



Compassoft

# **DaCS Programmers Guide**

3.5

Copyright © 2007 Compassoft, Inc.  
All rights reserved

Compassoft, Compassoft and the Compassoft logo are trademarks or registered trademarks of Compassoft, Inc.

Windows, Windows NT and Windows XP are registered trademarks of Microsoft Corporation in the United States, other countries, both. Solaris, Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. Linux is a registered trademark of Linus Torvalds in the United States other countries, or both. UNIX is a registered trademark of the Open Group in the United States, other countries, or both. InstallAnywhere and the InstallAnywhere logo are trademarks or registered trademarks of Zero G Software, Inc. in the Unites States, other countries or both. This product includes software developed by The Apache Software Foundation (<http://www.apache.org/>). Portions of this software were developed at the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana-Champaign. This software contains code derived from the RSA Data Security Inc. MD5 Message-Digest Algorithm, including various modifications by Spyglass Inc., Carnegie Mellon University, and Bell Communications Research, Inc. (Bellcore). Regular expression support is provided by the PCRE library package, which is open source software, written by Philip Hazel, and copyright by the University of Cambridge, England. The original software is available from <ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/>.

All other trademarks are the property of their respective owners.

# Contents

---

<b>PREFACE .....</b>	<b>5</b>
About this Book.....	5
DaCS Documentation .....	5
Conventions.....	5
Contacting Compassoft Support.....	6
<b>INTRODUCTION.....</b>	<b>7</b>
DaCS Interface Workbook.....	8
Interacting with DaCS from an External Process .....	9
Macros .....	12
Changes to Excel Programming .....	13
Viewing the WSDACS Object Model .....	13
Classes.....	14
<b>CAUDIT .....</b>	<b>15</b>
Using CAudit .....	15
CAudit Property Defaults .....	16
CAudit Members .....	16
Action .....	16
AfterChange.....	17
AuditWorksheet.....	17
AutoCompleteTarget .....	18
AutoFillValue.....	18
BeforeChange .....	18
BOF .....	18
ChildName.....	18
CommitChange.....	19
DateTimeStamp .....	19
DateTimeStampString .....	19
EnableAutoFill .....	19
EOF .....	19
FileDirty .....	20
NumberFormatAfter .....	20
NumberFormatBefore .....	20
ObjectName.....	20
ObjectTypeExt.....	21
ParentName.....	21
Position .....	22
PromptAudit.....	22
Read .....	22
Reason .....	22
ReasonMaxLength.....	23
ReasonMinLength .....	23
ReasonSpellCheck.....	23

## Compassoft DaCS Programmers Guide

SetDateTimeStamp .....	23
UserFullName .....	23
UserID .....	23
ValidateReason .....	24
WriteAuditTrail .....	24
<b>CERRLOGGER (LASTERROR) .....</b>	<b>25</b>
Using CErrLogger .....	25
CErrLogger Property Defaults .....	26
CErrLogger Members .....	26
ClassName .....	26
Clear .....	26
Description .....	27
DisplayError .....	27
DisplayErrorMessage .....	27
ErrorFileName .....	28
ErrorFilePath .....	28
GetExcelInformation .....	28
LastError .....	28
Line .....	28
LoadError .....	28
LogError .....	29
Message .....	29
Number .....	29
Routine .....	29
SetDefaults .....	29
SetProperties .....	30
Source .....	30
TimeFormat .....	30
Verbose .....	31
WriteErrorLog .....	31
<b>CFILE (MENU) .....</b>	<b>33</b>
Using CFile .....	33
CFile Members .....	33
AboutDaCS .....	33
ChangeUserPassphrase .....	33
ChangeUserPassword .....	33
ChartWizard .....	34
CloseWorkbook .....	34
CreateNewDaCSWorkbook .....	34
CreateNewWorkbookFromTemplate .....	35
DaCSEnabled .....	36
InsertSheet .....	36
LastError .....	37
MoveCopySheets .....	37
OpenWorkbook .....	37
ProtectWorkbookStructure .....	38
RenameSheet .....	38
SaveWorkbook .....	39
SaveWorkbookAs .....	39

SetSheetProtection .....	39
SetWorkbookProtection .....	40
ShowDaCSAdminPanel .....	41
SignWorksheets .....	41
SynchronizeGhostWorkbook .....	41
UnhideWindow .....	42
UnprotectWorkbookStructure .....	42
WorkbookStructureProtected .....	42
<b>CINSTRUMENT .....</b>	<b>43</b>
Using CInstrument .....	44
CInstrument Property Defaults .....	45
CInstrument Enumerations .....	46
CInstrument Members .....	46
AliasUser .....	46
AliasUserFromDatabase .....	47
CompanyName .....	49
GetProfileObject .....	49
InstrumentID .....	49
InstrumentName .....	50
LastError .....	50
License .....	50
LoadExtendedUserProperties .....	50
LoginProfile .....	50
ParseInstrumentFromProfile .....	51
SoftwareName .....	52
SystemCertificate .....	52
SystemCertificateKey .....	52
SystemID .....	52
SystemPublicKey .....	52
CInstrument Example Code .....	53
<b>CPROFILE .....</b>	<b>55</b>
Profile Structure .....	56
CProfile Enumerations .....	58
Security and Cryptography in CProfile .....	59
CProfile Members .....	59
DecryptEncryptedProfile .....	59
DecryptProfile .....	60
EncryptedProfile .....	60
EncryptedProfileDataFormat .....	60
LastError .....	60
Profile .....	60
ProfileDataFormat .....	60
ProfileDecrypted .....	61
SessionKey .....	61
SessionKeyDataFormat .....	61
<b>CSYSTEM .....</b>	<b>63</b>
Using CSystem .....	63

## Compassoft DaCS Programmers Guide

CSystem Property Defaults .....	64
CSystem Members .....	64
ActiveWorkbookStatus .....	64
AuditSilent .....	65
AuditSilentReason .....	65
Build .....	65
DaCSStatus .....	65
ExcelVersion .....	66
GetAuditObject .....	66
GetInstrumentObject .....	66
Instance .....	67
InstanceOK .....	67
Interactive .....	67
LastError .....	67
Menu .....	67
Parent .....	68
SetInterfaceWorkbookSystemObject .....	68
SetEnvironment .....	68
Shutdown .....	68
SignatureVersion .....	68
Start .....	69
Version .....	69
WorkbookStatus .....	69
<b>CCONSTDACS .....</b>	<b>71</b>
Using CConstDaCS .....	71
CConstDaCS Members .....	71
PROFILE_FIELD_DELIMITER .....	71
PROFILE_FIELD_TAG .....	71
CConstDaCS Enums .....	72
<b>GLOSSARY .....</b>	<b>ERROR! BOOKMARK NOT DEFINED.</b>
<b>INDEX .....</b>	<b>79</b>

# Preface

---

## About this Book

---

The DaCS Programmer's Guide is intended for developers who want to create Excel add-ins, VBA macros, or standalone applications that interact with DaCS.

This guide is organized into two major parts:

- ▶ **Introduction:** provides an overview of the DaCS programming model and describes how external processes can interact with DaCS.
- ▶ **Class Reference:** Describes the classes that are used to communicate with DaCS.



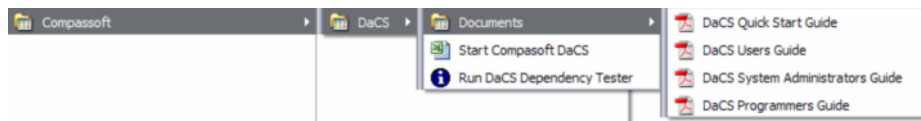
## DaCS Documentation

---

The DaCS documentation set contains four books:

- ▶ **DaCS Quickstart Guide:** describes how to install DaCS and provides basic usage information.
- ▶ **DaCS User's Guide:** describes how to use DaCS and work with DaCS-controlled workbooks.
- ▶ **DaCS Administrator's Guide:** describes how to install and configure DaCS for use in a distributed environment.
- ▶ **DaCS Programmer's Guide:** describes how to customize and extend DaCS by modifying the Interface Workbook and interact with DaCS from other applications, Excel Add-ins, or VBA macros.

To view the documentation, select **All Programs > Compassoft > DaCS > Documents** from the Windows Start menu.



## Conventions

---

The following typographical conventions are used in Compassoft documentation:

- ▶ *italic text* is used to highlight file and path names, web and email addresses, references to other documents, and terms introduced for the first time. It is also used in command examples to indicate values that you need to replace with your own data.
- ▶ **bold text** is used to highlight button, menu, and field names, as well as list items.
- ▶ `monospace text` is used for command and code examples. Values that you need to replace with your own data are italicized.

## **Contacting Compassoft Support**

---

Compassoft strives to serve you better. If you have any feedback or comments on this document, fill in the support request form at [www.compassoft.com](http://www.compassoft.com) or send an email to [support@compassoft.com](mailto:support@compassoft.com). Please mention the version of the application and title of the document in your message.

You can also send us your comments by mail at:

Compassoft Corporation  
1800 Green Hills Road  
Scotts Valley, CA 95066

## Introduction

---

The DaCS programming model enables you can extend the functionality of DaCS, change the way DaCS procedures execute, and automate tasks. To interact with DaCS programmatically, you need to understand the basic operational concepts described in this chapter.

### In This Chapter

DaCS Interface Workbook.....	8
Interacting with DaCS from an External Process .....	9
Macros .....	12
Changes to Excel Programming .....	13
Viewing the WSDACS Object Model .....	13
Classes .....	14

### DaCS Interface Workbook

---

The DaCS Interface Workbook (IWB) acts as a bridge between Excel and DaCS. Under most circumstances, the IWB is used to start DaCS.

The IWB contains DaCS configuration parameters as well as a collection of VBA macros that communicate with the DaCS executable, WSDACS.EXE. The Interface Workbook makes it possible to:

- ▶ Link CommandBarControls and Shortcut Keys to DaCS procedures.
- ▶ Store configuration parameters in a secure, audited file.
- ▶ Allow users to choose between controlled (DaCS) and regular Excel.
- ▶ Allow administrators to set up different controlled environments.
- ▶ Allow programmers to extend or customize DaCS functions.
- ▶ Allow programmers to incorporate DaCS with their own programs.

Using the IWB as a bridge between Excel and DaCS is necessary because CommandBarControls and Shortcut Keys cannot point directly to the DaCS executable; they must point to VBA macros. These VBA macros are defined in the code module MInterface.

The link between the IWB and DaCS is established through a reference to the main object in WSDACS.EXE, CSystem. In the MMain module in the IWB, an object is declared as a publicly-available object of the type CSystem. For example:

```
Public gobjDaCS as CSystem
```

As long as the IWB is open, the declared CSystem object (in this case `gobjDaCS`) will have scope, and all procedures in the IWB will be able to access it.

**Note:** The name of the CSystem object does not have to be `gobjDaCS`. In addition, the object used to communicate with WSDACS can alternatively be declared as the generic type `Object`. Declaring as `Object` has some important implications, which are described later.

The Interface Workbook uses the declared CSystem object to hold a reference to the WSDACS `CSystem` object, start an instance of DaCS, and bind the Interface Workbook to it (which binds the instance of Excel to the instance of DaCS). The CSystem object's reference to the running instance of DaCS enables the VBA macros to communicate with DaCS.

It is the `StartDaCS` macro in `MMain` that actually creates the instance of DaCS, binds the objects, and runs DaCS. When the macro completes, DaCS will be running and controlling Excel and the declared CSystem object will refer to `WSDACS.CSYSTEM`.

## Interacting with DaCS from an External Process

---

The DaCS programming model enables a variety of external processes to interact with DaCS, including:

- ▶ Stand-alone programs written in languages such as C++, Java, or VB
- ▶ Excel Add-Ins (XLA)
- ▶ VBA macros in an XLS workbook

Multiple processes can interact with DaCS at the same time by obtaining references to the running instance of DaCS. For example, you might have a data analysis add-in and an analytical instrument that exports data to Excel, and both programs need to control DaCS.

While `gobjDaCS` is available to any procedure in the Interface Workbook, it is not available to any external process. However, it can be accessed through one of two methods.

The first way to obtain a reference is through the following bit of code. It is added to the `ThisWorkbook` class module of the Interface Workbook:

```
Public Property Get DaCSInterface() As CSystem
    Set DaCSInterface = gobjDacS
End Property
```

(Note that the type declaration may be `As Object` instead of `As CSystem`.)

Now any external process can obtain a reference to DaCS by setting an object to the Interface Workbook's `DaCSInterface` property. In practice, what you would normally do is run the following loop:

```
Dim wb as Object
On Error Resume Next
For each wb in Application.Workbooks
    Set oDaCS = wb.DaCSInterface
    If Not oDaCS is Nothing then Exit For
Next wb
On Error Goto 0
```

Here's how the code works: First, `wb` is declared as an `object` (not a `Workbook`). Before we start, we want to ignore any errors because any other workbook that is open will cause an error when you try to access its `DaCSInterface` property- it doesn't have one. Only the Interface Workbook has the property `DaCSInterface`. Now we loop through all of

## Compassoft DaCS Programmers Guide

the `Workbook` objects in Excel's `Application` object, trying to set our object `oDaCS` to that workbook's `DaCSInterface` property. When (if) we get to the Interface Workbook, and DaCS is running, `oDaCS` will be set to the running instance of `CSystem`, so we can exit the loop. When finished, turn error handling back on.

