MyAge[2]").AsNumeric := 25
```

Illegal Values

The only values currently checked are the names of the answers themselves and whether they are correctly loop controlled. Calling `Document.AnswerByName("Fred")` is an error if 'Fred' is not the name of question. Equally, it is an error if it is a loop controlled question (since no loop index in square brackets has been given). In the example given above, calling `AnswerByName("MyName[3]")` is an error since the loop index is out of range.

Saving and Closing the Document

- To save the document:

```
TheDocument.Save
```

- To close the document:

```
TheDocument.Unload
```

If the connection to the interface is persistent, ie you are using the same `TheDocument` variable to access a number of documents sequentially, then it is necessary to call `TheDocument.Unload` between documents which unloads the document from the server and clears the values required to specify the document. It is not necessary to call `Unload` when the `TheDocument` variable is going out of scope.

Note The document must be unloaded first before opening another document with the same COM instance or variable. The document need not be unloaded if the `TheDocument` variable is going out of scope.

Deleting a Document

- To delete the document:

```
TheDocument.Delete
```

- If the document was opened from a file, it is deleted from the same file location. Similarly, if it was opened from a document store, the file is deleted from the store. Delete will also close (unload) the document from the COM interface.
- The Delete method can be used without having first opened the document.

Password Protected Documents

- To test whether the document is password protected and enter the password if it is the case:

```
If TheDocument.PasswordProtected = True Then
    TheDocument.Password = "My Password"
EndIf
TheDocument.Open
```

- This test should be done before the document is opened.

Transferring a Document

Transfer Methods

The Save property always returns a file to wherever it came from. If there is a requirement to move or copy a file between document stores or files then it is necessary to use the Transfer methods.

- There are two methods as follows:

```
TheDocument.TransferToStore(Mode of Transfer, Destination
Port number, Destination IP address)
```

```
TheDocument.TransferToFile(Mode of Transfer, pathname)
```

- Mode of Transfer takes one of three possible parameters:
 - **dtCopyCopy**: Creates a read only copy in the destination.
 - **dtCopyOriginal**: Creates a read/write copy in the destination, leaving a read only copy in the source.
 - **dtMove**: Creates a read/write copy in the destination and removes the equivalent file in the source.
- The destination may be the same as the source. This is achieved by setting the destination port number and IP address as the same as those used to open the document. In the case of a file, set the destination pathname to be the same as `DosName` used to open the document.
- It is possible to transfer documents between one document store to another, from a file to a store, from file to file or from a store to a file.

Transfer Methods Summary

The possible outcomes of the **TransferToStore** method are summarized below:

Source	Destination	Mode of Transfer	Port Number	IPAddress	Result in Source	Result in Destination
DataStore1	DataStore1	dtCopyCopy	DS1 Port	DS1 IP	Existing doc retained. New	(as source)

Source	Destination	Mode of Transfer	Port Number	IPAddress	Result in Source	Result in Destination
					read only copy created.	
DataStore1	DataStore1	dtCopyOriginal	DS1 Port	DS1 IP	Existing doc becomes read only. New read/write copy created.	(as source)
DataStore1	DataStore1	DtMove	DS1 Port	DS1 IP	Read/write copy created. Existing doc removed.	(as source)
DataStore1	DataStore2	dtCopyCopy	DS2 Port	DS2 IP	Existing doc retained.	New read only copy created.
DataStore1	DataStore2	dtCopyOriginal	DS2 Port	DS2 IP	Doc becomes read only.	Read/Write copy created.
DataStore1	DataStore2	dtMove	DS2 Port	DS2 IP	Existing doc removed.	Read/Write copy created.
File1	DataStore1	dtCopyCopy	DS1 Port	DS1 IP	Existing doc retained.	New read only copy created.
File1	DataStore1	dtCopyOriginal	DS1 Port	DS1 IP	Doc becomes read only.	Read/Write copy created.
File1	DataStore1	dtMove	DS1 Port	DS1 IP	Existing doc removed.	Read/Write copy created.

The possible outcomes of the ***TransferToFile*** method are summarized below:

Source	Destination	Mode of Transfer	Pathname	Result in Source	Result in Destination
File1	File1	dtCopyCopy	Path\File1	Existing doc overwritten. New read only copy created.	(as source)
File1	File1	dtCopyOriginal	Path\File1	Existing doc overwritten. New read/write copy created.	(as source)
File1	File1	dtMove	Path\File1	Read/write copy created. Existing doc removed.	(as source)
File1	File2	dtCopyCopy	Path\File2	Existing doc retained.	New read only copy created.
File1	File2	dtCopyOriginal	Path\File2	Doc becomes read only.	Read/write copy created.
File1	File2	dtMove	Path\File2	Existing doc removed.	Read/write copy created.
DataStore1	File1	dtCopyCopy	Path\File1	Existing doc retained.	New read only copy created.
DataStore1	File1	dtCopyOriginal	Path\File1	Doc becomes read only.	Read/write copy created.

DataStore1	File1	dtMove	Path\File1	Existing doc removed.	Read/write copy created.
------------	-------	--------	------------	-----------------------	--------------------------

See Appendix C for a diagramatic explanation of this process.

Transfer Examples

- To move a document from a file to a specified document store:

```
TheDocument.DosName = "C:\Rapidocs\document1.rap"
TheDocument.Open
TheDocument.TransferToStore(dtMove, "9000", "127.0.0.1")
TheDocument.Unload
```

- To open a document from a document store, then save a copy of it to a different store:

```
TheDocument.PortNumber="9000"
TheDocument.IPAddress="127.0.0.1"
TheDocument.GUID="2F5CC2B3-E753-4E68-BFBE-1CD87221ECE6"
TheDocument.Open
TheDocument.TransferToStore(dtCopyCopy, "9000",
"193.123.5.152")
```

Visual Basic Examples

Example 1 – Listing the Answers

This code opens a blue document and puts all the names of the answers as well as the answer contents into a column of an Excel spreadsheet alongside the question index.

```
Sub ListTheAnswers()
Dim I As Integer
Dim TheDocument As New Document
TheDocument.DosName = "C:\Rapidocs\document1.rap"
TheDocument.Open
For I = 0 To TheDocument.AnswerCount - 1
Sheet1.Cells(I+1,1)=I
Sheet1.Cells(I+1,2)=TheDocument.Answers(I).Name
Sheet1.Cells(I+1,3)=TheDocument.Answers(I).AsString Next I
TheDocument.Unload
End Sub
```

Note In order to use the `Answers(Answer Index)` property, the answers in the document must already exist. If you were to use the `Answers(I)` property for an empty answer, then an error is returned.

If you wish to populate a document with Answers of which some may be empty, you must use the `AnswerByName(Answer Name)` property.

Example 2 – Writing New Answers

This code opens a blue document from a document store (using the store IP address and port number together with the document GUID). It then goes through the answers and prompts the user to change the value of each one. The new answers are then saved back to the document store.

```

Sub WriteNewAnswers()
    Dim TheDocument As New Document
    Dim I as Integer
    TheDocument.GUID="15AA3D63-DA6A-4BE2-86CF-B4682074392E"
    TheDocument.IPAddress="127.0.0.1"
    TheDocument.PortNumber="9000"
    TheDocument.Open
    For I=0 to TheDocument.AnswerCount-1
        TheDocument.Answers(I).AsString=InputBox$( )
    Next I
    TheDocument.Save
    TheDocument.Unload
End Sub

```

Example 3 – Searching and Manipulating Documents

In the two previous examples it is assumed that the document GUID is known beforehand. Generally however this would not be the case. The following example shows how to obtain documents without knowing their GUID or filename beforehand.

This code searches a document store for all documents of type 'rap' (i.e. Rapidocs documents) and then using DataServer outputs the document name of each document found.