The above method can be used with add-ins and VBA macros. It can also be used with VB and other languages where you can declare generic `Object` types, and this is why: the Interface Workbook's `Workbook` object has been extended to include `DaCSInterface` as a custom property. It no longer fits the type definition for `Workbook`. Compilers will complain because `DaCSInterface` isn't listed as an available property of their definition for an `Excel.Workbook` object. However, the generic `Object` type has no definition and will be usable.

After you have your reference to DaCS, your program can do what it wants. You need to be concerned with shutdown, because all references need to be released before the Interface Workbook can be closed and DaCS can terminate. This is generally accomplished through a procedure similar to the following, which iterates through all workbooks requesting that they set their internal references to DaCS to `Nothing`:

```
Public Sub ReleaseExternalDaCSReferences()  
  
    Dim wb As Object  
  
10    On Error Resume Next  
  
20    Application.EnableCancelKey = xlDisabled  
  
30    For Each wb In Application.AddIns  
  
40        Workbooks(wb.Name).DaCSRequestRelease  
  
50    Next wb  
  
60    Set wb = Nothing  
  
70    For Each wb In Application.Workbooks  
  
80        Workbooks(wb.Name).DaCSRequestRelease  
  
90    Next wb  
  
100    Set wb = Nothing  
  
End Sub
```

You'll note that it is similar to the code to set a reference; in this case you put the following code in your workbook or add-in's `Workbook` class as a public function:

```
Public Function DaCSRequestRelease As Boolean  
  
    Set oDaCS = Nothing  
  
    DaCSRequestRelease = oDaCS Is Nothing
```

End Function

If the function is present, it is executed to set that workbook's reference to Nothing. If it isn't present, the error is skipped over.

Here's how the whole scenario works in practice. Keep in mind that the "someone" below can be the user, an add-in, or an external program.

1. Someone opens the Interface Workbook.
2. The Interface Workbook procedure `MMain.StartDaCS` instantiates `WSDACS.CSystem` and starts DaCS.
3. External processes obtain their references from the Interface Workbook.
4. Programs run.
5. DaCS shuts down.
6. External processes are requested to release their references.
7. The Interface Workbook is closed.

This all works fine with workbooks and add-ins, because the Interface Workbook can be set up to call the `ReleaseExternalDaCSReferences` function when DaCS quits. But it can't call external programs; for that you would need to trap Excel's events to check if the Interface Workbook is being closed, or Excel is shutting down. (The next version of DaCS should remedy this by having a `BeforeQuit` event you can trap.)

The only problem with referencing so far is that it won't work with languages that require early binding, like C++, where you can't declare a variable as `Object`. You can still set a reference to `objDaCS`, but you have to do it in a backwards sort of way: the external program needs to start DaCS itself, then bind an Interface Workbook to it. Here is how it would work:

1. The program instantiates `WSDACS.CSystem`.
2. The program instantiates Excel.
3. The program opens the Interface Workbook.
4. The program calls `CSystem.SetInterfaceWorkbookSystemObject`, passing in a reference to the Interface Workbook.
5. The program binds DaCS to the Interface Workbook by setting `CSystem.Instance` to the Interface Workbook.
6. The program (not the Interface Workbook) starts DaCS by calling `CSystem.Start`.

### Macros

---

When creating programs that work with DaCS, one of the things you need to keep in mind is the effect of what you are doing on the audit trail. You want to make sure all changes are captured so that they can be "reversed", while minimizing the size of the audit trail. The "reversibility" of the audit trail means that a person (or program) can read backwards through the audit trail and "undo" all the changes in it. By minimizing the audit trail, we want to write as few changes to it as possible. One way to do this is with the bulk importation of data. If a program is bringing over or calculating a large amount of data, it is common to write a single entry in the audit trail for this.

Keep in mind that the 21 CFR 11 requirements for audit trails is that they apply to changes the operator makes, not your program. So while the operator has control over the spreadsheet, DaCS will track changes. While your program has control, you can either let DaCS do the auditing or turn it off and write the audit trail yourself.

There are two ways to take care of auditing while your program is running. You can either use Silent Audit or you can handle the audit trail yourself. Silent Audit means that DaCS still tracks every cell change, but never prompts the user for a reason for the change; whatever you tell DaCS the reason is, will be used. See the sections on `CSystem.AuditSilent` and `CSystem.AuditSilentReason` for details.

The second method is handling the audit trail yourself. This is done by turning Excel's events off so that DaCS doesn't see what you're doing. Then you use the `CAudit` object to write to the audit trail yourself. When finished, you turn Excel's events back on. This allows you to do things like paste a large amount of data into a worksheet and write just one audit trail entry. Here is some example code:

```
Application.EnableEvents = False

Range("A1").Select

Selection.Paste

WriteAuditTrail "Sheet1", "A1:C100", "Imported Data"

Application.EnableEvents = True
```

In the above example, we have another macro in our workbook to handle using `CAudit` called `WriteAuditTrail`. It's a good idea to have one, to reduce the amount of code you have to write.

## Changes to Excel Programming

---

If you're writing code to use with DaCS, you need to do a few things differently from regular Excel VBA code. In some cases, you need to replace your regular code with DaCS functions. The particular functions you need to replace are found under the `CFile` documentation. In general, they deal with the workbook structure. In DaCS, the workbook structure is always protected. (It's the only way to keep users from renaming sheets, since they can double-click on sheet tabs and rename them.) However, the workbook may not "appear" to be protected to the user, allowing them to use the **Rename Sheet** and **Move** or **Copy Sheets** menu commands. Other functions in `CFile` are available as a convenience, as they will take care of the ghost workbook and the audit trail for you.

If you're doing anything with Excel's events, you need to keep in mind when the event will fire in your code in relation to the Interface Workbook and DaCS. Exactly when will depend on the position of each in Excel's event call stack.

## Viewing the WSDACS Object Model

---

The WSDACS library and its members can be viewed using the VB or VBA Object Browser.

**Classes**

---

The following classes appear in the WSDACS library, however not all of them are intended to be used. The italicized classes in the table below are either cryptographic classes not available to the programmer, or contain no functionality, and are not described in this manual.

<b>Class</b>	<b>Purpose</b>
<i>CAlgInfo</i>	Cryptography algorithm handler.
<i>CExcelLibrary</i>	Excel® Application stub. No inherent functionality.
CAudit	DaCS audit trail handler. Used to read/write to workbook audit trail.
CConstDaCS	DaCS constants for error codes, profile structure.
<i>CConstWCCO</i>	Cryptography constants
<i>CErrorHandler</i>	Cryptography error handler
CErrLogger	DaCS error handler
CFile	DaCS menu items and commands
CInstrument	DaCS instrument interface
CProfile	DaCS instrument profile manager
CSystem	DaCS application manager

## CAudit

---

### In This Chapter

Using CAudit .....	15
CAudit Property Defaults .....	16
CAudit Members .....	16

### Using CAudit

---

CAudit is used to read from or write to the audit trail (the `_AUDIT` worksheet) of a DaCS workbook. It is preferred when you need a CAudit object that you access it through the CSystem procedure `GetAuditObject`.

#### Preferred Method:

```
Dim oAudit as Object

bln = [CSystem].GetAuditObject(o_oAudit:=oAudit)
```

#### Late Binding:

```
Dim oAudit As Object

Set oAudit = CreateObject("WSDACS.CAudit")
```

#### Early Binding (requires Reference to WSDACS.EXE)

```
Dim oAudit As CAudit

Set oAudit = New CAudit
```

You can perform multiple `WriteAuditTrail` or `Read` calls with a single instance of CAudit. However, you must never perform a `Read` followed by a `WriteAuditTrail`, or the `UserFullName` and `UserID` may be corrupted. Just set the instance of CAudit to `Nothing`, and create a new instance.

You should set the instance of CAudit to `Nothing` as soon as you are finished, because it will always point to the audit worksheet of the active workbook when the object was created.

It is not recommended that you keep multiple instances of CAudit open at the same time. CAudit was designed to be used either for a single read or write operation, then immediately destroyed.

The CAudit object and its members can be viewed using the VB or VBA Project Explorer.

## CAudit Property Defaults

---

The following properties are set on creation of an instance of a `CAudit` object. What the properties are depends on if `CSystem.Instance` has been set to an Interface Workbook or not:

Property	Not Set	Set
ReasonMinLength	0	6 or AUDIT.DATA.REASONS.MIN_LENGTH
ReasonMaxLength	0	255 or AUDIT.DATA.REASONS.MAX_LENGTH
ReasonSpellCheck	FALSE	FALSE or AUDIT.DATA.REASONS.SPELLCHECK
UserFullName	Empty	Full Name of logged-in user
UserID	Empty	UserID of logged-in user
AuditWorksheet	Nothing	ActiveWorkbook.Worksheets("_AUDIT")
Position	0	AuditWorksheet.UsedRange.Rows.Count
BOF	-1	-1 only if Position < 2
EOF	0	-1 only if Position > 65526
Date Format	Empty	"DD-MMM-YYYY HH:MM:SS" or SYSTEM.TIME_FORMAT

## CAudit Members

---

### Action

Type: Property  
Format: String  
I/O: Read/Write  
Description: Returns/sets the Action field. Text descriptor of what happened. You can put anything in this field you like. Common entries are as follows:

Action	When Used
AliasUserID	Aliasing a UserID
Create	Enter cell data, inserting charts, inserting sheets, creating a new workbook, creation of parameter, copy sheet
Delete	Deleting cell data
ERROR	Entry in audit trail for override of corrupt digital signature
Import	Conversion of regular file to DaCS format
Merge	Merging a range
Modify	Change cell data, change metadata parameter, change protection, rename sheet
Moved sheet	Moving a sheet's location
Open	Opening a file
Save	Saving the file
Sign	Digital or electronic signature
Unmerge	Unmerging a range

### AfterChange

Type: Property

Format: Variant

I/O: Read/Write

Description: The final value of the target; that is, what is after the audit trail has been written.

### AuditWorksheet

Type: Property

Format: Object (*Worksheet*)

I/O: Read-Only

Description: Pointer to the "\_AUDIT" worksheet.

Notes: Currently only allows you to read from the current "\_AUDIT" worksheet of a workbook.

## AutoCompleteTarget

Type: Property  
Format: String  
I/O: Write-Only  
Description: Allows you to set the text displayed in the "Automatically complete remaining entries in:" text box on the Audit Trail form.

## AutoFillValue

Type: Property  
Format: Long  
I/O: Read-Only  
Description: Allows you to read the "Automatically complete remaining entries in:" checkbox on the Audit Trail form. 0 = unchecked, 1 = checked. Default is 0 (unchecked).

## BeforeChange

Type: Property  
Format: Variant  
I/O: Read/Write  
Description: The previous value of the target; that is, what was before the audit trail was written.

## BOF

Type: Property  
Format: Long  
I/O: Read-Only  
Description: Set when the `CAudit` object is created or a new "\_AUDIT" worksheet is created. Will return -1 if the `Position` is < 2 or a new "\_AUDIT" worksheet has been created. Will return 0 otherwise. Use when reading the audit trail to determine if you have reached the start or you need to switch to another worksheet.

## ChildName

Type: Property  
Format: String  
I/O: Read/Write  
Description: "Child" field (Column F) in the audit trail. Name of the lowest-granularity object the change directly applies to. If the `ChildName` is sufficient to distinctly identify the target, then `ObjectName` and `ParentName` do not need to be specified. For example, if the action applies to range \$B\$6 on worksheet "\_SYSTEM", the `ChildName` is set to "\$B\$6" and `ObjectName` is set to "\_SYSTEM".

**CommitChange**

Type: Property  
 Format: Boolean  
 I/O: Read/Write  
 Description: Returns whether or not the user accepted the change on the Audit Trail form (`TRUE`) or cancelled it (`FALSE`).

**DateTimeStamp**

Type: Property  
 Format: Date  
 I/O: Read-Only  
 Description: The date/time of the audit trail entry, "Local Time" (Column A). Only set when you read an audit trail entry (`CAudit.Read`), write to the audit trail (`CAudit.WriteAuditTrailEntry`), or use `CAudit.SetDateTimeStamp`. The date returned will be as interpreted by the operating system, so be wary of date formatting issues. `DateTimeStampString` is a safer choice when reading the audit trail, as it will return the `.Text` property of the cell.

**DateTimeStampString**

Type: Property  
 Format: String  
 I/O: Read-Only  
 Description: The `.Text` property of the date/time stamp cell in the audit trail, "Local Time" (Column A). Only set when using the `CAudit.Read` function.

**EnableAutoFill**

Type: Property  
 Format: Boolean  
 I/O: Write-Only

Description: Allows you to enable or disable the "Automatically complete remaining entries in:" check box on the Audit Trail form. Default is `FALSE`.

Notes: To read the value of the check box, use `AutoFillValue`.

**EOF**

Type: Property  
 Format: Long  
 I/O: Read-Only  
 Description: Set when the `CAudit` object is created, `CAudit.Position` is set, or a new `_AUDIT` worksheet is created. Will return `-1` if

## Compassoft DaCS Programmers Guide

`CAudit.Position` is > 65526 or exceeds the `CAudit.AuditWorksheet.UsedRange.Rows.Count` property. (Note that `CAudit.WriteAuditTrail` will set `CAudit.Position` to the last row, but `EOF` will remain 0.) If `CAudit.Position` is "within bounds", then `EOF` will return 0. Use when reading the audit trail to determine if you have reached the end of the audit trail or need to switch to another worksheet.

### FileDirty

Type: Property  
Format: Boolean  
I/O: Read-Only  
Description: Determines if the file has changed (is dirty) because of actions rolled back on the audit log. Not implemented in current versions of DaCS.

### NumberFormatAfter

Type: Property  
Format: String  
I/O: Read/Write  
Description: The number format (`Range.NumberFormat` property) of the cell for the `AfterChange` value. When `CAudit.WriteAuditTrail` is called, the `NumberFormat` of the cell in column K of the audit trail will be set to this value. However, if the value is a formula or the length of the entry is > 255 characters, the `NumberFormat` will default to "General".

### NumberFormatBefore

Type: Property  
Format: String  
I/O: Read/Write  
Description: The number format (`Range.NumberFormat` property) of the cell for the `BeforeChange` value. When `CAudit.WriteAuditTrail` is called, the `NumberFormat` of the cell in column J of the audit trail will be set to this value. However, if the value is a formula or the length of the entry is > 255 characters, the `NumberFormat` will default to "General".

### ObjectName

Type: Property  
Format: String  
I/O: Read/Write  
Description: "Object" field (Column E) in the audit trail. Name of the parent object of the `Child` object the change directly applies to. If the `ChildName` is sufficient to distinctly identify the target, then

`ObjectName` and `ParentName` do not need to be specified. For example, if the action applies to range `$B$6` on worksheet `"_SYSTEM"`, the `ChildName` is set to `"$B$6"` and `ObjectName` is set to `"_SYSTEM"`.

## ObjectTypeExt

Type: Property  
 Format: wsCAudit\_ObjectType (Long)  
 I/O: Read/Write  
 Description: Sets or returns the "Type" field of the audit trail (column G). This field is included so that automatic processing routines may determine what the change applies to. The number written is additive; use bitwise `AND` operations to screen for objects types. When you set this property through a macro, it will always add `wsCAudit_External` to the value. If the user is aliased, then `wsCAudit_Aliased` is always added to the value. Values defined are:

wsCAudit_ObjectType	Value	Hex
wsCAudit_External	1	&H1
wsCAudit_System	2	&H2
wsCAudit_Workbook	4	&H4
wsCAudit_Worksheet	8	&H8
wsCAudit_ChartSheet	16	&H10
wsCAudit_ChartObject	32	&H20
wsCAudit_Series	64	&H40
wsCAudit_Range	128	&H80
wsCAudit_Cell	256	&H100
wsCAudit_Signature	512	&H200
wsCAudit_Aliased	1024	&H400

## ParentName

Type: Property  
 Format: String  
 I/O: Read/Write

## Compassoft DaCS Programmers Guide

Description: "Parent" field (Column D) in the audit trail. Name of the parent of the `Object` the change directly applies to. If the `ObjectName` is sufficient to distinctly identify the target, then `ParentName` does not need to be specified. In most cases, `ParentName` will remain blank. It was originally added for support of auditing `ChartObjects`.

### Position

Type: Property  
Format: Long  
I/O: Read/Write  
Description: Returns or sets the row number of the audit trail. Used in conjunction with `CAudit.Read`. If the row is `< 2` then `BOF` will be set to `TRUE`, while if the row is `> 65526` or past `CAudit.AuditWorksheet.Usedrange.Rows.Count`, then `EOF` will be set to `TRUE`. In both cases the actual `Position` will be left unchanged. If the row is valid, then `Position` will be set to the input row number and `BOF` and `EOF` will both be set to `FALSE`. Note this does not affect `CAudit.WriteAuditTrail`.

### PromptAudit

Type: Function  
Format: Boolean  
Description: Displays the Audit Trail form. Returns `TRUE` unless there was an error showing the form. Use `CAudit.CommitChange` to find out if they accepted or reverted the change.

### Read

Type: Function  
Format: Boolean  
Description: Reads a line from the audit trail, specified by `CAudit.Position`. If `BOF` or `EOF` are `TRUE`, or if `AuditWorksheet` is `Nothing`, then this function will return `FALSE`. The audit trail entry is loaded into the `CAudit` object. The following properties are set: `DateTimeStamp`, `DateTimeStampString`, `UserFullName`, `UserID`, `ParentName`, `ObjectName`, `ChildName`, `ObjectTypeExt`, `Action`, `Reason`, `BeforeChange`, `AfterChange`. Other properties are left as-is.

### Reason

Type: Property  
Format: String  
I/O: Read/Write  
Description: The "Reason" field (column I) in the audit trail. This is free-entry text describing why the entry was made.

**ReasonMaxLength**

Type: Property  
 Format: Long  
 I/O: Read/Write  
 Description: Maximum allowable length for the `Reason` field on the Audit Trail form. Default is the parameter `AUDIT.REASONS.MAXLENGTH` from the Interface Workbook `SETTINGS` worksheet.

**ReasonMinLength**

Type: Property  
 Format: Long  
 I/O: Read/Write  
 Description: Minimum allowable length for the `Reason` field on the Audit Trail form. Default is the parameter `AUDIT.REASONS.MINLENGTH` from the Interface Workbook `SETTINGS` worksheet.

**ReasonSpellCheck**

Type: Property  
 Format: Boolean  
 I/O: Read/Write  
 Description: If the `Reason` field on the Audit Trail form is spell-checked. Default is the parameter `AUDIT.REASONS.SPELLCHECK` from the Interface Workbook `SETTINGS` worksheet.

**SetDateTimeStamp**

Type: Property  
 Format: Date  
 I/O: Read-Only  
 Description: When this property is returned, it will set `CAudit.DateTimeStamp` to `Now()`. This allows the programmer to lock in (and return) the timestamp in the audit trail.

**UserFullName**

Type: Property  
 Format: String  
 I/O: Read-Only  
 Description: The full name of the user that appears in the "User" field of the audit trail, column B.

**UserID**

Type: Property  
 Format: String

## Compassoft DaCS Programmers Guide

I/O: Read-Only  
Description: The UserID that appears in the "UserID" field of the audit trail, column C.

### ValidateReason

Type: Function  
Format: Boolean  
Description: Returns the results of a validation check on `CAudit.Reason`. This will check the length of the `Reason` against `ReasonMinLength` and `ReasonMaxLength`. If `ReasonSpellCheck` is TRUE, then the `Reason` will be spell-checked. If validation fails, then a prompt will be displayed to the user. `CAudit.PromptAudit` calls this function.

### WriteAuditTrail

Type: Function  
Format: Boolean  
Description: Writes a new entry to the audit trail. Returns TRUE unless there was an error (or `AuditWorksheet` is Nothing). Increments `Position`. If this puts `Position` past 65526, then a new audit worksheet is created. Writes `DateTimeStamp`, `UserFullName`, `UserID`, `ParentName`, `ObjectName`, `ChildName`, `ObjectType`, `Action`, `Reason`, `BeforeChange`, `AfterChange`.

Notes: The `NumberFormat` of `DateTimeStamp` is derived from the `SYSTEM.DATEFORMAT` parameter in the Interface Workbook "SETTINGS" worksheet. The default is "DD-MMM-YYYY HH:MM:SS".

The `Range.NumberFormat` property of the cells for `ParentName`, `ObjectName`, and `ChildName` will be set to `Text` when that property is numeric. (For example, a sheet named "1234". This avoids a problem with the displayed field changing to scientific notation for very large numbers.)

The `Range.NumberFormat` property of the cells for `BeforeChange` and `AfterChange` will be set to "General" when that property is a formula or the length is > 255 characters. This prevents errors with recording formulas or large amounts of text.

NEVER call `WriteAuditTrail` if you call `Read`, as it will use the `UserFullName` and `UserID` that was last read. Always use a new object instance.

## CErrLogger (LastError )

---

### In This Chapter

Using CErrLogger.....	25
CErrLogger Property Defaults.....	26
CErrLogger Members.....	26

### Using CErrLogger

---

This class is implemented as the `LastError` object of the other DaCS classes. It serves to extend Visual Basic's `Err` object by adding an error logging feature. You would generally only use it to determine what error DaCS encountered after a procedure failure. For example, if your call to `CInstrument.LoginProfile` returns `False`, you would probably find that `CInstrument.LastError.Description` would be `INSTRUMENT_ERROR_PROFILE_NOT_LOADED`.

## CErrLogger Property Defaults

---

Property defaults are set by the parent class when the `CErrLogger` object is instantiated. At that time, the `ClassName` is set to the name of the parent class. When the parent class loads an error, it calls `LoadError`, which sets the properties as required to populate the class.

Property	Default
<code>ClassName</code>	Empty
<code>Description</code>	Empty
<code>DisplayError</code>	TRUE
<code>ErrorFileName</code>	Empty
<code>ErrorFilePath</code>	Empty
<code>LastError</code>	Empty
<code>Line</code>	0
<code>Message</code>	Empty
<code>Number</code>	0
<code>Routine</code>	Empty
<code>Source</code>	"DaCS"
<code>TimeFormat</code>	Empty
<code>Verbose</code>	Empty

## CErrLogger Members

---

### ClassName

Type: Property  
Format: String  
I/O: Read/Write  
Description: Name of the parent class of `CErrLogger`.

### Clear

Type: Function  
Format: Boolean  
Syntax: `Boolean = Clear ( )`

Description: Sets class properties as follows:

Property	Value
Number	0
Description	Empty
Line	0
Source	"DaCS"
Verbose	Empty
Routine	Empty
DisplayError	CSystem.Interactive
Message	Empty

Arguments: None

### Description

Type: Property

Format: String

I/O: Read/Write

Description: `Err.Description` or the `DACS_ERROR` Enum string.

### DisplayError

Type: Property

Format: Boolean

I/O: Read/Write

Description: If set to `TRUE`, the error will be displayed to the user. If `FALSE`, the error message will not be displayed to the user. Note that if `CSystem.Interactive = FALSE`, the error message will not be displayed to the user.

### DisplayErrorMessage

Type: Function

Format: Boolean

Syntax: `Boolean = DisplayErrorMessage ( )`

Description: Displays the error message to the user in a `MessageBox` with the `Title` "DaCS Error". The message displayed is the `Message` property followed by `Verbose`.

Arguments: None

Note: If the error log was not written, the error message will be prefixed by the following text (not in italics, however):

There was a problem writing the error log. Please write down following error message and contact your system administrator:

### **ErrorFileName**

Type: Property  
Format: String  
I/O: Read/Write  
Description: Name of the error file to write to. Set by `SetDefaults` and `SetProperties`.

### **ErrorFilePath**

Type: Property  
Format: String  
I/O: Read/Write  
Description: Path of the error file to write to. Set by `SetDefaults` and `SetProperties`.

### **GetExcelInformation**

Type: Function  
Format: Boolean  
Syntax: `Boolean = GetExcelInformation ( )`  
Description: Sets the `FileFullName`, and Excel version/build.  
Arguments: None  
Note: Always returns FALSE

### **LastError**

Type: Property  
Format: Variant  
I/O: Read-Only  
Description: Default property of class. Returns the same value as `Verbose`.

### **Line**

Type: Property  
Format: Long  
I/O: Read/Write  
Description: Line number where the error occurred in the procedure.

### **LoadError**

Type: Function  
Format: Boolean  
Syntax: `Boolean = LoadError( _`

```
ByVal i_oErr As ErrObject, _  
ByVal i_sRoutine As String _  
)
```

Description: Loads an `Err` object into the class. Parses the `Err.Description` property; if it corresponds to a `DACS_ERROR` Enum from `CConstDaCS`, the property `CErrLogger.Description` will be set to the Enum string. Clears the `Err` object and then executes `LogError`.

Arguments: `i_oErr` Required `Err` object.  
`i_sRoutine` Required String. Name of the procedure generating the error.

## LogError

Type: Function  
Format: Boolean  
Syntax: `Boolean = LogError ( )`  
Description: Executes the following members of the class, in order: `SetDefaults`, `SetProperties`, `GetExcelInformation`, `WriteErrorLog`, `DisplayErrorMessage`.  
Arguments: None

## Message

Type: Property  
Format: String  
I/O: Read/Write  
Description: Informational error text for the user.

## Number

Type: Property  
Format: Long  
I/O: Read/Write  
Description: `Err.Number`

## Routine

Type: Property  
Format: String  
I/O: Read/Write  
Description: Procedure generating the error.

## SetDefaults

Type: Function  
Format: Boolean  
Syntax: `Boolean = SetDefaults ( )`

## Compassoft DaCS Programmers Guide

Description: Sets the following defaults for the class:

Property	Value
TimeFormat	"dd-mmm-yyyy hh:mm:ss"
UserFullName	"Unknown"
Location	"Unknown"
ErrorFileName	"dacserror.txt"
ErrorFilePath	App.Path
FileFullName	"None"
ExcelVersion	Empty
ExcelBuild	Empty

Arguments: None

Note: Always returns FALSE. The italicized entries above are private variables for the class, used in creating the error log entry text.

## SetProperties

Type: Function

Format: Boolean

Syntax: `Boolean = SetProperties ( )`

Description: Sets the following properties of the class to the global properties of DaCS: *TimeFormat*, *UserFullName*, *Location*, *ErrorLogName*, *ErrorFilePath*. Sets the *TimeOfError* to be `Now()`. Sets the *Source* to be the `App.Title.Class.Routine`. Sets *Verbose* to be the *Number* in decimal and hexadecimal format, the *Description*, *Line*, and *Source*.