```

Sub Main()
    Dim TheSearch As New RapidocsDocCOM.DataStore
    Dim TheDocument As New DataServer.Document
    Dim ResultCount As Long
    Dim N As Integer
    Dim DocGUID As String
    On Error Resume Next
    TheSearch.IPAddress = "127.0.0.1"
    TheSearch.PortNumber = "9000"
    TheSearch.BeginSearch
    Call TheSearch.AddStringCriterion("doctype", ssEqual,
"rap")
    Call TheSearch.ExecuteSearch(ResultCount)
    For N = 0 To ResultCount - 1
        Call TheSearch.GetResultGuid(N, DocGUID)
        TheDocument.IPAddress = "127.0.0.1"
        TheDocument.PortNumber = "9000"
        TheDocument.GUID = DocGUID
        TheDocument.Open
        MsgBox (TheDocument.Header.DocumentName)
        TheDocument.Unload
    Next N
End Sub

```

Delphi Environment for Data Server

This is a guide to the possibilities offered by Rapidocs Data Server in conjunction with Delphi. In the samples of code below, the underlined text designates an arbitrarily named variable, which should be declared before use as the type specified in Appendix A.

Preparing the Delphi Environment

- Load Delphi application
- Import type library
- Project | Import Type Library.../ Add
- Browse to DataServer.dll file and press the 'Install' button
- Similarly, browse for RapidocsDocCOM.dll and press 'Install'.

Setting Up the DataServer and RapidocsDocCOM Object Variables

Declaring the Data Server Object Variable

To use Data Server within the client, a new server object variable must be declared.

DataServer supports three types of automation: Interface, Disinterface and Variants:

1. TheDocument: IDocument ;

or

2. TheDocument: IDocumentDisp ;

or

```
3. TheDocument: OleVariant;
```

Although all three interface methods are supported by Delphi, using IDocument (by specifically importing the TypeLib) is much the preferred method since it supports early binding and is many times quicker.

Creating an Instance of DataServer

```
1. TheDocument:= CoDocument.Create;
```

or

```
2. TheDocument:= CreateComObject(CLASS_Document) as  
IDocumentDisp;
```

or

```
3. TheDocument:= CreateOleObject('DataServer.Document');
```

Declaring the RapidocsDocCOM Object Variable

To use RapidocsDocCOM within the client, a new server object variable must be declared.

RapidocsDocCOM supports three types of automation: Interface, Disinterface and Variants:

```
1. TheSearch: IDataStore;
```

or

```
2. TheSearch: IDataStoreDisp;
```

or

```
3. TheSearch: OleVariant;
```

Although all three interface methods are supported by Delphi, using IDataStore (by specifically importing the TypeLib) is much the preferred method since it supports early binding and is many times quicker.

Creating an Instance of RapidocsDocCOM

```
1. TheSearch:= CoDataStore.Create;
```

or

```
2. TheSearch:= CreateComObject(CLASS_DataStore) as  
IDataStoreDisp;
```

or

```
3. TheSearch:=  
CreateOleObject('RapidocsDocCOM.IDataStore');
```

Opening a Rapidocs Document from a File

- Specify the name and location of the document:

```
TheDocument.DosName := 'C:\PATHNAME\DOCUMENTNAME.RAP';
```

- Open the document from a file.

```
TheDocument.Open;
```

Opening a Rapidocs File from a Data Store

- Each document contains a unique reference number called the GUID (Global Unique Identifier) which is used to locate a document within a document store.
- The GUID used to call a document from a store is equivalent to the InstanceGUID in the Header Properties (see Appendix B and C).
- Specify the IP address and port number of the document store, and the GUID of the document to load.

```
TheDocument.IPAddress := '127.0.0.1';
```

```
TheDocument.PortNumber := '9000';
```

```
TheDocument.GUID := '2F5CC2B3-E753-4E68-BFBE-1CD87221ECE6';
```

- Sample GUID values can be obtained by clicking on the document 'properties' for a Rapidocs document in Rapidocs Professional.
- Open the document from the store using:

```
TheDocument.Open;
```

Note In either case, opening from a Data Store or opening from a file, it is only possible to write to the answers within the document if it is blue. If the document to be opened is a red template, Data Server converts it upon opening to a blue read/write document with a new *InstanceGUID* and *DocumentGUID* (See Appendix C).

Searching for Documents According to Certain Criteria

- In many cases the GUID of the document is unknown (unlike in the above example).
- If this is the case, use RapidocsDocCOM together with a set of search criteria to retrieve the GUID(s) of the document(s) required.

- Setup initial variables:

```
ResultCount : LongInt;
```

```
N : Integer;
```

```
docGUID : String;
```

- Point the COM variable at the data store to search:

```
TheSearch.IPAddress := '127.0.0.1';
```

```
TheSearch.PortNumber := '9000';
```

- Free any previous searches, and initialize search:

```
TheSearch.BeginSearch;
```

- Give the Search Criteria required (in this case all documents of type “rap”). There are three search methods: AddStringCriterion, AddDateCriterion and AddColorCriterion. Any number of these criteria can be included in one search. For more information see Appendix D.

```
TheSearch.AddStringCriterion('doctype',ssEqual,'rap');
```

- Execute the search, with ResultCount output as the number of results in the search. Then loop through each result and retrieve the GUID using GetResultGUID. Finally, output to the screen the value of each GUID (Optional).

```
TheSearch.ExecuteSearch(ResultCount);
For N = 0 To ResultCount - 1 Do Begin
  TheSearch.GetResultGuid(N, docGUID);
  ShowMessage (docGUID);
End;
```

Retrieving Answer Information from a Rapidocs Document

- Find out the number of answers in the document:

```
NumberOfAnswers := TheDocument.AnswerCount;
```

Note AnswerCount is zero unless at least one of the questions has been answered in the document. If a document is opened from a red template, then AnswerCount is always zero initially.

- Refer to answers by index (ranging from 0 to AnswerCount - 1):

```
TheDocument.Answers[AnswerIndex];
```

- Retrieve answer properties:

```
With TheDocument.Answers[AnswerIndex] Do Begin
  EditAnswer.Text := .AsString;
  CheckBoxAsked.Checked := .Asked;
  CheckBoxBlank.Checked := .Blank;
  EditName.Text := .Name;
  CheckBoxR.Checked := .Required;
  EditID.Text := IntToStr(.ID);
End;
```

(For more properties see Appendix A)

- If a document has been conventionally assembled and you are accessing the document to retrieve those answers, use:

```

For I := 0 To TheDocument.AnswerCount - 1 Do Begin
    Answer := TheDocument.Answers[I];
    Value := Answer.AsString;
End;

```

Note All answer types (string, numeric, date, multi, and flag) support AsString. Additionally, NumericAnswers support AsNumeric, DateAnswers support AsDateTime and AsNull, and Flag answers support AsBoolean.

Writing New Answers to the Document

If you are accessing the document in order to write answers to it, then a different strategy is required as AnswerCount will initially be zero.

Use `TheDocument.AnswerByName['Name']` to access an existing answer or to create a new one. Note that AnswerCount will change its value if a new answer is created. Note that `Answers[I]` is not guaranteed to retain the same value if answers are added.

```
TheDocument.Answers[I].AsString := 'Epoch Software';
```

Or if the Answers are initially empty:

```
TheDocument.AnswerByName['TheCompanyName'].AsString :=
'Epoch Software';
```

Loop Controlled Answers

Before you can input a loop controlled answer, you must set the loop control to the correct value. If the source document has a loop MyLoop containing two questions MyName and MyAge then begin by setting the Loop to the correct value (for example, two):

```
TheDocument.AnswerByName['MyLoop'].AsNumeric := 2;
```

Then set the answers, with the loop index in square brackets:

```
TheDocument.AnswerByName['MyName[1]'].AsString := 'Oliver';
```

```
TheDocument.AnswerByName['MyAge[1]'].AsNumeric := 32;
```

```
TheDocument.AnswerByName['MyName[2]'].AsString := 'Joshua';
```

```
TheDocument.AnswerByName['MyAge[2]'].AsNumeric := 25;
```

Illegal Values

The only values currently checked are the names of the answers themselves and whether they are correctly loop controlled. Calling `Document.AnswerByName['Fred']` is an error if 'Fred' is not the name of question. Equally, it is an error if it is

a loop controlled question (since no loop index in square brackets has been given). In the example given above, calling `AnswerByName['MyName[3] ']` is an error since the loop index is out of range.

Saving and Closing the Document

- Save the document (only applicable to blue documents or red templates):

```
TheDocument.Save;
```

- Close the document:

```
TheDocument.Unload;
```

If the connection to the interface is persistent, i.e. you are using the same `TheDocument` variable to access a number of documents sequentially, then it is necessary to call `TheDocument.Unload` between documents which unloads the document from the server and clears the values required to specify the document. It is not necessary to call `Unload` when the `TheDocument` variable is going out of scope.

Note The document must first be unloaded before opening another document with the same COM instance or variable. The document need not be unloaded if the `TheDocument` variable is going out of scope.

Deleting a Document

- To delete the document:

```
TheDocument.Delete;
```

- If the document was opened from a file, it is deleted from the same file location. Similarly, if it was opened from a document store, the file is deleted from the store. `Delete` will also close (unload) the document from the COM interface.
- The `Delete` method can be used without first having opened the document.

Password Protected Documents

- To test whether the document is password protected and enter the password if this is the case:

```
If TheDocument.PasswordProtected = True Then  
  TheDocument.Password = 'My Password';  
TheDocument.Open
```

- This test should be done before the document is opened.

Transferring a Document

- See section on Transferring a document in Visual Basic.

Transfer Examples

- To move a document from a file to a specified document store:

```
TheDocument.DosName := 'C:\Rapidocs\document1.rap';  
TheDocument.Open;  
TheDocument.TransferToStore(dtMove, '9000', '127.0.0.1');  
TheDocument.Unload;
```

- To open a document from a document store and then save a copy of it to a different store:

```
TheDocument.PortNumber:= '9000';  
TheDocument.IPAddress:= '127.0.0.1';  
TheDocument.GUID:= '2F5CC2B3-E753-4E68-BFBE-1CD87221ECE6';  
TheDocument.Open;  
TheDocument.TransferToStore(dtCopyCopy, '9000',  
'193.123.5.152');
```

Delphi Examples

Example 1 – Listing the Answers

This code opens a blue document and puts all the names of the answers as well as the answer contents into a column of a form:

```
Procedure ListTheAnswers;  
Var I : Integer;  
    TheDocument : IDocument;  
Begin  
    TheDocument := CoDocument.Create;      TheDocument.DosName  
:= 'C:\Rapidocs\document1.rap';  
    TheDocument.Open  
    For I := 0 To TheDocument.AnswerCount - 1 Do  
Begin  
    StringGrid.Cells[0, I+1] :=  
TheDocument.Answers[I].Name;  
    StringGrid.Cells[1, I+1] :=  
TheDocument.Answers[I].AsString;  
    End;  
    TheDocument.Unload;  
    TheDocument := NIL;  
End;
```

Note In order to use the `Answers(Answer Index)` property, the answers in the document must already exist. If you were to use the `Answers(I)` property for an empty answer, then an error is returned.

If you wish to populate a document with answers of which some may be empty, you must use the `AnswerByName(Answer Name)` property.

Example 2 – Writing New Answers

This code opens a blue document from a document store (using the store IP address and port number together with the document GUID). It then goes through the answers and prompts the user to change the value of each one. The new answers are then saved back to the document store.

```

Procedure WriteNewAnswers()
Var TheDocument : IDocument;
    I : Integer;
Begin
    TheDocument := CoDocument.Create;
    TheDocument.GUID := '15AA3D63-DA6A-4BE2-86CF-
B4682074392E';
    TheDocument.IPAddress := '127.0.0.1';
    TheDocument.PortNumber := '9000';
    TheDocument.Open;
    For I :=0 To TheDocument.AnswerCount-1 Do Begin
        TheDocument.Answers[I].AsString := InputBox('Input
Box', 'Prompt', 'Default string');
    End;
    TheDocument.Save;
    TheDocument.UnLoad;
    TheDocument := NIL;
End;

```

Example 3 – Searching and Manipulating Documents

In the previous example, it is assumed that the document GUID is known beforehand. Generally however this would not be the case. The following example shows how to obtain documents without knowing their GUID or filename beforehand.

This code searches a document store for all documents of type 'rap' (i.e. Rapidocs documents) and then using `DataServer` outputs the document name of each document found.

```

Procedure Main;
Var TheSearch : IDataStore;
    TheDocument : IDocument;
    ResultCount : Long;
    DocGUID : String;
    N : Integer;
    DocGUID : String;

```

```
Begin
  TheSearch:= CoDataStore.Create;
  TheDocument:=CoDocument.Create;
  Try
    Try
      TheSearch.IPAddress := '127.0.0.1';
      TheSearch.PortNumber := '9000';
      TheSearch.BeginSearch;
      TheSearch.AddStringCriterion('doctype', ssEqual,
'rap');
      TheSearch.ExecuteSearch(ResultCount);
      For N = 0 To ResultCount - 1 Do Begin
        TheSearch.GetResultGuid(N, DocGUID);
        TheDocument.IPAddress := '127.0.0.1';
        TheDocument.PortNumber := '9000';
        TheDocument.GUID := DocGUID;
        TheDocument.Open;
        ShowMessage(TheDocument.Header.DocumentName);
        TheDocument.Unload;
      End;
    Except
      Raise.Exception.Create('Error when searching for the
documents');
    End;
  Finally
    TheSearch := NIL;
    TheDocument := NIL;
  End;
End;
```

Header Properties

The Header properties of Rapidocs documents can be accessed using the Data Server Header method in the IDocument interface. These properties include information such as the Rapidocs document title, author, creation date etc. In many cases, these properties are read only; a list of properties is given in Appendix B.

These properties are accessed and manipulated in much the same way as the document properties in DataServer.

Using the Header Properties

The process of using the Header method will vary from one application to another, and is covered in detail for the VBA and Delphi environments below.

Visual Basic Environment

This is a guide to the possibilities offered by the Data Server Header method in conjunction with Visual Basic for Applications. In the samples of code below, the underlined text designates an arbitrarily named variable, which should be declared before use as the type specified in Appendix B.

Retrieving Information from the Document Header

The document must be opened (from a file or document store) in the same way as shown in 1.3.2.

- To retrieve selected document header information (for example):

```
With TheDocument.Header  
.Author  
.Category  
.CopyRight  
.Description
```

```
.Keywords  
.Owner  
.TemplateName  
.Version  
.Subject  
End With
```

For more properties, see Appendix B.

Writing to the Document Header

- To change the copyright information and version number within the document:

```
TheDocument.Header.Subject="Demonstration document"
```

```
TheDocument.Header.Version=" 2.9"
```

Example Procedure

- This program changes the Author and Description fields of a Rapidocs document, and then saves the document.

```
Sub NewDocumentAuthor ()  
  Dim TheDocument As New Document  
  TheDocument.DosName = "C:\PATHNAME\DOCUMENTNAME.RAP"  
  TheDocument.Open  
  
  TheDocument.Header.Author="John Smith"  
  TheDocument.Header.Department="Technical Division"  
  TheDocument.Save  
  TheDocument.Unload  
End Sub
```

Delphi Environment

This is a guide to the possibilities offered by the Data Server Header method in conjunction with Delphi. In the samples of code below, the underlined text designates an arbitrarily named variable, which should be declared before use as the type specified in Appendix B.

Retrieving Information from the Document Header

The document must be opened (from a file or document store) in the same way as shown in 1.4.2.

- To retrieve selected document header information (for example):

```
With TheDocument.Header Do Begin  
  Memo.Lines.Add(.Author);  
  Memo.Lines.Add(.Category);
```

```
Memo.Lines.Add(.CopyRight);
Memo.Lines.Add(.Description);
Memo.Lines.Add(.Keywords);
Memo.Lines.Add(.Owner);
Memo.Lines.Add(.TemplateName);
Memo.Lines.Add(.Version);
Memo.Lines.Add(.Subject);
End
```

For more properties, see Appendix B.

Writing to the Document Header

- To change the copyright information and version number within the document:

```
TheDocument.Header.Subject := ' Demonstration Document';
TheDocument.Header.Version := '2.9';
```

Example Procedure

- This program changes the Author of a Rapidocs document:

```
Procedure NewDocumentAuthor;
Var TheDocument : Idocument;
Begin
    TheDocument := CoDocument.Create;
    TheDocument.DosName := 'C:\PATHNAME\DOCUMENTNAME.RAP';
    TheDocument.Open;
    TheDocument.Header.Author := 'John Smith';
    TheDocument.Header.Department := 'Technical Division';
    TheDocument.Save;
    TheDocument.Unload;
    TheDocument := NIL;
End;
```

Using RapidocsX

Introduction

RapidocsX is essentially the Rapidocs Classic application running as an ActiveX control in a web page or embedded in a form using another programming language. The benefit of this is that it simplifies the installation process of Rapidocs Classic and enables server-side document management and workflow.

From a client perspective, when a user first visits a website that contains RapidocsX and selects a Rapidocs document, the control is automatically downloaded onto their machine. Following this download, the document is then immediately opened within RapidocsX (which is embedded within the web browser). Hence, from the users point of view it is as if they are able to edit the Rapidocs document on-line through their browser.

Once the ActiveX control has been downloaded by a user, there is never a need to go through the download process a second time. In other words, the next time the user opts to edit a document it opens immediately within RapidocsX and requires no additional download. However, if the ActiveX control is updated on the server, this will automatically update the client's copy.

RapidocsX utilises a three-tier database architecture for opening and saving Rapidocs documents (see Appendix E).

Client Machine

RapidocsX runs within MS Internet Explorer on the client machine. The client machine needs access to the Internet or Intranet to allow RapidocsX to transfer Rapidocs documents to and from the Rapidocs server machine. RapidocsX is configured to point to a specific IP address on the network, which is a computer running the Rapidocs document server. A specific PORT is configured for communications, typically port 9000. The protocol is always TCP/IP.

Most versions of Netscape Navigator do not support ActiveX controls, and there is currently no Netscape plugin equivalent of RapidocsX. Therefore, if a user wishing to download a Rapidocs document has Netscape, it will be necessary to offer them the off-line equivalent (i.e. Rapidocs Classic).

Server Machine

The server is configured to be the Rapidocs document server. It is configured with a unique IP address on the Internet/Intranet within which it exists.

A Rapidocs document server interface runs on this computer and is the front-end to all communications with the Rapidocs document store. The RapidocsX client communicates directly with this interface. The document server is configured to process all communications with RapidocsX clients on port 9000 or, in fact, any port it chooses.

An SQL database is set up to run as the Rapidocs document store. In this case, the database is used to store both “red” Rapidocs document templates and “blue” Rapidocs documents that users have completed and saved.

Each document in the document store is identified with a Global Unique Identifier, GUID. Blue and red documents are stored in the same way, except that blue documents have extra information stored in the records to identify the owner. In the same way as Rapidocs Classic, whenever a “red” Rapidocs template is opened in RapidocX, it is immediately converted to a “blue” document with a new GUID. This GUID is equivalent to the InstanceGUID in the document Header properties.

Implementing RapidocsX

Essentially, RapidocsX client has four basic functions:

- Opens a blue document from a red template from the document store.

Note Opening from a red template can only be achieved if the red template is unrestricted or opened from an e-commerce site (see Rapidocs Basics).

- Saves a completed blue document to the document store, therefore creating a new record and GUID.
- Opens/reopens a blue completed document.
- Overwrites a completed blue document with an amended version.

For RapidocsX client to be able to open a blue document from a red template, it first needs five parameters to be configured. These are **SourceGUID**, **SourcePort**, **SourceDataStore**, **TargetDataStore** and **TargetPort**.

SourceGUID points to the document in the document store that is to be transferred and opened. If it points to a red document template, a blue document is created and opened simultaneously. (The *SourceGUID* is equivalent to the *InstanceGUID* in the header properties).

SourcePort selects the port used to communicate with the (source) document server and has to match the port number the document server is set up to use.

SourceDataStore is the IP address of the document server that the RapidocsX client is to use for opening documents.

TargetDataStore is the IP address of the document server that the RapidocsX client is to use for saving documents. This may be the same as SourceDataStore.

TargetPort selects the port used to communicate with the (target) document server. This may or may not match SourcePort.

These five parameters are sent to the RapidocsX client by using static HTML code on the web page that encapsulates the RapidocsX ActiveX control.

Example of Opening a Document from a Store

Below is an example of this HTML code:

```
<OBJECT name="rapX"
classid="clsid:E5C97835-6865-443E-8C33-671D9C71A6D0"
codebase="RapidocsX.cab#version=1,0,999,0"
width=640
height=480
align=center
hspace=0
vspace=0>

<Param Name = "SourcePort" Value = "9000">
<Param Name = "TargetPort" Value = "9000">
<Param Name = "SourceDataStore" Value = "123.123.123.123">
<Param Name = "TargetDataStore" Value = "123.123.123.123">
<Param Name = "SourceGUID" Value = "E1C1BA17-64FF-4C52-AAC6-
333B30A2CE72">

</OBJECT>
```

This HTML code will set up the RapidocsX client to open a document that resides in the Rapidocs document store. In this case the source and target document store is the same.

Saving is performed within the RapidocsX client itself by simply pressing the save icon or by clicking 'save' on the drop-down

menu. This action transfers the user's completed blue document to the document server and finally to the Rapidocs document store.

Example of Opening a Document from a File

It is also possible (although perhaps less desirable) to open a document directly from its locally stored file. This is done using the **SourceDosName** parameter as the pathname and filename of the file required. If the return location of the file is to be different to the SourceDosName location, the option of setting the **TargetDosName** parameter is available and should be assigned before SourceDosName. By default this parameter equals SourceDosName.

The following is an example of how the HTML code could look:

```
<OBJECT name="rapX"
classid="clsid:E5C97835-6865-443E-8C33-671D9C71A6D0"
codebase="RapidocsX.cab#version=1,0,999,0"
width=640
height=480
align=center
hspace=0
vspace=0>

<Param Name = "SourceDosName" Value =
"C:\pathname\filename.rap">

</OBJECT>
```

Proxy Server

If either the source or target document server(s) use a proxy server, then this can be incorporated using the following parameters:

SourceProxy: The IP address of the proxy server for the source document server.

TargetProxy: The IP address of the proxy server for the target document server.

SourceProxyPort: The port number of the proxy server for the source document server.

TargetProxyPort: The port number of the proxy server for the target document server.

Mechanism for Selecting Documents

The browsing and selection of documents for opening or reopening using Internet Explorer is performed using a server-side markup language such as Cold Fusion or ASP that can read the contents of an SQL database i.e the Rapidocs document store. RapidocsX client and Rapidocs document server will be totally unaware of this mechanism. The only

thing RapidocsX client requires is a static web page, as above, for setting the essential parameters for opening a blue or red document.

RapidocsDocCOM should be called by the server-side language in order to obtain the GUID to be used by RapidocsX in the static HTML.

Furthermore, prior to opening the document in RapidocsX, DataServer can be used to pre-populate some or all of the answer fields.

Event Handlers

Event handlers are used to inform a program when an outside event has occurred, for example when a Rapidocs document has been successfully loaded into the RapidocsX client. There are three event handlers supported by RapidocsX:

- `onLoad` – Triggered when a Rapidocs document has been fully loaded in RapidocsX
- `onSave` – Triggered when a user has chosen to save the current document in RapidocsX
- `onUnLoad` – Triggered when a user has chosen to close the document in RapidocsX
- `onHelpClick` – Triggered when a user clicks Help | Contents in the menu bar in RapidocsX

The best way to use them in the web-environment is to define the actions that should be triggered by the events using JavaScript. This can be done by inserting the JavaScript in the HTML Header section, as follows:

```
<script language="JavaScript" event="onEvent" for="RapidocsX
Object Name">
/*
Insert the action after the event is triggered here
*/
</script>
```

Example

In the following example a Java alert will be displayed when a user saves a Rapidocs document that they are editing within RapidocsX. In this case the RapidocsX object Name is given as 'rapX' (as used in previous examples), however this is arbitrary.

```
<script language="JavaScript" event="onSave" for="rapX">
alert('Document is saved')
</script>
```

RapidocsX Parameters

A summary of the essential document-handling parameters used to call RapidocX from an HTML page are shown below:

Parameter	Access	Required for Doc store?	Required for Doc in file?	Explanation
SourceDataStore	Write Only	Yes	No	Set the IP address of the source data store.
SourceDosName	Write Only	No	Yes	Set the pathname of the source file.
SourceGUID	Write Only	Yes	No	Set the GUID of the rap file in the data store.
SourcePort	Write Only	Yes	No	Set the Port number of source data store.
SourceProxy	Write Only	Optional	No	Set the proxy IP address of the source data store.
SourceProxyPort	Write Only	Optional	No	Set the proxy port number of the source data store
TargetDataStore	Write Only	Yes	No	Set the IP address of the target data store.
TargetDosName	Write Only	No	Optional	Set the pathname of the target file store.
TargetPort	Write Only	Yes	No	Set the Port number of the target data store.
TargetProxy	Write Only	Optional	No	Set the proxy IP address of the target data store.
TargetProxyPort	Write Only	Optional	No	Set the proxy port number of the target data store.

A list of other optional RapidocsX parameters is given below (these are all write only). For more details on each, see the Header properties in Appendix B.

Parameter	Parameter
Author	Keywords
Category	Language
CheckedBy	Manager
ClientCompany	MemberGroup
ClientContact	Office
Department	Owner
Description	Project
DocumentComment	ReceivedFrom
DocumentCreationDate	ReturnEmailAddress
DocumentName	Subject
ForAttentionOf	Version

These parameters can be written to in the same way as the document handlers using HTML, *providing it's done before the document handling parameters are invoked*, namely:

```
<OBJECT...
...>
```

```
<Param Name = "Parameter Name" Value = "Parameter Value">  
</OBJECT>
```

Appendix A: Rapidocs DataServer Object Properties and Methods

Document

The document methods and properties are available when a Rapidocs document has been opened by using the **IDocument** interface.

A summary of the functions of this interface are as follows:

Can	Cannot
Open red or blue documents. Red becomes blue when opened.	Write questions
Read/Write answers from a blue document	Open black, or pink documents
Save from file to file	Save red documents as red documents.
Save from store to store	Read question information for a document

Appendix A

Save blue documents from red/blue	
Transfer/copy documents from one store to another	
Transfer/copy documents from one file to another	

The properties for IDocument are given below:

Property/Syntax	Type	Access	Explanation
AnswerByName(<i>Name as string</i>)	WideString	Read/Write	Returns or sets the answer according to the answer name.
AnswerCount	Integer	Read Only	Returns the number of answers in the document.
Answers(<i>Index: Integer</i>)	IAnswer	Read/Write	Returns or sets the answer according to the index.
DosName	WideString	Write Only	Sets the pathname and filename of the Rapidocs document to be opened by the COM object.
GUID	WideString	Write Only	Sets the document GUID of the file to be opened.
IPAddress	WideString	Write Only	Sets the IP address of the document data store.
Password	WideString	Write Only	Sets the password for password protected documents.
PasswordProtected	WordBool	Read Only	Returns true if the document is password protected
PortNumber	WideString	Write Only	Sets the port number of the document data store.
ProxyIPAddress	WideString	Write Only	Sets the proxy IP address of the document data store.
ProxyPortNumber	WideString	Write Only	Sets the proxy port number of the document data store.
RapidocsLicencePath	WideString	Write Only	Manually sets the pathname of the RUM licence (see note in this section under Manually Setting the Licence Path).

The methods for IDocument are given below:

Method/Syntax	Explanation
Header	Exposes the header properties in Rapidocs. (see section 2 and Appendix B)
Open	Opens the document for the COM variable.
Save	Saves document back to the document store of file it originated from.
TransferToFile(<i>mode of transfer, Pathname</i>)	Transfers document to a file given by pathname \filename.
TransferToStore(<i>mode of transfer, Store Port, Store IP</i>)	Transfers document to a document store given by store IP address and Port.
Unload	Frees-up internal storage of document properties.

Answer Properties

The Answer properties are available by using either AnswerByName(Name as String) or Answers(Answer Index) properties.

Property/Syntax	Type	Access	Explanation
-----------------	------	--------	-------------

Property/Syntax	Type	Access	Explanation
Annotation	WideString	Read/Write	Returns or sets the annotation for the answer.
AsBoolean	WordBool	Read/Write	Returns or sets the answer to a boolean question.
AsDateTime	Double	Read/Write	Returns or sets the answer to a date question.
Asked	WordBool	Read Only	Returns true if question has been asked. ¹
AsNumeric	Double	Read/Write	Returns or sets the answer to a numeric question.
AsString	WideString	Read/Write	Returns or sets the answer to a string question.
Blank	WordBool	Read Only	Returns true if the answer is blank. ²
ID	Integer	Read Only	Returns the type ID for the answer. ³
IDString	WideString	Read Only	Returns the answer type by name e.g. "string"
Name	WideString	Read Only	Returns the answer name.
NullDate	WordBool	Read/Write	Return or set true if the date question is Null..

Manually Setting the Licence Path

The property `RapidocsLicencePath` exists in both `DataServer` and `RapidocsDocCOM` as a means to manually set the file location of the RUM licence file. This property is optional and generally does not require to be set, however in certain cases it has been found that a hosting application is unable to locate the RUM licence when the COM objects are invoked. As they are licence controlled it means that neither `RapidocDocCOM` nor `DataServer` will run properly in this case.

The solution is to hardcode the licence pathname as shown below prior to invoking any of the properties or methods of the COM objects:

```
Dim TheDocument as New DataServer.Document
Dim TheSearch as New RapidocsDocCOM.DataStore