Arguments: None

Note: Always returns FALSE

## Source

Type: Property

Format: String

I/O: Read/Write

Description: Default is `App.Title ("DaCS")`, until `SetProperties` sets it to `App.Title.ClassName.Routine`.

## TimeFormat

Type: Property

Format: String

I/O: Read/Write  
Description: Time format to use in the error log for the *date/time*. Set by `SetDefaults` and `SetProperties`.

### Verbose

Type: Property  
Format: String  
I/O: Read/Write  
Description: The `Number` in decimal and hexadecimal format, the `Description`, `Line`, and `Source`, concatenated as a string.

### WriteErrorLog

Type: Function  
Format: Boolean  
Syntax: `Boolean = WriteErrorLog( )`  
Description: Writes the properties of the `CErrLogger` class to the error log. The error log entry takes the following format, where italicized entries are variable, and `Verbose` and `Message` are the class properties:

\*\*\*\*\*

```
TimeOfError  
Verbose  
Error Message: Message  
DaCS Version 3.0.1.x build 20031113   Environ("OS")  
   Excel Version  
Running Excel ExcelVersion build ExcelBuild  
User: UserFullName Location: Location File: FileFullName
```

The "Running Excel" line will only be written if Excel is running.

Arguments: None



## CFile (Menu)

---

### In This Chapter

Using CFile .....	33
CFile Members .....	33

### Using CFile

---

`CFile` is used to execute various DaCS commands. Generally these are commands the user selects from the menu (hence the notation `CSystem.Menu`.) Access this object from you instance of `CSystem`. The Menu object is available after the Instance property of `CSystem` has been set.

Example: `goDaCS.Menu.DaCSEnabled`

The `CFile` object and its members can be viewed using the VB or VBA Project Explorer.

**Note:** Make sure the workbook you're working on is the ActiveWorkbook! Otherwise, the audit trail for actions may be written to whatever other workbook is active.

### CFile Members

---

#### AboutDaCS

Type: Subroutine  
Format: N/A  
Syntax: `AboutDaCS ()`  
Description: Shows the About DaCS dialog.

#### ChangeUserPassphrase

Type: Function  
Format: Boolean  
Syntax: `Boolean = ChangeUserPassphrase ()`  
Description: Shows the form for changing the user's passphrase.

#### ChangeUserPassword

Type: Function  
Format: Boolean  
Syntax: `Boolean = ChangeUserPassword ()`

## Compassoft DaCS Programmers Guide

Description: Shows the form for changing the user's password.

### ChartWizard

Type: Subroutine

Format: N/A

Syntax: ChartWizard ()

Description: Shows the ChartWizard. (Comments from source code: Controls 1957 and 436 are the same; they both invoke the Chart Wizard, and executing the control is the only way to get the wizard to show correctly (can't do `Dialogs().Show`). 1957 is remapped to this sub, 436 is not made available to the user.)

### CloseWorkbook

Type: Function

Format: Boolean

Syntax: `Boolean = CloseWorkbook ( _  
[i_bln_BeforeCloseEvent As Boolean = False] _  
)`

Description: Closes the `ActiveWorkbook`. Saves, closes, or prompts per the following table:

	CSystem.	CFile.	ActiveWorkbook.
<b>Saves?</b>	ActiveWorkbookStatus	DaCSEnabled	Saved
YES	2	TRUE	FALSE
NO	2	TRUE	TRUE
NO	2	FALSE	N/A
PROMPT	1	N/A	FALSE
NO	1	N/A	TRUE

Arguments: `i_bln_BeforeCloseEvent`

Optional Boolean. Default is FALSE. This argument is used internally for when the `CloseWorkbook` command is being called by the `Application.Workbook_BeforeClose` event handler. This keeps it from actually being closed by this function.

### CreateNewDaCSWorkbook

Type: Function

Format: Boolean

Syntax: `Boolean = CreateNewDaCSWorkbook ( _  
[io_str_newWorkbookName As String], _  
[i_str_templateFile As String], _`

```
[i_int_sheetsInNewWorkbook As Integer], _
[i_bln_showDialog As Boolean = True] _
)
```

Description: Creates a new DaCS workbook.

Arguments: `io_str_newWorkbookName`  
Optional String. Full path + file name of the workbook. Subject to restriction checks. Default is `userID + timestamp.xls`. Actual filename used is returned in this argument.

`i_str_templateFile`  
Optional String. Full path + template name. Subject to restriction checks. Default is "Book.xlt" or as set by parameter "FILES.TEMPLATES.BLANK".

`i_int_sheetsInNewWorkbook`  
Optional String. Number of blank sheets to add to workbook, per the following table. If the dialog will be shown, then the number of sheets is placed on the dialog for confirmation. Note that the dialog will only hold 16 sheets, so if this argument is > 16 and the dialog is shown, it will only go up to 16 sheets.

Value	Sheets Added
< -1	1
-1	No sheets added. Dialog not shown.
0	3 or FILES.SHEETS_IN_NEW_WORKBOOK
> 0	Value

`i_bln_showDialog`  
Optional Boolean. TRUE to show Save File, **Add Sheets dialogs. FALSE to suppress. Note that setting** `i_int_sheetsInNewWorkbook` to -1 or `CSystem.Interactive` to FALSE will override this and set it to FALSE. Default is TRUE.

### CreateNewWorkbookFromTemplate

Type: Function

Format: Boolean

Syntax: `Boolean = CreateNewWorkbookFromTemplate ( _`  
`[i_str_templateFullName As String], _`  
`[i_bln_showDialog As Boolean = True] _`  
`)`

Description: Creates a new DaCS workbook from a template. This is used when you have an existing template and don't need to add any sheets to it.

Arguments: `i_str_templateFullName`

## Compassoft DaCS Programmers Guide

Optional String. Full path + template name. Subject to restriction checks. If not specified, the user is prompted for the template. Default filename is userID + timestamp.

`i_bln_showDialog`

Optional Boolean. TRUE to show Open Template, Save File dialog. FALSE to suppress. Note that setting `CSystem.Interactive` to FALSE will override this and set it to FALSE. Default is TRUE.

### DaCSEnabled

Type: Property

Format: Boolean

I/O: Read/Write

Syntax: `Boolean = DaCSEnabled`  
`DaCSEnabled = Boolean`

Description: Returns the state of DaCS, telling you if it is enabled or disabled. If you set the property to its inverse (i.e., set to FALSE when it is currently TRUE), it will run some environment setup (enabling/disabling commands per the interface workbook,) and toggle event traps. This corresponds to the menu item "DaCS | Disable DaCS" in the example Interface Workbook provided with the product. If DaCS is disabled, controlled workbooks won't be processed, unlocked, etc. when opened. Open DaCS-controlled workbooks must be saved before DaCS is changed from enabled to disabled. Once DaCS is disabled, all auditing controls are turned off; however, controlled workbooks cannot then be saved. This was added to allow people to "play" with datasets and prevent them from saving.

### InsertSheet

Type: Function

Format: Boolean

Syntax: `Boolean = InsertSheet ( _`  
`[io_wb As Object], _`  
`[io_str_sheetName As String], _`  
`[i_vnt_beforeSheet = "*"], _`  
`[i_bln_showDialog As Boolean = True], _`  
`[i_lng_sheetType As wsSheetType_Enum =`  
`wsSheetType_Worksheet] _`  
`)`

Description: Inserts a blank chart or worksheet.

Arguments: `io_wb`

Optional Object. Target workbook to add sheet to. If `Nothing` then `ActiveWorkbook` is used. Workbook must not be protected. (Check using `CFile.WorkbookProtected`, the `_SYSTEM`

parameter WORKBOOK.PROTECTION.PROTECT\_STRUCTURE must be FALSE.)

`io_str_sheetName`

Optional String. Name of sheet to add. If the sheet exists or is invalid, then Sheet<sub>n</sub> is used. The actual sheet name will be returned.

`i_vnt_beforeSheet`

Optional Variant. What sheet to put it before. If "\*" then is made the last sheet. Will accept a sheet object or sheet name.

`i_bln_showDialog`

Optional Boolean. TRUE to show Insert Sheet dialog. FALSE to suppress. Note that setting `CSystem.Interactive` to FALSE will override this and set it to FALSE. Default is TRUE.

`i_lng_sheetType`

Optional Long (`wsSheetType_Enum`). Type of sheet to add. Default is 1.

<b>wsSheetType_Enum</b>	<b>Value</b>
<code>wsSheetType_Worksheet</code>	1
<code>wsSheetType_Chart</code>	2

## LastError

Type: Property

Format: Object (`CErrLogger`)

I/O: Read/Write

Syntax: `oLastError = LastError`

Description: Handle to the `LastError` object for the class.

## MoveCopySheets

Type: Subroutine

Format: N/A

Syntax: `MoveCopySheets ()`

Description: Shows the Move or Copy Sheet dialog.

## OpenWorkbook

Type: Function

Format: N/A

Syntax: `Boolean = OpenWorkbook ( _`

## Compasssoft DaCS Programmers Guide

```
[i_str_filename As String], _  
[i_bln_showDialog As Boolean = True] _  
)
```

Description: Opens the specified workbook.

Arguments: i\_str\_filename

Optional String. Full path + name of the file to open.

i\_bln\_showDialog

Optional Boolean = TRUE to show Open File dialog. FALSE to suppress. Note that setting `CSystem.Interactive` to FALSE will override this and set it to FALSE. Default is TRUE.

## ProtectWorkbookStructure

Type: Function

Format: Boolean

```
Syntax: Boolean = ProtectWorkbookStructure ( _  
[i_wb As Object] _  
)
```

Description: Protects the workbook structure FOR REAL.

(`ActiveWorkbook.ProtectStructure = TRUE`). This is the normal state a controlled workbook is always in. This does not take any password argument, because it uses the internal encrypted password (`_SYSTEM` sheet parameter "PASSWORD".) Be sure you call this after you are done modifying the workbook structure, before saving. The intended use of this is to allow the programmer to insert/delete TEMPORARY sheets/charts. It is expected that you delete them before saving the file. If you want to insert a permanent sheet, use `CFile.InsertSheet`.

Arguments: i\_wb Optional Object. Workbook to protect structure of.

## RenameSheet

Type: Function

Format: Boolean

```
Syntax: Boolean = RenameSheet ( _  
[io_sheet As Object], _  
[io_str_newName As String], _  
[i_bln_showDialog As Boolean = True] _  
)
```

Description: Renames the specified sheet.

Arguments: io\_sheet

Optional Object. Sheet to rename. If not specified, the `ActiveSheet` is used.

`io_str_newName`

Optional String. The new sheet name. If successful, the final sheet name is returned. If no name is supplied then the dialog is shown. You can't use `_` or `~` to start the sheet name, and it will be trimmed to 31 characters.

`i_bln_showDialog`

Optional Boolean. TRUE to show dialog. FALSE to suppress. Note that setting `CSystem.Interactive` to FALSE will override this and set it to FALSE. Default is TRUE.

## SaveWorkbook

Type: Function  
 Format: Boolean  
 Syntax: `Boolean = SaveWorkbook ()`  
 Description: Saves the `ActiveWorkbook`.

## SaveWorkbookAs

Type: Function  
 Format: Boolean  
 Syntax: `Boolean = SaveWorkbookAs ( _  
     [io_str_SaveAsFileName As String], _  
     [i_bln_showDialog As Boolean = True] _  
 )`  
 Description: Saves the `ActiveWorkbook` under a new name.  
 Arguments: `io_str_SaveAsFileName`  
 Optional String. If not specified, the dialog is shown. The final file name is returned.  
`i_bln_showDialog`  
 Optional Boolean. TRUE to show dialog. FALSE to suppress. Note that setting `CSystem.Interactive` to FALSE will override this and set it to FALSE. Default is TRUE.

## SetSheetProtection

Type: Function  
 Format: Boolean  
 Syntax: `Boolean = SetSheetProtection( _  
     [io_sheet As Object], _  
     [i_str_password As String], _  
     [i_bln_protect As Boolean = True], _  
     [i_bln_showDialog As Boolean = True] _`  
 Description: Sets protection on the specified sheet.  
 Arguments: `io_sheet`

Optional Object. Sheet to protect.

`i_str_password`

Optional String. Password to use. The password will be encrypted and stored on the `_SYSTEM` sheet as parameter `SHEETS(name).PASSWORD`.

`i_bln_protect`

Optional Boolean. Protect or unprotect the sheet. Default is `TRUE`.

`i_bln_showDialog`

Optional Boolean. `TRUE` to show dialog. `FALSE` to suppress. Note that setting `CSystem.Interactive` to `FALSE` will override this and set it to `FALSE`. Default is `TRUE`.

### SetWorkbookProtection

Type: Function

Format: Boolean

Syntax: `Boolean = SetWorkbookProtection ( _  
    [io_wb As Object], _  
    [i_bln_showDialog As Boolean = True], _  
    [i_bln_protect As Boolean = True], _  
    [i_str_password As String], _  
    [i_bln_autoUpdateProtectedRanges As Boolean = True],  
    _  
    [i_bln_autoUpdateNameReferences As Boolean = True],  
    _  
    [i_bln_passwordProtectSavedStructure As Boolean =  
True] _  
)`

Description: Sets workbook protection, from the user perspective. Returns `TRUE` if successful, `FALSE` otherwise. Note that `ActiveWorkbook.ProtectStructure` is always `TRUE` with a DaCS-controlled workbook. Procedures check to see if the `_SYSTEM` parameter `WORKBOOK.PROTECTED` is `TRUE` or `FALSE`, to allow the procedure to run. The password is encrypted and stored in `_SYSTEM` parameter `WORKBOOK.PASSWORD`.

Arguments: `io_wb`

Optional Object. Default is `ActiveWorkbook`.

`i_bln_showDialog`

Optional Boolean. `TRUE` to show dialog. `FALSE` to suppress. Note that setting `CSystem.Interactive` to `FALSE` will override this and set it to `FALSE`. Default is `TRUE`.

`i_bln_protect`

Optional Boolean. Default is `TRUE`, to protect. `FALSE` will unprotect. Sets `_SYSTEM` parameter `WORKBOOK.PROTECTION.PROTECT_STRUCTURE`.

`i_str_password`



that it will be synchronized with any changes made while events were off. Used primarily for data import macros.

Arguments: `i_wb` Optional Object. Default is `ActiveWorkbook`.

## UnhideWindow

Type: Subroutine

Format: N/A

Syntax: `UnhideWindow ()`

Description: Shows the Unhide Window dialog.

## UnprotectWorkbookStructure

Type: Function

Format: Boolean

Syntax: `Boolean = UnprotectWorkbookStructure ( _  
[i_wb As Object] _  
)`

Description: Unprotects the workbook structure FOR REAL. (`ActiveWorkbook.ProtectStructure = FALSE`). This does not take any password argument, because it uses the internal encrypted password (`_SYSTEM` sheet parameter "PASSWORD"). Be sure you call `CFile.ProtectWorkbookStructure` after you are done modifying the workbook structure, before saving. The intended use of this is to allow the programmer to insert/delete TEMPORARY sheets/charts. It is expected that you delete them before saving the file. If you want to insert a permanent sheet, use `CFile.InsertSheet`.

Arguments: `i_wb`  
Optional Object. Workbook to unprotect structure of.

## WorkbookStructureProtected

Type: Function

Format: Boolean

Syntax: `Boolean = WorkbookStructureProtected ( _  
[io_wb As Object] _  
)`

Description: Returns if the workbook structure is protected (TRUE) or unprotected (FALSE) from the user's perspective. If a regular workbook, returns `io_wb.ProtectStructure`. If there is an error, returns TRUE assuming that it is protected. If a controlled workbook, then it returns the `_SYSTEM` parameter `WORKBOOK.PROTECTION.PROTECT_STRUCTURE`. You should call this function instead of trying to use `ActiveWorkbook.ProtectStructure`, as controlled workbooks will always return TRUE.

Arguments: `io_wb` Optional Object. Workbook to check protect structure of.

# Chapter 5

## CInstrument

---

### In This Chapter

Using CInstrument.....	44
CInstrument Property Defaults .....	45
CInstrument Enumerations .....	46
CInstrument Members.....	46
CInstrument Example Code .....	53

### Using CInstrument

---

CInstrument is used to allow an instrument to log in and control DaCS, including the ability to log in users transparently. This allows the instrument to pass down a user logged into the instrument, so that the audit trail will record the user's identity. The user is not required to be present in the DaCS user database.

A reference to CInstrument is returned from CSystem. Since CSystem is scoped within an Interface Workbook, you return the reference through the Interface Workbook. To do this, you need to expose CSystem through a public property of the ThisWorkbook class of the Interface Workbook:

```
'// Note that g_oDaCS is declared as a public Object  
  
Public Property Get DaCSInterface() As Object  
  
    If g_oDaCS Is Nothing Then  
  
        Set g_oDaCS = CreateObject("WSDACS.CSystem")  
  
    End If  
  
    Set DaCSInterface = g_oDaCS  
  
End Property
```

Now in your program you can get the reference:

```
'// Declarations (note it is preferable to declare all these  
'// as Object instead of using early binding.)  
  
Dim oWorkbook as Workbook  
  
Dim oDaCS as CSystem  
  
Dim oInstrument as CInstrument  
  
'// Get a reference to the workbook  
  
Set oWorkbook = Excel.Application.Workbooks([interface workbook])  
  
'// Get a reference to CSystem  
  
Set oDaCS = oWorkbook.DaCSInterface  
  
'// Set CSystem.Interactive to False to make sure error  
'// messages are suppressed.  
  
oDaCS.Interactive = False  
  
'// Get a reference to CInstrument
```

```
Boolean = oDaCS.GetInstrumentObject (o_oInstrument:=oInstrument)
```

If the `GetInstrumentObject` succeeds, it will return `TRUE` and `o_oInstrument` will contain a reference to `CInstrument`. (See `CSystem` Programming Documentation for details.) You should get the reference to `CInstrument` prior to starting DaCS. Once you have a reference to `CInstrument`, you can load an encrypted Profile and log the instrument in. From there, you can alias in users.

Example code showing how `CInstrument` can be used is given at the end of this document.

The `CInstrument` object and its members can be viewed using the VB or VBA Project Explorer.

## **CInstrument Property Defaults**

---

The following properties are set on creation of an instance of a `CSystem` object. What the properties are depends on if `CSystem.Instance` has been set to an Interface Workbook or not:

<b>Property</b>	<b>Default</b>
CompanyName	Empty
InstrumentID	Empty
InstrumentName	Empty
LastError	New CErrLogger
License	Empty
LoadExtendedUserProperties	False
SoftwareName	Empty
SystemCertificate	Empty
SystemCertificateKey	Empty
SystemID	Empty
SystemPublicKey	Empty

## CInstrument Enumerations

---

The following Enum is defined, but not currently implemented:

wsCInstrument_ControlMode_Enum	Value
wsCInstrument_ControlMode_User	0
wsCInstrument_ControlMode_Instrument	1

## CInstrument Members

---

### AliasUser

Type:            Function  
Format:          Boolean  
Syntax:          Boolean = AliasUser ( \_  
  i\_sUsername As String, \_  
  [i\_sUserFullName As String], \_  
  [i\_sLocation As String],  
  [i\_lAuthority as Long],  
  [i\_sCertificate as String],  
  [i\_sPublicKey as String],  
  [i\_sPrivateKey as String] \_  
  )

Description:    Aliases (logs in) a User from the specified database. This allows an external software package to pass a login into DaCS, without the user having to log into DaCS. If `LoadExtendedUserProperties` is `FALSE`, then the user is logged in as a user-level account (never administrator) and without electronic signature capability. If `LoadExtendedUserProperties` is `TRUE`, then the user is logged in at the authority level specified and electronic signatures may be executed if the Private Key is loaded. Returns `TRUE` if the user was aliased. Returns `FALSE` if there was an error. You should try `AliasUserFromDatabase` first, then resort to this if the user can't be located.

Arguments:      i\_sUsername  
                  Required String. The UserID in DaCS used for login. Not to be confused with the user's full name. This is the UserID field in the workbook audit trail.

`i_sUserFullName`

Optional String. The full name of the user. This is the User Name field in the workbook audit trail.

`i_sLocation`

Optional String. The location, department, etc. of the user. This is an extra field that is recorded as part of the user's metadata on the `_SYSTEM` sheet of the workbook.

`i_lAuthority`

Optional Long. The user authority level. 1 = Administrator, 2 = User. Default is 2. Note that if `LoadExtendedUserProperties` is `FALSE` then the value of this argument is ignored and the user authority level will be set to the default value of 2.

`i_sCertificate`

Optional String. The electronic signature Certificate for the user.

`i_sCertificateKey`

Optional String. The electronic signature Certificate Key for the user.

`i_sPublicKey`

Optional String. The electronic signature Public Key for the user.

`i_sPrivateKey`

Optional String. The electronic signature Private Key for the user. Note that if `LoadExtendedUserProperties` is `FALSE` then the argument is ignored and the Private Key will be empty.

Errors:

If no Instrument is logged in:

```
DACS_ERROR.INSTRUMENT_ERROR_INSTRUMENT_NOT_LOGGED_IN
```

If `i_sUsername = Empty`:

```
DACS_ERROR.INSTRUMENT_ERROR_USERNAME_NOT_SPECIFIED
```

## AliasUserFromDatabase

Type:	Function
Format:	Boolean
Syntax:	Boolean = AliasUserFromDatabase ( _

## Compasssoft DaCS Programmers Guide

```
        i_sUsername As String, _  
        [i_sConnectionString As String], _  
        [i_sDatabaseDateFormat As String], _  
        [i_sDBSignatureDateFormat As String] _  
    )
```

**Description:** Aliases (logs in) a User from the specified database. This allows an external software package to pass a login into DaCS, without the user having to log into DaCS. If `LoadExtendedUserProperties` is `FALSE`, then the user is logged in as a user-level account (never administrator) and without electronic signature capability. If `LoadExtendedUserProperties` is `TRUE`, then the user is logged in at the authority level defined for their account in the database and electronic signatures may be executed if the user has a certified keypair for their account in the database. Returns `TRUE` if the user was aliased. Returns `FALSE` if there was an error. If this function returns `FALSE`, you can use `AliasUser` instead.

**Arguments:**

- `i_sUsername`  
Required String. The UserID in DaCS used for login. Not to be confused with the user's full name.
- `i_sConnectionString`  
Optional String. A valid connection string to the database. If not specified, the connection string from the Interface Workbook will be used if `CSystem.Instance` has been set.  
(DATABASE.CONNECTION)
- `i_sDatabaseDateFormat`  
Optional String. The format dates are displayed in the database. If not specified, the date format from the Interface Workbook will be used if `CSystem.Instance` has been set.  
(DATABASE.FIELDS.DATA\_FORMAT)
- `i_sDBSignatureDateFormat`  
Optional String. The format dates are interpreted for digital signature checks (account integrity) in the database. If not specified, the date format from the Interface Workbook will be used if `CSystem.Instance` has been set.  
(DATABASE.SIGNATURE.DATE\_FORMAT)

**Errors:** Note the `LastError.Line` property of the error. If it is in the 200's, then it applies to the SYSTEM account. If it is in the 300's, then it applies to the User account.

Error	Condition
INSTRUMENT_ERROR_INSTRUMENT_NOT_LOGGED_IN	Instrument isn't logged in
INSTRUMENT_ERROR_USERNAME_NOT_SPECIFIED	i_sUsername = Empty
DATABASE_ERROR_CONNECTION_NOT_SPECIFIED	i_sConnectionString = Empty
DATABASE_ERROR_RECORD_NOT_FOUND	Username not in database
DATABASE_ERROR_RECORD_CORRUPT	account is corrupt; i_sDatabaseDateFormat or i_sDBSignatureDateFormat are not the correct format
USER_ERROR_CERTIFICATE_INVALID	SYSTEM certificate is invalid
USER_ERROR_ACCOUNT_LOCKED	User account is locked

### CompanyName

Type: Property  
 Format: String  
 I/O: Read-Only  
 Description: Full name of the company of the external software package. See `CProfile.CompanyName`.

### GetProfileObject

Type: Function  
 Format: Boolean  
 Syntax: `Boolean = GetProfileObject (o_oProfile As CProfile)`  
 Description: Returns a reference to the `CProfile` object of the class. Returns `TRUE` if successful, `FALSE` if there is an error.

### InstrumentID

Type: Property  
 Format: String  
 I/O: Read/Write

## Compassoft DaCS Programmers Guide

Description: UserID of the instrument. See `CProfile.InstrumentID`.

### InstrumentName

Type: Property

Format: Boolean

I/O: Read/Write

Description: Full name of the instrument. See `CProfile.InstrumentName`.

### LastError

Type: Object (`CErrLogger`)

Format: Boolean

Description: Reference to the `LastError` object for the class. The last error encountered will be logged and retained in this object.

### License

Type: Property

Format: Boolean

I/O: Read/Write

Description: Unique identifier for the software package (instrument). See `CProfile.License`.

### LoadExtendedUserProperties

Type: Property

Format: Boolean

I/O: Read-Only

Description: If set to `TRUE`, then aliasing a user includes setting the user's authority level (allowing administrator access) and private key (allowing electronic signatures). If set to `FALSE`, then electronic signatures will be disabled by virtue of disallowing the user's private key to be set, and the user's authority level will be set to `User`. This property can only be set by loading a Profile with this property value defined in it. See `CProfile..`

### LoginProfile

Type: Function

Format: Boolean

Syntax: Boolean = `LoginProfile ()`

Description: "Logs in" the Instrument as a user within DaCS. The Instrument has the authority level of a standard user. Any Aliased user is cleared. Returns `TRUE` if successful, `FALSE` if there was an error.

Errors: If a valid profile has not been loaded using `CInstrument.ParseInstrumentFromProfile`, then the following error will be logged:

## INSTRUMENT\_ERROR\_PROFILE\_NOT\_LOADED

**ParseInstrumentFromProfile**

Type: Function

Format: Boolean

Syntax: Boolean = ParseInstrumentFromProfile ()

Description: Unloads properties from the `CProfile` object into the Instrument. A `CProfile` object must be loaded and decrypted first. Use `CInstrument.GetProfileObject` to do this. The profile will be parsed for the properties listed below; see `CProfile` documentation for details.