TheDocument.RapidocsLicencePath = "C:\Program
Files\Rapidocs\rapidocs.RUM"
TheSearch.RapidocsLicencePath = "C:\Program
Files\Rapidocs\rapidocs.RUM"
```

This is known to be a problem when using certain web servers in conjunction with the COM objects. In particular, Microsoft IIS will require the licence path to be set manually.

¹ If Asked=True, then the user will not be prompt for the answer when assembling in Rapidocs

² If Blank=True, then the user will not see the answer during assembly.

³TypeID number: String=0, Boolean=1, Numeric=3, Date=4, Multi=5

Appendix B: Rapidocs Document Header Object Properties

Properties

The table below lists the properties of the Rapidocs Header along with their function, type and access properties.

Property/Syntax	Type	Access	Explanation
Author	WideString	Read/Write	Returns or sets the document author.
Authorizable	WordBool	Read Only	Returns true if authorization is needed for the document.
AuthorizableBy	WideString	Read Only	Returns the name(s) of the individual(s) from whom authorization is required
Authorized	WordBool	Read Only	Returns true if the document has been authorized.
AuthorizedBy	WideString	Read Only	Returns the name of the individual who authorized the document.
AuthorizedOn	TDateTime	Read Only	Returns the date of authorization.
Category	WideString	Read/Write	Returns or sets the document category.

Appendix B

Property/Syntax	Type	Access	Explanation
ChallengeResponseMessage	WideString	Read Only	Returns the message displayed before the document is unlocked.
CheckedBy	WideString	Read/Write	Returns or sets the name of the individual that the document was checked out by.
ClientCompany	WideString	Read/Write	Returns or sets the document client company.
ClientContact	WideString	Read/Write	Returns or sets the document client contact details.
CompletedDate	TDateTime	Read/Write	Returns or sets the document completion date.
CopyRight	WideString	Read Only	Returns the document copyright.
CreatedDocs	Integer	Read Only	Returns the number of documents created from the template (red only).
Department	WideString	Read/Write	Returns or sets the document department.
Description	WideString	Read/Write	Returns the document description.
DocumentColor	TDocumentColor	Read Only	Returns the document colour. ⁴
DocumentComment	WideString	Read/Write	Returns the comment associated with the document
DocumentCreationDate	TDateTime	Read Only	Returns the document creation date for the original document.
DocumentGUID	WideString	Read Only	Returns the GUID of the document. (see note below)
DocumentName	WideString	Read/Write	Returns or sets the document name.
ECommerceDocumentPage	WideString	Read Only	Returns the Ecommerce document page.
ECommerceHomePage	WideString	Read Only	Returns the document Ecommerce home page.
ECommerceMessage	WideString	Read Only	Returns the Ecommerce message for the document.
ECommerceTokenPage	WideString	Read Only	Returns the Ecommerce token page for the document.
Expirable	WordBool	Read Only	Returns true if the document is set to expire.
ExpirationDate	TDateTime	Read Only	Returns the date the document is set to expire.
Expired	WordBool	Read Only	Returns true if the document has expired.

⁴ The document colour indicates whether the document is an unpublished template (black), a published precedent (pink), a published template (red), or an individual document (blue). The variable type takes one of the following: dcBlack=4, dcPink=1, dcRed=2, or dcBlue=3 respectively. If none of these, it is dcIllegal.=0

Property/Syntax	Type	Access	Explanation
ForAttentionOf	WideString	Read/Write	Returns or sets the "For the Attention of" field in the document.
InstanceGUID	WideString	Read Only	Returns the GUID representing the "primary key" of the document in a document store. (see note below)
Keywords	WideString	Read/Write	Returns the document keywords.
Language	WideString	Read/Write	Returns or sets the (written) language of the document.
Manager	WideString	Read/Write	Returns or sets the name of the document manager.
MaxDocs	Integer	Read Only	Returns the max. number of documents to be generated by the template (red only).
MemberGroup	WideString	Read/Write	Returns or sets the document Member Group field.
Office	WideString	Read/Write	Returns or sets the document Office field.
Owner	WideString	Read/Write	Returns or sets the document owner.
PaclockString	WidsString	Read Only	Returns the padlock string.
PadlockColor	TPadlockColor	Read Only	Returns the padlock colour. ⁵
Password	WideString	Write Only	Sets the document password.
PasswordHint	WideString	Read Only	Returns the password hint.
PasswordProtected	WordBool	Read Only	Returns true if the document is password protected.
PriceWithoutSupport	WideString	Read Only	Returns the document price without support.*
PriceWithSupport	WideString	Read Only	Returns the document price with support.*
Project	WideString	Read/Write	Returns or sets the document project field.
PublicationDate	TdateTime	Read Only	Returns the document publication date.*
PublishedFrom	WideString	Read Only	Returns the name of the document that the document was published from.
Publisher	WideString	Read Only	Returns the document publisher.
PublisherCreationDate	TdateTime	Read Only	Returns the date when the pink, red or blue document was published from the black original.
PublisherHomePage	WideString	Read Only	Returns the publisher home page.

⁵ The PadlockColor determines the security used within the document. pcNone(0) = Not locked, pcBlue(1) = challenge response, pcGold(2) = e-commerce lock, pcSilver(3) = password protected, pcRed = illegal, pcOlive(4) = in transit, and pcWhite = not copyable

Appendix B

Property/Syntax	Type	Access	Explanation
ReceivedFrom	WideString	Read/Write	Returns the "received from" field in the document.
RegistrationCC	WideString	Read Only	Returns the registration CC field.*
RegistrationCountry	WideString	Read Only	Returns the registration Country.*
RegistrationDate	TDateTime	Read Only	Returns the registration date.*
RegistrationEMail	WideString	Read Only	Returns the registration Email address.
RegistrationName	WideString	Read Only	Returns the registration name.*
RegistrationOrderReference	WideString	Read Only	Returns the registration order reference details.*
RegistrationOrganisation	WideString	Read Only	Returns the registration organisation.*
RegistrationStreet	WideString	Read Only	Returns the registration street name.*
RegistrationZip	WideString	Read Only	Returns the registration zip code.*
ReturnDataStore	WideString	Read Only	Returns the return data store name.
ReturnEmailAddress	WideString	Read/Write	Returns the return email address for the document.
Subject	WideString	Read/Write	Returns or sets the document subject.
SupportMinutes	WideString	Read Only	Not used.*
TemplateComment	WideString	Read Only	Returns the comment within the template (red only).
TemplateGUID	WideString	Read Only	Returns the GUID of the original template document from which the blue document was generated. (see note below)
TemplateName	WideString	Read Only	Returns the template name (red only).
Version	WideString	Read/Write	Returns or sets the document version details.

* These properties are not currently used by Rapidocs.

Appendix C: DataServer Document Version Control

Data server enables users to not only save a document back to the document store or file it came from, but also move or copy the document.

In order to keep control of the document versions, copies of a document are usually made *read only*. This is controlled by three properties in the Header: TemplateGUID, InstanceGUID and DocumentGUID.

TemplateGUID: When a blue document is created from a red template, the template's GUID becomes the TemplateGUID in the document. This identifies the template the document was created from and never changes.

At that point, two other GUIDS are created. These are the DocumentGUID and the InstanceGUID and are initially set to the same value.

InstanceGUID: This is the unique identifier of a document in the store.

DocumentGUID: This identifies the document as it was originally created.

If the InstanceGUID and the DocumentGUID are the same, then this is the original document and any other documents with the same DocumentGUID are copies of it. If the DocumentGUID and the InstanceGUID are the same, then the document is read-write while if they are different the document is read only.

This diagram illustrates how the process works:

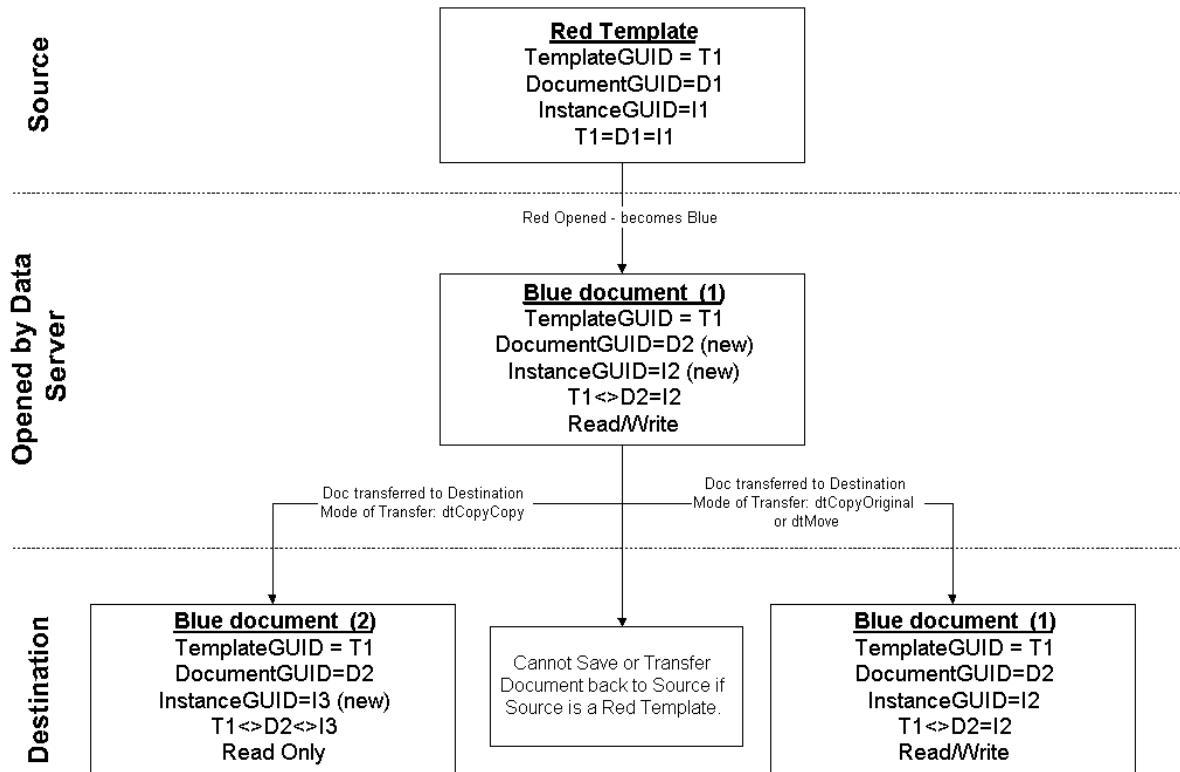


Figure C1 Rapidocs Document GUIDs

Appendix D: RapidocsDocCOM Object Properties and Methods

The properties for IDataStore are given below:

Property/ Syntax	Type	Access	Explanation
HostName	WideString	Write Only	Sets the host name of the document store (alternative to IPAddress).
IPAddress	WideString	Write Only	Sets the IP address of the document store
PortNumber	WideString	Write Only	Sets the port number of the document store.
RapidocsLicencePath	WideString	Write Only	Manually sets the pathname of the RUM licence (see note in Data Server Properties under Manually Setting the Licence Path).

The methods for IDataStore are given below:

Method/Syntax	Explanation
AddDateCriterion (<i>FieldName</i> as String, <i>SearchType</i> as DateSearchType, <i>Value</i> as String)	Includes date criterion within search. - Field upon which to search (table D1) - Search type defines search qualifier (table D2) - Value to search upon
AddDocColorCriterion (<i>FieldName</i> as String, <i>SearchType</i> as DocColorSearchTypes, <i>Value</i> as DocumentColors)	Searches for specific colour(s) of document - Field upon which to search (table D1) - Include/Exclude search value (dcsIs, dcsIsNot) - Value to search upon (table D3)
AddStringCriterion (<i>FieldName</i> as String, <i>SearchType</i> as StringSearchTypes, <i>Value</i> as String)	Includes string criterion within search - Field upon which to search (table D1) - Search type defines search qualifier (table D4) - Value to search upon
BeginSearch	Frees previous searches and begin new search.
ExecuteSearch (<i>ResultCount</i> as Long)	Executes the search according to the criteria set. - <i>ResultCount</i> contains the number of search results.
ExecuteSearchV (<i>ResultCount</i> as Variant)	Executes search with variant parameter (see note on VBScript and ASP)
GetErrorStatus (<i>Error</i> as ErrorCodes)	Checks the result of a call. - See error code table D5
GetErrorStatusV (<i>Error</i> as ErrorCodes)	Gets error status with variant parameters (see note on VBScript and ASP)
GetResultGUID (<i>Index</i> as Long, <i>GUID</i> as String)	Following search, retrieves GUID of document found - Index of document in result set (0 – ResultCount) - <i>GUID</i> contains the GUID of the indexed document
GetResultGUIDV (<i>Index</i> as Variant, <i>GUID</i> as Variant)	Gets result GUID with variant parameters (see note on VBScript and ASP)

Searches Using RapidocsDocCOM

Field Names

The following table (D2) lists the field names available for use as the first parameter in the Add Criterion methods of the form:

```
AddxxxCriterion("FieldName",SearchType,"Value")
```

for example:

```
TheSearch.AddStringCriterion("author",ssEqual,"John Smith")
```

In this case 'author' is the fieldname that the search will perform matches against. This search will retrieve all documents where the author of the document has the value 'John Smith'.

FieldName	DisplayName	Description	Search Type
author	Author	The author of the document.	String
category	Category	The category the document	String

FieldName	DisplayName	Description	Search Type
		belongs to.	
checked_by	Checked By	The person who checked the document out of the document database.	String
client_client	Client Contact	The contact at the client.	String
client_company	Client Company	The client the document is for.	String
completed_date	Completed Date	The document completion date.	Date
color	Document Colour	The colour of the document. Available only with the function: AddDocColorCriterion	DocumentColor
department	Department	The department	String
doctype	Document Type	The document type, like "rap" or "doc" or "dot", etc...	String
document_creation_date	Document Creation Date	The document creation date.	Date
forward_to	For The Attention Of	The person the document is marked for the attention of	String
import_date	Import Date	The date the document was imported	Date
keywords	Keywords	The keywords	String
language	Language	The language in which the document was written	String
manager	Manager	The manager	String
member_group	Member Group	A member of group	String
name	Name	The name of the document	String
office	Office	The office	String
owner	Owner	The owner of the document	String
Project	Project	The name of the project	String
publication_date	Publication Date	The document publication date.	Date
publisher	Publisher	The name of the document publisher.	String
received_from	Received From	The person who sent the document	String
subject	Subject	The subject of the document	String
publisher_creation_date	Template Creation Date	The date the document's template was created	Date
template_name	Template Name	The template the document was created from	String
checked_out_by	Checked Out By	The person who checked the document out	String
version	Version	The version of the document	String
comments	Document Comments	The document comments	String
copyright	Copyright	The document's copyright	String

Table D2 Field Names for Search Criteria

Note If using a string criterion where the string to search against contains an apostrophe, it is necessary to repeat the apostrophe wherever it occurs. Failure to do this will result in a runtime error. E.g. "John's employer" is written as "John''s employer".

AddDateCriterion

This method is used to specify search criteria involving a date range, for example:

```
TheSearch.AddDateCriterion("publication_date",dsOn,"31.05")
```

This will search for all documents where the Publication Date is the 31st of May (the default year is the current one).

Enumerated Type	Search Type	Format of search string
00	dsToday	None
01	dsYesterday	None
02	dsLastWeek	None
03	dsThisWeek	None
04	dsLastMonth	None
05	dsThisMonth	None
06	dsBetween	"dd.mm.yy, dd.mm.yy" or "dd.mm, dd.mm"
07	dsOn	dd.mm.yy or dd.mm
08	dsOnAfter	dd.mm.yy or dd.mm
09	dsOnBefore	dd.mm.yy or dd.mm
0A	dsInTheLast	dd

Table D2 Search types for AddDateCriterion

AddDocColorCriterion

This method is used to specify search criteria involving a specific document colour, for example:

```
TheSearch.AddDocColorCriterion("color",dcsIs,dcBlue)
```

This will search for all documents where the document colour is blue.

There are only two search types here, `dcsIs` and `dcsIsNot` (enumerated values 00 and 01 respectively).

A list of acceptable search values for `AddDocColorCriterion` is provided below:

Enumerated Value	Value	Document Colour	Document Type
00	dcIllegal	Illegal	N/A
01	dcPink	Pink	Precedent
02	dcRed	Red	Template
03	dcBlue	Blue	Document
04	dcBlack	Black	Unpublished (WIP)

Table D3 Acceptable values for AddDocColorCriterion

AddStringCriterion

This method is used to specify search criteria involving a text string, for example:

```
TheSearch.AddStringCriterion("keywords",
    ssIncludesWords,"Employment Contract")
```

This will search for all documents where the Keywords field include the words Employment Contract.

Enumerated Type	Search Type	Explanation
00	ssEqual	Exactly equal
01	ssNotEqual	Not equal
02	ssIncludesWords	Includes words (delimited by spaces)
03	ssIncludesPhrase	Includes exact phrase
04	ssBegins	Field begins with...
05	ssEnds	Field ends with..

Table D4 Search types for AddDateCriterion

Note (For Tables D2, D3, D4) If the programming language does not generate type libraries you cannot use enumerated types and you must use numbers. However, if you want to use enumerated types you must define the constant yourself.

Using RapidocsDocCOM with ASP or VBScript

ASP and VBScript (and a number of other scripting languages) will only accept the Variant data type for variables. If RapidocsDocCOM is to be used with one of these languages, then care must be taken with certain methods whose parameters will not accept variants. In particular, `ExecuteSearch(ResultCount as Long)`, `GetErrorStatus(Error as ErrorCodes)` and `GetResultGUID(Index as Long, GUID as String)` insist on non-variant data types. For this reason, three additional methods exist `ExecuteSearchV(ResultCount as Variant)`, `GetErrorStatusV(Error as ErrorCodes)` and `GetResultGUIDV(Index as Variant, GUID as Variant)`. In this case all parameters are variants, hence these methods should be employed in the case with ASP or VBScript is being used as an interfacing language.

Error Codes

The method `GetErrorStatus(Error as ErrorCodes)` can be used after a call is made to check the result of the call.

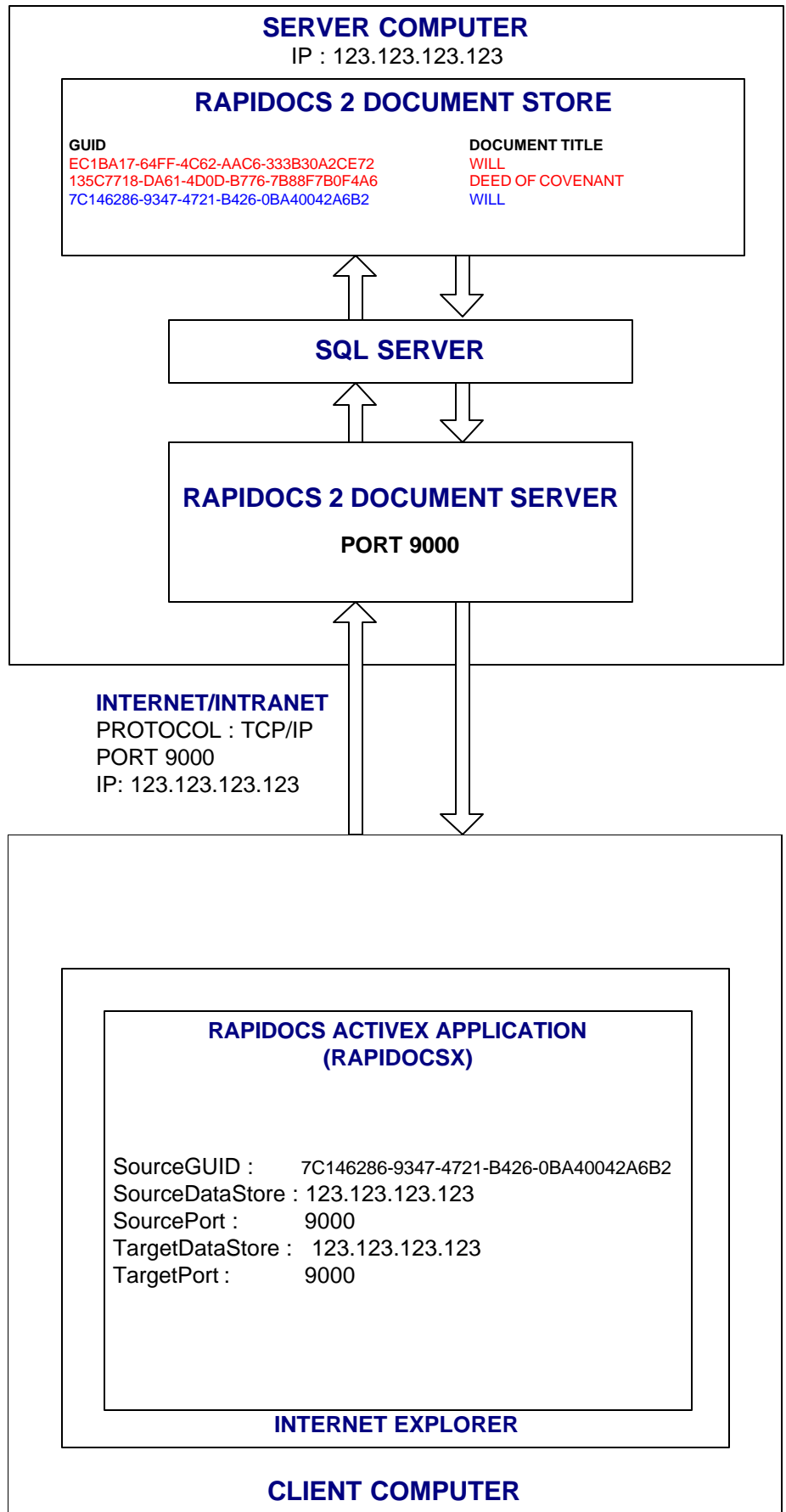
The following table lists the error codes associated with this method.

Error Number	Error Code	Explanation
--------------	------------	-------------

Error Number	Error Code	Explanation
00	ecNoError	No error
01	ecNoConnection	Unable to connect to the specified server
02	ecNoDocument	Unable to locate the specified document
03	ecBadPath	The path specified does not exist
04	ecDuplicate	Attempt to insert a duplicate document into the store
05	ecUndefined	Undefined error, not currently used
06	ecOutOfBounds	Attempt to read a message which is outside the range of the available messages, i.e. read msg 50 from a document with only 49 messages.

Table D5 RapidocsDocCOM Error Codes

Appendix E: RapidocsX Client-Server Overview



Addendum

Using RapidocsX in Netscape

Introduction

As discussed in the chapter on Using RapidocsX, it is possible to embed the Rapidocs Assembler as an ActiveX control within a browser. The drawback with this, as with all ActiveX controls, is that it is only compatible with Internet Explorer. However, a solution is available that overcomes this restriction and enables Netscape 4.x users to open RapidocsX.

Netscape compatibility is achieved using the off-the-shelf Asgard Container Plugin™. This is effectively a wrapper that embeds the ActiveX control within Netscape. The mode of deployment is slightly different than for Internet Explorer so this chapter seeks to describe this process in detail.

Implementing RapidocsN

It is advised that careful attention be paid to the chapter entitled Using RapidocsX before attempting to implement RapidocsX in Netscape, since all of the attributes and parameters are the same in both implementations.

Two files are needed for Netscape deployment: the first one is RapidocsX.cab, which is the cabinet file containing the RapidocsX activex control, the second is npaxctrl.jar which is a Java Archive file used to install the Asgard Container Plugin™ (npaxctrl.dll). This dll file will install itself in the Netscape plugins folder on the client machine.

When a user downloads RapidocsX, the digital signature will be displayed in a similar way with Netscape as if it were Internet Explorer.

Example of Opening a Document from a Store

Below is an example of the JavaScript code (courtesy of Asgard Software) used to install the Container Plugin in the plugins folder. This should be embedded after the <BODY> tag.