Property	Counter
SystemID	1
CompanyName	2
License	4
SystemPublicKey	8
SystemCertificate	16
SystemCertificateKey	32
InstrumentID	64
SoftwareName	128
InstrumentName	256
LoadExtendedUserProperties	512

Errors: If the profile is incomplete (not all required properties were found), the function will return `FALSE` and an error will be logged. (However, the property `LoadExtendedUserProperties` is optional and will not return an error if missing.) The `Description` will be a number corresponding to the missing fields; see the Counter value in the table above:

DACS\_ERROR.INSTRUMENT\_ERROR\_PROFILE\_INCOMPLETE

Example: If missing `SoftwareName` and `InstrumentName`, `LastError.Description` will equal 384 (128+256). If the profile hasn't been loaded or decrypted, then the following error will be returned:

INSTRUMENT\_ERROR\_CANT\_PARSE\_ENCRYPTED\_PROFILE

### SoftwareName

Type: Property  
Format: Boolean  
I/O: Read/Write  
Description: Full name of the software package. See `CProfile.SoftwareName`.

### SystemCertificate

Type: Property  
Format: Boolean  
I/O: Read/Write  
Description: Certificate (digital signature) for the SYSTEM account. See `CProfile.SystemCertificate`.

### SystemCertificateKey

Type: Function  
Format: Boolean  
Description: Certificate public key for the SYSTEM account. See `CProfile.SystemCertificateKey`.

### SystemID

Type: Property  
Format: Boolean  
I/O: Read/Write  
Description: UserID of the DaCS SYSTEM account. See `CProfile.SystemID`.

### SystemPublicKey

Type: Property  
Format: Boolean  
I/O: Read/Write  
Description: Public key of the SYSTEM account. See `CProfile.SystemPublicKey`.

## CInstrument Example Code

---

Obviously, this code is just an example as it does not have error handling, etc. This sub assumes you have a reference to `g_oDaCS` somewhere else in your code (see Using CInstrument above for how to do this.)

```

'// Declarations (preferable to use Object instead)

Public g_oDaCS as CSystem

Public g_oInstrument as CSystem

Public g_oProfile as CProfile

Sub AliasExample()

    '// Set up strings, replace with your own code

    sProfilePath = "C:\Program Files\Compassoft\DaCS\Install\Profiles\"

    sEncryptedProfileFileName = "MyProfile.txt"

    sSessionKeyFileName = "MyKey.txt"

sConnectionString = "Driver=Microsoft Access Driver (*.mdb);" & _
    "DBQ=[C:\Program Files\Compassoft\" & _
    "DaCS\Install\dacsuser.mdb];"

    sDatabaseDateFormat = "EX"

    sDatabaseSigDateFormat = "EX"

    sUsername = "DWIMMER"

    sUserFullName = "Finance"

    sLocation = "Bioanalytical"

    '// Get the object we need to use

    g_oDaCS.GetInstrumentObject o_oInstrument:=g_oInstrument

    g_oInstrument.GetProfileObject o_oProfile:=g_oProfile

    '// Load and decrypt the Profile

    If g_oProfile.DecryptEncryptedProfile ( _
        i_sEncryptedProfile:= _
            sProfilePath & sEncryptedProfileFileName, _
        i_nEncryptedProfileDataFormat:=0, _

```

## Compassoft DaCS Programmers Guide

```
        i_sSessionKey:=sProfilePath & sSessionKeyFileName, _
        i_nSessionKeyDataFormat:=0) _
Then
    '//  The profile was decrypted, so we can parse it
    '//  into CInstrument and login using that profile
With g_oInstrument
    .ParseInstrumentFromProfile
    .LoginProfile
    '//  Now try to alias the user from the database
If Not .AliasUserFromDatabase ( _
    i_sUsername:=sUsername, _
    i_sConnectionString:=sConnectionString, _
    i_sDatabaseDateFormat:= sDatabaseDateFormat, _
    i_sDBSignatureDateFormat:= sDBSignatureDateFormat) _
Then
    '//  That didn't work, so force the alias.
    . AliasUser _
        i_sUsername:=sUserName, _
        i_sUserFullName:=sUserFullName, _
        i_sLocation:=sLocation
    End If
End With
End If
End Sub
```

## CProfile

---

### In This Chapter

Profile Structure .....	56
CProfile Enumerations .....	58
Security and Cryptography in CProfile .....	59
CProfile Members .....	59

### Profile Structure

---

A profile object essentially contains a static image of a user database, containing both a SYSTEM account and a "user" account for the instrument. The profile can be readily generated by setting up a DaCS user database and then extracting certain fields. The fields are tagged, yielding a plain-text profile consisting of a property tag followed by a value. The profile is then encrypted; the process results in an `EncryptedProfile` and a `SessionKey`. DaCS decrypts the `EncryptedProfile` using the `SessionKey` and then parses the tagged data. The decrypted `Profile` string looks similar to the following; however there are no line breaks:

```
SystemID:=SYSTEM;

CompanyName:=Compassoft;

License:=COMPASSOFT;

SystemPublicKey:=[hex data];

SystemCertificate:=[hex data];

SystemCertificateKey:=[hex data];

InstrumentID:=EXTERNAL;

SoftwareName:=External Control Software System;

InstrumentName:=DaCS Update Agent;

LoadExtendedUserProperties:=True;
```

The property tags in the `Profile` match the `CProfile` properties. These properties map to a SYSTEM and "user" (here, Instrument) account in a DaCS database. When a workbook is opened, the metadata parameters on the `_SYSTEM` worksheet listed are updated with the corresponding property value:

Property	Account	Database Field	Metadata Parameter
CompanyName	SYSTEM	ws_full_name	USERS(SYSTEM).FULLNAME
License	SYSTEM	ws_location	USERS(SYSTEM).LICENSE
SystemCertificate	SYSTEM	ws_certificate	USERS(SYSTEM).CERTIFICATE
SystemCertificateKey	SYSTEM	ws_certificate_key	USERS(SYSTEM).CERTIFICATE_KEY
SystemID	SYSTEM	ws_username	N/A
SystemPublicKey	SYSTEM	ws_public_key	USERS(SYSTEM).PUBLIC_KEY

InstrumentID	Instrument	ws_username	N/A
InstrumentName	Instrument	ws_location	USERS(Instrument).LOCATION
SoftwareName	Instrument	ws_full_name	USERS(Instrument).FULLNAME

Additionally, the following metadata parameters on the \_SYSTEM worksheet are set:

`CURRENT_INSTRUMENT_ID = CProfile.InstrumentID`

`CURRENT_SYSTEM_ID = CProfile.SystemID`

`CURRENT_ALIASED_SYSTEM_ID = SYSTEM account from database`

`CURRENT_USER_ID = Aliased User ID`

Note the above listing includes an aliased user.

## CProfile Enumerations

DATA_FORMAT	Value
FILE_BINARY	0
FILE_HEX	1
STRING_HEX	2
STRING_VB	3
CURRENT_KEY_CONTAINER	4

Note: CURRENT\_KEY\_CONTAINER should not be used.

### Using DATA\_FORMAT

Each String Property has a Format Property that describes the data represented by that string:

String Property	Format Property
EncryptedProfile	EncryptedProfileDataFormat
Profile	ProfileDataFormat
SessionKey	SessionKeyDataFormat

As can be seen from the Enumerations table for DATA\_FORMAT, the String Property may represent either a string or a file, in either binary or hexadecimal format. When the Format Property is FILE\_BINARY or FILE\_HEX, then the String Property is a pointer to a file that either contains binary data or hexadecimal data. When the Format Property is STRING\_HEX or STRING\_VB, then the String Property is either a hexadecimal string or a "Visual Basic" string (null-terminated). Binary format represents each byte as an 8-bit unsigned integer value from 0-255 inclusive, so a text representation will contain linefeeds and non-printable characters. Hexadecimal format represents each byte as an alphanumeric from 0-9 and A-F inclusive. Example formats of the data is illustrated below:

DATA_FORMAT	Example Data
FILE_BINARY	Nè:¶þóR□ÂæÇñ Ð
FILE_HEX	1EF47D032C2A
STRING_HEX	1EF47D032C2A
STRING_VB	Nè:¶þóR□ÂæÇñ Ð

It is recommended that `EncryptedProfile` be a file, and that `SessionKey` be a string resource provided by your software package. This allows you to distribute updated instrument profiles and not have to worry about managing the session key or unauthorized use. (See below.)

## Security and Cryptography in CProfile

---

`CProfile` makes use of cryptography to provide a secure method for an external software package to log into DaCS and also to log users into DaCS transparently.

A cryptographic structure called a session key is generated from a secret keyphrase. The session key is used to encrypt the plain-text profile, yielding an `EncryptedProfile` file. The session key is encrypted using a public key known to DaCS to yield a `SessionKey` file. (The `SessionKey` should then be securely embedded into the external program as a string resource to prevent use by others.)

When DaCS is instructed to load the profile, it needs to decrypt it. DaCS gains access to the `SessionKey`, but it is encrypted. DaCS has a private key (corresponding to the public key) that it uses to decrypt the `SessionKey`. The decrypted session key is used to decrypt the `EncryptedProfile`.

This scheme allows for a measure of security surrounding the use of a profile for automated and aliased logins. Because the profile is encrypted, it is secure against a lay user creating their own profile or spoofing an existing one. The lay user cannot load any profile through their own program without the `SessionKey`; hence the need to embed it rather than leave it as a file. And the lay user cannot create their own `SessionKey` nor encrypt their own profiles, because they do not have access to the generating keyphrase or DaCS private key used to encrypt a `SessionKey`. Only a program with the `SessionKey` will be able to load profiles encrypted with that particular `SessionKey`.

## CProfile Members

---

### DecryptEncryptedProfile

Type: Function

Format: Boolean

Syntax: `Boolean = DecryptEncryptedProfile ( _  
           [i_sEncryptedProfile As String], _  
           [i_nEncryptedProfileDataFormat As _  
           DATA_FORMAT = FILE_BINARY], _  
           [i_sSessionKey As String], _  
           [i_nSessionKeyDataFormat As _  
           DATA_FORMAT = FILE_BINARY] _  
           )`

Description: Decrypts an encrypted profile. Returns the decrypted results into the `Profile` property as a string (`STRING_VB`). If any arguments are omitted, then any previous property settings will

be applied. If arguments are included, the corresponding property of the class will be set.

### DecryptProfile

Type: Function

Format: Boolean

Syntax: Boolean = DecryptProfile ()

Description: Decrypts an encrypted profile. Returns the decrypted results into the `Profile` property based on the setting of `ProfileDataFormat` (i.e., either a string or a file.) This function takes no arguments, as it uses whatever the class properties are for `EncryptedProfile`, `EncryptedProfileDataFormat`, `Profile`, `ProfileDataFormat`, `SessionKey`, and `SessionKeyDataFormat`.

### EncryptedProfile

Type: Property

Format: String

I/O: Read/Write

Description: Reference to the encrypted profile. This may be a path to a file or a string, depending on `EncryptedProfileDataFormat`.

### EncryptedProfileDataFormat

Type: Property

Format: DATA\_FORMAT

I/O: Read/Write

Description: Data format for the `EncryptedProfile`.

### LastError

Type: Object (`CErrLogger`)

Format: Boolean

Description: Reference to the `LastError` object for the class. The last error encountered will be logged and retained in this object.

### Profile

Type: Property

Format: String

I/O: Read/Write

Description: Reference to the decrypted profile. This may be a path to a file or a string, depending on `ProfileDataFormat`.

### ProfileDataFormat

Type: Property

**Format:** DATA\_FORMAT  
**I/O:** Read/Write  
**Description:** Data format for the Profile. Ignored if you use `DecryptEncryptedProfile`.  
**Note:** This property will only be used if you use `DecryptProfile`. Note that if you set this to `FILE_BINARY` or `FILE_HEX`, then when you call `DecryptProfile`, the Profile will be written to the filename specified in `Profile`. Note that this will overwrite any existing file data; if the written output is shorter than the original file, the file will retain the remainder of the original file as garbage. Therefore, you should ensure the file doesn't already exist or is deleted prior to calling `DecryptProfile`. If you use `DecryptEncryptedProfile`, this property is ignored and the `Profile` is returned as `STRING_VB`.

### ProfileDecrypted

**Type:** Property  
**Format:** Boolean  
**I/O:** Read-Only  
**Description:** Returns if the loaded profile has been decrypted (`TRUE`) or not (`FALSE`).

### SessionKey

**Type:** Property  
**Format:** String  
**I/O:** Read/Write  
**Description:** Reference to the encrypted session key. This may be a path to a file or a string, depending on `SessionKeyDataFormat`.

### SessionKeyDataFormat

**Type:** Property  
**Format:** DATA\_FORMAT  
**I/O:** Read/Write  
**Description:** Data format for `SessionKey`.



## CSystem

---

### In This Chapter

Using CSystem.....	63
CSystem Property Defaults .....	64
CSystem Members.....	64

### Using CSystem

---

CSystem is the top-level object in DaCS and is used to start up the system. See the Interface Workbook procedure StartDaCS for an example.

#### Late Binding (preferred):

(declarations)

```
Public goDaCS As Object
```

(startup procedure)

```
Set goDaCS = CreateObject("WSDACS.CSystem")
```

#### Early Binding (requires Reference to WSDACS.EXE)

(declarations)

```
Public goDaCS As CSystem
```

(startup procedure)

```
Set goDaCS = New CSystem
```

The CSystem object and its members can be viewed using the VB or VBA Project Explorer.

### CSystem Property Defaults

---

The following properties are set on creation of an instance of a `CSystem` object. What the properties are depends on if `CSystem.Instance` has been set to an Interface Workbook or not:

Property	Not Set	Set
<code>ActiveWorkbookStatus</code>	0	0
<code>AuditSilent</code>	FALSE	FALSE or <code>AUDIT.SILENT</code>
<code>AuditSilentReason</code>	Empty	"Silent Audit" or <code>AUDIT.SILENT.REASON</code>
<code>Build</code> (see note below)	20060727	20060727
<code>DaCSStatus</code>	4	2 prior to login, then 3
<code>ExcelVersion</code>	empty	Excel 97 (or as detected)
<code>Instance</code>	N/A	N/A
<code>InstanceOK</code>	FALSE	TRUE
<code>Interactive</code>	TRUE	TRUE
<code>LastError</code>	Empty	Empty
<code>Menu</code>	Nothing	<code>CFile</code>
<code>SignatureVersion</code>	1.3	1.3
<code>Version</code>	DaCS version 3.0.1	DaCS version 3.0.1
<code>WorkbookStatus</code>	N/A	N/A

**Note:** The build property above reflects the date of the latest build in the following format (YYYYMMDD).

### CSystem Members

---

#### ActiveWorkbookStatus

Type: Property  
Format: `wsCSystem_WorkbookStatus_Enum`  
I/O: Read-Only  
Description: Returns the status of the `ActiveWorkbook` in DaCS.

wsCSystem_WorkbookStatus_Enum	Value
wsCSystem_WorkbookError	0
wsCSystem_WorkbookRegularWorkbook	1
wsCSystem_WorkbookDaCSControlled	2
wsCSystem_WorkbookDaCSGhost	4
wsCSystem_WorkbookDaCSInterface	8

## AuditSilent

Type: Property  
Format: Boolean  
I/O: Read/Write  
Description: Allows external program to turn silent audit on/off. Set to TRUE to turn silent auditing on. Set to FALSE to turn silent auditing off. If auditing is silent, all changes are committed with the `AuditSilentReason` given as the reason for the change.

## AuditSilentReason

Type: Property  
Format: String  
I/O: Read/Write  
Description: What is recorded in the audit trail Reason field when silent auditing is on. If `AuditSilent = TRUE`, all changes are committed with the `AuditSilentReason` given as the reason for the change.

## Build

Type: Property  
Format: String  
I/O: Read-Only  
Description: Build identifier for DaCS.

## DaCSStatus

Type: Property  
Format: `wsCSystem_DaCSStatus_Enum`  
I/O: Read-Only  
Description: Returns the status of the application. Additive property.

## Compasssoft DaCS Programmers Guide

wsCSystem_DaCSStatus_Enum	Value
wsCSystem_DaCSShutdown	0
wsCSystem_DaCSRRunning	1
wsCSystem_DaCSEnabled	2
wsCSystem_DaCSDisabled	4

### ExcelVersion

Type: Property

Format: String

I/O: Read-Only

Description: Returns the version name of Excel that DaCS is attached to.

Excel Version	Value
Excel 97	Excel 97
Excel 2000	Excel 2000
Excel 2002 (XP)	Excel 2002 (XP)
Excel 2003	Excel 2003

### GetAuditObject

Type: Function

Format: Boolean

Syntax: Boolean = GetAuditObject (o\_oAudit As CAudit)

Description: Returns a new CAudit object. TRUE if successful. FALSE if there was an error.

Arguments: o\_oAudit Required Object (CAudit). Reference to a new instance of CAudit.

### GetInstrumentObject

Type: Function

Format: Boolean

Syntax: Boolean = GetInstrumentObject ( \_  
o\_oInstrument As CInstrument \_

)

Description: Returns a new `CInstrument` object. TRUE if successful. FALSE if there was an error.

Arguments `o_oInstrument` Required Object (`CInstrument`). Reference to a new instance of `CInstrument`.

## Instance

Type: Property

Format: Object (Workbook or Excel.Application)

I/O: Write-Only

Description: Connects DaCS to an instance of Excel for event hooks. Also performs the following tasks: connects to Interface Workbook SETTINGS worksheet, sets `ExcelVersion`, sets `InstanceOK`, hides Interface Workbook, loads parameters from SETTINGS, creates temporary folder, sets `Menu`, loads SYSTEM user from database (if an instrument isn't logged in.) Success/failure is set in `InstanceOK`.

## InstanceOK

Type: Property

Format: Boolean

I/O: Read-Only

Description: Returns TRUE if Instance was properly set (all tasks occurred without error.) Returns FALSE if Instance has not been set.

## Interactive

Type: Property

Format: Boolean

I/O: Read/Write

Description: Use to turn alerts, errors, prompts off for instrument control. Set to TRUE to show alerts, errors, and prompts. Set to FALSE to suppress.

## LastError

Type: Object (`CErrLogger`)

Format: Boolean

Description: Reference to the `LastError` object for the class. The last error encountered will be logged and retained in this object.

## Menu

Type: Property

Format: Object (`CFile`)

I/O: Read-Only

Description: Accessor property for `CFile`.

## Parent

Type: Property  
Format: Object (App)  
I/O: Read-Only  
Description: Accessor property for Visual Basic App object.

## SetInterfaceWorkbookSystemObject

Type: Function  
Format: Boolean  
Syntax: 

```
Boolean = SetInterfaceWorkbookSystemObject ( _  
        io_Workbook As Object _  
    )
```

  
Description: Sets the Interface Workbook bound to an instance of CSystem. Used by external programs that start DaCS instead of an Interface Workbook. TRUE if successful. FALSE if there was an error.  
Arguments: io\_Workbook  
Required Object (Workbook). Reference to an Interface Workbook.

## SetEnvironment

Type: Function  
Format: Long  
Syntax: 

```
Long = SetEnvironment ( _  
        [i_bln_returnToNormal As Boolean = False] _  
    )
```

  
Description: Triggers the process of setting up the user interface of Excel for DaCS, or returning it to "normal". This encompasses toolbar remapping, shortcut key remaps, custom command bars, etc. Always returns 0.

## Shutdown

Type: Subroutine  
Format: Boolean  
Syntax: Shutdown()  
Description: Closes all open workbooks and calls SetEnvironment(TRUE). Does not terminate CSystem.

## SignatureVersion

Type: Property  
Format: String  
I/O: Read-Only  
Description: Returns the digital/electronic signature engine version used by DaCS.

**Start**

Type: Function  
 Format: Boolean  
 Syntax: 

```
Boolean = Start( _
                [i_str_username As String], _
                [i_str_password As String] _
            )
```

Description: Wrapper for user login and DaCS startup. Logs in a user and calls `SetEnvironment (FALSE)`. If login is cancelled, handles DaCS shutdown. Returns TRUE if the user logged in. Returns FALSE if the user quit DaCS or cancelled login. Warning: With the default Interface Workbook, if the username/password is incorrect, it will lock the user out!

Arguments: `i_str_username` Optional String. UserID to use for login.  
`i_str_password` Optional String. Password to use for login.

**Version**

Type: Property  
 Format: String  
 I/O: Read-Only  
 Description: Returns the DaCS application version.

**WorkbookStatus**

Type: Property  
 Format: `wsCSystem_WorkbookStatusEnum`  
 I/O: Read-Only  
 Syntax: 

```
Long = WorkbookStatus (io_wb As Object)
```

Description: Returns the status of the specified workbook in DaCS.

Arguments: `io_wb` Required Object. Workbook to return the status of.

<code>wsCSystem_WorkbookStatus_Enum</code>	Value
<code>wsCSystem_WorkbookError</code>	0
<code>wsCSystem_WorkbookRegularWorkbook</code>	1
<code>wsCSystem_WorkbookDaCSControlled</code>	2
<code>wsCSystem_WorkbookDaCSGhost</code>	4
<code>wsCSystem_WorkbookDaCSInterface</code>	8



## CConstDaCS

---

### In This Chapter

Using CConstDaCS.....	71
CConstDaCS Members.....	71
CConstDaCS Enums .....	72

### Using CConstDaCS

---

This class is a globally available class containing information on the formatting of Instrument Profiles and the DACS\_ERROR enum.

### CConstDaCS Members

---

#### PROFILE\_FIELD\_DELIMITER

Type:	Property
Format:	String
I/O:	Read-Only
Description:	The delimiter used between data elements in an Instrument Profile. Returns a semicolon character, <code>CHR(59)</code> or <code>“;”</code> .

#### PROFILE\_FIELD\_TAG

Type:	Property
Format:	String
I/O:	Read-Only
Description:	The tag used between a field name and data element in an Instrument Profile. Returns a colon character followed by the “equals sign”, <code>CHR(58)</code> followed by <code>CHR(61)</code> or <code>“:=”</code> .

## CConstDaCS Enums

The following show up in the library <globals> as members of WSDACS.DACS\_ERROR. As these enums are used for error messages, the associated information for the errors is shown. Additional error information can be found in the DaCS Administrator's Guide.

Error Number	Hex	Enum/Descriptions	Module	Procedure	Failure Trigger	Message
-2147220413	&H80040443	DACS_UNDEFIN ED_ERROR			Reserved	
-2147220412	&H80040444	SYSTEM_ERROR _EXCEL_INSTAN CE_NOT_SET	CSystem	Start	Excel instance not set	DaCS cannot start because the link to Excel hasn't been set. Please contact your System Administrator.
-2147220411	&H80040445	SYSTEM_ERROR _TOO_MANY_CO NCURRENT_FOL DERS	CSystem	CreateTempFold er	More than 999 folders with same timestamp	Too many concurrent folders for [path/folder]
-2147220410	&H80040446	SYSTEM_ERROR _DACS_ALREAD Y_RUNNING	CSystem	CheckInstance	Workbook that the DACS_LOG control points to is an open Interface Workbook.	DaCS is already running in this instance of Excel. You can't have two open at once. Contact your System Administrator if you need help.
-2147220409	&H80040447	SYSTEM_ERROR _DACS_IMPROP ERLY_TERMINAT ED	CSystem	CheckInstance	Workbook that the DACS_LOG control points to is not open.	The Interface Workbook closed but did not terminate properly.
-2147220408	&H80040448	INSTRUMENT_E RROR_CANT_PA RSE_ENCRYPTE D_PROFILE	CInstrument	ParseInstrument FromProfile	Instrument profile has not been decrypted yet.	
-2147220407	&H80040449	INSTRUMENT_E RROR_PROFILE_ NOT_LOADED	CInstrument	LoginProfile	A valid profile has not been loaded yet.	
-2147220406	&H8004044A	INSTRUMENT_E RROR_INSTRUM ENT_NOT_LOGG ED_IN	CInstrument	AliasUserFromDa tabase	Instrument has not been logged in	
-2147220405	&H8004044B	INSTRUMENT_E RROR_USERNAM E_NOT_SPECIFI ED	CInstrument	AliasUserFromDa tabase	Username argument of method is empty or missing.	

## Chapter 8 CConstDaCS

-2147220404	&H8004044C	INSTRUMENT_ERROR_PROFILE_INCOMPLETE (The description reflects the missing elements, see CInstrument.ParseInstrumentFromProfile for details.)	CInstrument	ParseInstrumentFromProfile	Missing data elements in instrument profile.	
-2147220403	&H8004044D	DATABASE_ERROR_CONNECTION_NOT_SPECIFIED	CInstrument	AliasUserFromDatabase	Database connection for DaCS is not specified.	
-2147220402	&H8004044E	DATABASE_ERROR_RECORD_NOT_FOUND	CInstrument	AliasUserFromDatabase	SYSTEM account could not be found (line 260), or username could not be found (line 340).	
-2147220401	&H8004044E	DATABASE_ERROR_RECORD_NOT_FOUND	CSystem	fnLoadSystemUser	SYSTEM account could not be found.	The user database has not been configured correctly for the internal SYSTEM account.  You will not be able to edit DaCS files.  Please contact your System Administrator for assistance.
-2147220401	&H8004044F	DATABASE_ERROR_RECORD_CORRUPT	CInstrument	AliasUserFromDatabase	Database record does not have a valid digital signature for the SYSTEM user (line 270) or the username (line 350).	
-2147220401	&H8004044F	DATABASE_ERROR_RECORD_CORRUPT	CSystem	fnLoadSystemUser	Database record does not have a valid digital signature.	The user database has not been configured correctly for the internal SYSTEM account.  You will not be able to edit DaCS files.  Please contact your System Administrator for assistance.