```
<script language="JavaScript">
// npaxctrl_Install
//
// installs the Asgard ActiveX Container Plugin
//
// ieObjects - specifies that the page uses OBJECT tags for
IE
// pluginScript = specifies that the page needs to script
against the plugin
//
// returns false if failed
//
// location of jar file for plugin
var npaxctrl_jarurl = "http://yourwebaddress/npaxctrl.jar";
// mime type of plugin to install
var npaxctrl_mimetype = "application/x-pw-oleobject-2.0";
// alternate installation url
var npaxctrl_alturl =
"http://www.asgardsw.com/plgwiz2/npaxctrl.htm";
function npaxctrl_Check()
{
    if (navigator.mimeTypes[npaxctrl_mimetype] != null)
        history.go(0);
    else
        setTimeout("npaxctrl_Check()",1000);
}
function npaxctrl_OnLoad()
{
    setTimeout("npaxctrl_Check()",1000);
}
function npaxctrl_Install(ieObjects, pluginScript)
{
    var agt = navigator.userAgent.toLowerCase();
    var is_major = parseInt(navigator.appVersion);
    var is_nav = ((agt.indexOf('mozilla') != -1) &&
(agt.indexOf('spoofer') == -1)
    && (agt.indexOf('compatible') == -1) &&
(agt.indexOf('opera') == -1)
    && (agt.indexOf('webtv') == -1) &&
(agt.indexOf('hotjava') == -1));
    var is_ie = ((agt.indexOf("msie") != -1) &&
(agt.indexOf("opera") == -1));
    var is_win95 = ((agt.indexOf("win95") != -1) ||
(agt.indexOf("windows 95") != -1));
    var is_win98 = ((agt.indexOf("win98") != -1) ||
(agt.indexOf("windows 98") != -1));
    var is_winnt = ((agt.indexOf("winnt") != -1) ||
(agt.indexOf("windows nt") != -1));
```

```
var is_win32 = (is_win95 || is_winnt || is_win98 ||
              ((is_major >= 4) && (navigator.platform
== "Win32"))) ||
              (agt.indexOf("win32") != -1) ||
              (agt.indexOf("32bit") != -1));
if(ieObjects)
{
    if(is_ie) // browser is ie, return ok
        return true;
}

if (pluginScript)
{
    if(!is_nav || is_major < 4)
    {
        alert("This page requires Asgard ActiveX
Container Plugin scripting support.");
        return false;
    }
    if (navigator.mimeTypes[npaxctrl_mimetype] == null)
    {
        if(is_nav && is_major == 4 && is_win32)
        {
            window.onload = npaxctrl_OnLoad;
            if (!navigator.javaEnabled() )
            {
                alert("Java must be enabled in
order to install the Asgard ActiveX Container Plugin.");
                return false;
            }
            trigger = netscape.softupdate.Trigger;
            if (!trigger.UpdateEnabled())
            {
                alert("SmartUpdate must be
enabled in order to install the Asgard ActiveX Container
Plugin.");
                return false;
            }

            trigger.StartSoftwareUpdate(npaxctrl_jarurl,
trigger.DEFAULT_MODE);
        }
        else if(is_nav && is_major > 4 && is_win32)
        {
            window.onload = npaxctrl_OnLoad;
            if (!InstallTrigger.enabled())
            {
                alert("SmartUpdate
must be enabled in order to install the Asgard ActiveX
Container Plugin.");
                return false;
            }
            InstallTrigger.startSoftwareUpdate(npaxctrl_jarurl);
        }
        else
        {
            window.open(npaxctrl_alturl, "_new");
        }
        if (navigator.mimeTypes[npaxctrl_mimetype] ==
null)
            return false;
    }
}
```

```

        return true;
    }
    npaxctrl_Install(true, true);
</script>

```

The JavaScript runs as soon as the page is loaded, and forces the container plugin to be installed without any need to refresh or restart the browser. The npaxctrl.jar file should be downloadable from the webserver – notice the example address <http://www.yourwebaddress.com/npaxctrl.jar> is given in the above code.

The RapidocsX activeX control can then be embedded as follows anywhere within the HTML code.

```

<embed type="application/x-pw-oleobject"
name="rapX"
attr_classid="clsid:E5C97835-6865-443E-8C33-671D9C71A6D0"
attr_codebase="RapidocsX.cab#version=2,0,2,3069"
attr_ID="rapX"
width="100%"
height="100%"
align="center"
hspace="0"
vspace="0"
param_SourceDataStore="123.123.123.123"
param_SourcePort="9000"
param_TargetDataStore="123.123.123.123"
param_TargetPort="9000"
param_SourceGUID="E1C1BA17-64FF-4C52-AAC6-333B30A2CE72"
></embed>

```

This <EMBED> tag is used to invoke RapidocsX – notice the similarity with the equivalent <OBJECT> tag with RapidocsX (see chapter, Using RapidocsX). The calling parameters used to specify the Rapidocs document, port number etc are given here in the form `param_ParamName`, and the object attributes are of the form `attr_AttributeName`.

It is recommended that a browser-sniffer be deployed as soon as a user reaches a site where RapidocsN and RapidocsX are hosted. This will ensure that the user is directed to the plugin/activeX page most suitable for their browser.

Event Handlers

Event Handlers are also supported in Netscape using the Plugin Container. In order to invoke them, the following format is used:

```

<Script language="JavaScript">
function rapX_OnLoad()
{

```

```
alert('Document now Loaded')  
}  
</Script>
```

Where the function is declared in the form
ObjectName_Event(param1, param2,...)

The ObjectName is set in the <EMBED> tag by the attribute
attr_ID="rapX". The 4 events available are OnLoad, OnSave,
OnUnLoad and OnHelpClick (see the chapter on Using
RapidocsX).