## Compassoft DaCS Programmers Guide

-2147220400	&H80040450	USER_ERROR_ACCOUNT_LOCKED	CInstrument	AliasUserFromDatabase	User account is flagged as locked.	
-2147220399	&H80040451	USER_ERROR_CERTIFICATE_INVALID	CSystem	fnLoadSystemUser	System certificate is invalid	The user database has not been configured correctly for the internal SYSTEM account.  You will not be able to edit DaCS files.  Please contact your System Administrator for assistance.
-2147220399	&H80040451	USER_ERROR_CERTIFICATE_INVALID	CInstrument	AliasUserFromDatabase	System certificate is invalid	
-2147220398	&H80040452	WORKBOOK_ERROR_WORKBOOK_PROTECTION_FAILURE	CFile	ProtectWorkbookStructure	The target workbook is the Interface Workbook (line 110.)	
-2147220398	&H80040452	WORKBOOK_ERROR_WORKBOOK_PROTECTION_FAILURE	CFile	ProtectWorkbookStructure	bas_file.fnUnprotectWorkbookStructure failed to protect the workbook. (line 180)	
-2147220398	&H80040452	WORKBOOK_ERROR_WORKBOOK_PROTECTION_FAILURE	bas_file	fnProtectWorkbook	There was a failure when trying to protect the worksheets, chartsheets, or workbook structure.	An error occurred protecting the workbook. The file may not be completely protected. Please contact your System Administrator.
-2147220398	&H80040452	WORKBOOK_ERROR_WORKBOOK_PROTECTION_FAILURE	bas_file	fnUnprotectWorkbookStructure	There was a failure trying to protect the workbook.	
-2147220397	&H80040453	WORKBOOK_ERROR_WORKBOOK_UNPROTECTION_FAILURE	CFile	UnprotectWorkbookStructure	The target workbook is the Interface Workbook (line 110.)	
-2147220397	&H80040453	WORKBOOK_ERROR_WORKBOOK_UNPROTECTION_FAILURE	CFile	UnprotectWorkbookStructure	bas_file.fnUnprotectWorkbookStructure failed to unprotect the workbook. (line 180)	

## Chapter 8 CConstDaCS

-2147220397	&H80040453	WORKBOOK_ER ROR_WORKBOO K_UNPROTECTI ON_FAILURE	bas_file	fnUnprotectWork book	There was a failure trying to unprotect the workbook.	An error occurred trying to unprotect the workbook.
-2147220397	&H80040453	WORKBOOK_ER ROR_WORKBOO K_UNPROTECTI ON_FAILURE	bas_file	fnUnprotectWork bookStructure	The workbook could not be unprotected	
-2147220396	&H80040454	WORKBOOK_ER ROR_GHOST_PR OTECTION_FAIL URE	CFile	ProtectWorkbook Structure	bas_file.fnUnpro tectWorkbookStr ucture failed to protect the ghost workbook. (line 320)	
-2147220395	&H80040455	WORKBOOK_ER ROR_GHOST_UN PROTECTION_FA ILURE	CFile	UnprotectWorkb ookStructure	bas_file.fnUnpro tectWorkbookStr ucture failed to unprotect the ghost workbook. (line 320)	
-2147220394	&H80040456	WORKBOOK_ER ROR_NO_ACTIV E_WORKBOOK	CFile	UnprotectWorkb ookStructure	Excel.ActiveWor kbook is Nothing	
-2147220394	&H80040456	WORKBOOK_ER ROR_NO_ACTIV E_WORKBOOK	CFile	ProtectWorkbook Structure	Excel.ActiveWor kbook is Nothing	
-2147220393	&H80040457	WORKBOOK_ER ROR_WORKBOO K_NOT_FOUND			Reserved	
-2147220392	&H80040458	WORKBOOK_ER ROR_FAILED_FI LE_RESTRICTIO N_CHECK	bas_file	fnWorkbookOpe nEvent	The file failed the restriction checks.	
-2147220391	&H80040459	WORKBOOK_ER ROR_FILE_IS_R EAD_ONLY	bas_file	fnWorkbookOpe nEvent	The file is read- only or being modified by another user.	
-2147220390	&H8004045A	WORKBOOK_ER ROR_DIGITAL_S IGNATURE_MISS ING	bas_file	fnWorkbookOpe nEvent	The digital signature is missing from the workbook. (The last entry in the audit trail is not a digital signature.)	
-2147220389	&H8004045B	WORKBOOK_ER ROR_DIGITAL_S IGNATURE_INVA LID	bas_file	fnWorkbookOpe nEvent	The digital signature on the file is invalid.	
-2147220388	&H8004045C	WORKBOOK_ER ROR_DIGITAL_S IGNATURE_UN SUPPORTED_V ERSION	bas_file	fnWorkbookOpe nEvent	Invalid digital signature from version 1.1, 1, or the version information could not be determined.	

## Compassoft DaCS Programmers Guide

-2147220387	&H8004045D	WORKBOOK_ER ROR_ERROR_UP DATING_METAD ATA	bas_file	fnProtectWorkbo ok	Could not update the PASSWORD parameter on the _SYSTEM worksheet.	
-2147220386	&H8004045E	PROFILE_ERROR _COULD_NOT_C REATE_EXCHAN GE_KEYPAIR	CProfile	CreateExchange KeyPair	The session key could not be generated, or there was a problem generating or unloading the exchange keypair/public key.	
-2147220385	&H8004045F	PROFILE_ERROR _COULD_NOT_E NCRYPT_PROFIL E	CProfile	EncryptProfile	There was an error with the session key, the profile, or the exchange keys in trying to encrypt the profile.	
-2147220384	&H80040460	PROFILE_ERROR _COULD_NOT_D ECRYPT_PROFIL E	CProfile	DecryptProfile	There was an error with the session key, the profile, or the exchange keys in trying to decrypt the profile.	
-2147220383	&H80040461	CRYPTO_ERROR _DECRYPTION_F AILURE	bas_file	fnUnprotectWork book	The file's PASSWORD could not be decrypted.	An error occurred trying to unprotect the workbook.
-2147220383	&H80040461	CRYPTO_ERROR _DECRYPTION_F AILURE	bas_file	fnProtectWorkbo ok	The file's PASSWORD could not be decrypted, either the original PASSWORD (line 70), or the new PASSWORD (line 120), or the user password on a worksheet (line 200)	An error occurred protecting the workbook. The file may not be completely protected. Please contact your System Administrator.
-2147220383	&H80040461	CRYPTO_ERROR _DECRYPTION_F AILURE	bas_file	fnUnprotectWork bookStructure	The file's PASSWORD could not be decrypted.	

## Chapter 8 CConstDaCS

-2147220382	&H80040462	CRYPTO_ERROR_ENCRYPTION_FAILURE	bas_file	fnProtectWorkbook	The new password could not be encrypted.	An error occurred protecting the workbook. The file may not be completely protected. Please contact your System Administrator.
-2147220381	&H80040463	CRYPTO_ERROR_RANDOM_STRING_GENERATOR_FAILURE	bas_file	fnProtectWorkbook	There was a failure in the random string generator.	



# Index

---

## A

- About this Book • 5
- AboutDaCS • 33
- Action • 16
- ActiveWorkbookStatus • 66
  - wsCSystem\_WorkbookStatus\_Enum • 66
- AfterChange • 17
- AliasUser • 48
- AliasUserFromDatabase • 49
- AuditSilent • 67
- AuditSilentReason • 67
- AuditWorksheet • 17
- AutoCompleteTarget • 18
- AutoFillValue • 18

## B

- BeforeChange • 18
- BOF • 18
- Build • 67

## C

- CAudit • 15
  - Defaults • 16
  - MembersCAudit • 16
  - Using • 15
  - wsCAudit\_ObjectType • 21
- CAudit Members • 16
- CAudit Property Defaults • 16
- CConstDaCS • 73
  - DACS\_ERRORDACS\_ERROR • 74
  - Members • 73
  - Using • 73
- CConstDaCS Enums • 74
- CConstDaCS Members • 73
- CErrLogger • 25
  - Defaults • 26
  - Members/rCErrLogger • 26
  - Using • 25
- CErrLogger (LastError) • 25
- CErrLogger Members • 26
- CErrLogger Property Defaults • 26
- CFile • 33
  - MembersCFile • 33
  - Using • 33
- CFile (Menu) • 33
- CFile Members • 33
- Changes to Excel Programming • 13
- ChangeUserPassphrase • 33
- ChangeUserPassword • 33
- ChartWizard • 34
- ChildName • 18
- CInstrument • 45, 46
  - Defaults • 47

- Enums • 48
- Example Code • 55
- MembersCInstrument • 48
- Using • 46
- CInstrument Enumerations • 48
- CInstrument Example Code • 55
- CInstrument Members • 48
- CInstrument Property Defaults • 47
- Classes • 14
- ClassName • 26
- Clear • 26
- CloseWorkbook • 34
- CommitChange • 19
- CompanyName • 51
- Contacting Compasssoft Support • 6
- Conventions • 5
- CProfile • 57, 58
  - Enums • 60
  - Field Values • 53
  - Fields • 58
  - MembersCProfile • 61
  - Security, Cryptography • 61
  - Structure • 58
- CProfile Enumerations • 60
- CProfile Members • 61
- CreateNewDaCSWorkbook • 34
- CreateNewWorkbookFromTemplate • 35
- CSystem • 65
  - Defaults • 66
  - Members/rCSystem • 66
  - Using • 65
- CSystem Members • 66
- CSystem Property Defaults • 66

## D

- DaCS Documentation • 5
- DaCS Interface Workbook • 8
- DaCS Objects
  - CAudit • 15
  - CConstDaCS • 73
  - CErrLogger, LastError • 25
  - CFile • 33
  - CInstrument • 46
  - Classes • 14
  - CProfile • 58
  - CSystem • 65
  - Viewing • 13
- DaCSEnabled • 36
- DaCSInterface • 9
- DaCSRequestRelease • 9
- DaCSStatus • 67
  - wsCSystem\_DaCSStatus\_Enum • 67
- DateTimeStamp • 19
- DateTimeStampString • 19
- DecryptEncryptedProfile • 61
- DecryptProfile • 62
- Description • 27
- DisplayError • 27

## Compassoft DaCS Programmers Guide

DisplayErrorMessage • 27

### E

EnableAutoFill • 19

EncryptedProfile • 62

EncryptedProfileDataFormat • 62

Enums

DACS\_ERRORDACS\_ERROR • 74

DATA\_FORMAT • 60

wsCAudit\_ObjectType • 21

wsCInstrument\_ControlMode\_Enum • 48

wsCSystem\_DaCSStatus\_Enum • 67

wsCSystem\_WorkbookStatus\_Enum • 66

wsCSystem\_WorkbookStatusEnum • 71

wsSheetType\_Enum • 36

EOF • 19

ErrorFileName • 28

ErrorFilePath • 28

ExcelVersion • 68

### F

FileDirty • 20

### G

GetAuditObject • 68

GetExcelInformation • 28

GetInstrumentObject • 68

GetProfileObject • 51

### I

InsertSheet • 36

Instance • 9, 69

InstanceOK • 69

InstrumentID • 51

InstrumentName • 52

Interacting with DaCS from an External Process  
• 9

Interactive • 69

Interface Workbook

C++ • 9

DaCSInterface • 9

DaCSRequestRelease • 9

External Processes • 9

ReleaseExternalDaCSReferences • 9

Role • 8

SetInterfaceWorkbookSystemObject • 9

Introduction • 7

### L

LastError • 25, 28, 37, 52, 62, 69

License • 52

Line • 28

LoadError • 28

LoadExtendedUserProperties • 52

LogError • 29

LoginProfile • 52

### M

Macros • 12

Auditing • 12

Legacy • 13

Programming • 12

Menu • 26, 27, 28, 29, 30, 31, 69, 70, 73

Message • 29

MoveCopySheets • 37

### N

Number • 29

NumberFormatAfter • 20

NumberFormatBefore • 20

### O

Object Model • 13

ObjectName • 20

ObjectTypeExt • 21

OpenWorkbook • 37

### P

Parent • 70

ParentName • 21

ParseInstrumentFromProfile • 53

Position • 22

Preface • 5

Profile • 62

Profile Structure • 58

PROFILE\_FIELD\_DELIMITER • 73

PROFILE\_FIELD\_TAG • 73

ProfileDataFormat • 62

ProfileDecrypted • 63

PromptAudit • 22

ProtectWorkbookStructure • 38

### R

Read • 22

Reason • 22

ReasonMaxLength • 23

ReasonMinLength • 23

ReasonSpellCheck • 23

ReleaseExternalDaCSReferences • 9

RenameSheet • 38

Routine • 29

### S

SaveWorkbook • 39

SaveWorkbookAs • 39

Security and Cryptography in CProfile • 61

SessionKey • 63

SessionKeyDataFormat • 63

SetDateTimeStamp • 23

SetDefaults • 29

SetEnvironment • 70

SetInterfaceWorkbookSystemObject • 9, 70

SetProperties • 30

SetSheetProtection • 39

SetWorkbookProtection • 40

- ShowDaCSAdminPanel • 41
- Shutdown • 70
- SignatureVersion • 70
- SignWorksheets • 41
- SoftwareName • 54
- Source • 30
- Start • 9, 26, 27, 28, 29, 30, 31, 71
- SynchronizeGhostWorkbook • 41
- SystemCertificate • 54
- SystemCertificateKey • 54
- SystemID • 54
- SystemPublicKey • 54

**T**

- TimeFormat • 30

**U**

- UnhideWindow • 42
- UnprotectWorkbookStructure • 42
- UserFullName • 23
- UserID • 23
- Using CAudit • 15
- Using CConstDaCS • 73
- Using CErrLogger • 25
- Using CFile • 33
- Using CInstrument • 46
- Using CSystem • 65

**V**

- ValidateReason • 24
- Verbose • 31
- Version • 71
- Viewing the WSDACS Object Model • 13

**W**

- WorkbookStatus • 71
  - wsCSystem\_WorkbookStatusEnum • 71
- WorkbookStructureProtected • 42
- WriteAuditTrail • 24
- WriteErrorLog • 31
- wsSheetType\_Enum • 36