DEVELOPERS OF WORLD LEADING GIS SOFTWARE

Cadcorp SIS V6.0
Programming Reference Guide

All products in the Cadcorp SIS suite are created and owned by:

Cadcorp Ltd
Sterling Court
Norton Road
Stevenage
Herts
SG1 2JY
UK
Tel: + 44 (0)1438 747996
Fax: + 44 (0)1438 747997
Email: cadcorp@cadcorp.com
Website: www.cadcorp.com

Published January 1, 2003, by Cadcorp Ltd. ISBN 0-9544276-2-9

Contents

1	Cadcorp SIS Programming Reference Guide	1
2	Customising with GisLink	5
3	Cadcorp SIS Control	15
4	Cadcorp SIS OLE DB provider	35
5	Cadcorp SIS Active Server Component	39
6	Cadcorp SIS Map Server	67
7	Methods	75
8	Examples	223

Appendices

1	Availability of methods	299
2	Method summaries	309
3	ACOM commands	333
4	Cadcorp SIS properties	345
5	Global constants	411
6	Index dataset naming conventions	419
7	Setting up connections using data from OpenGIS Servers	427
8	ASCII character set	429
	Index	431

Chapter 1

Cadcorp SIS Programming Reference Guide

■ **Introduction** 1
■ **About this manual** 1
■ **Document conventions** 2

■ Introduction

There are two ways to extend and customise Cadcorp SIS products:

- use GisLink, together with Microsoft Visual Basic, to customise the Cadcorp SIS Map Manager, Cadcorp SIS Map Editor, and Cadcorp SIS Map Modeller applications. You can add your own commands to the menus, and remove system commands from them.
- use the Cadcorp SIS Control. This is a more integrated solution which allows you to define your own user interface to Cadcorp SIS functionality.

For web-based applications, you can program with:

- Cadcorp SIS Active Server Component, for large-scale applications
- Cadcorp SIS Map Server for small-scale applications

■ About this manual

This manual is organised as follows.

Chapter	Contains
2 Customising with GisLink	an introduction to customising Cadcorp SIS desktop products using Microsoft Visual Basic
3 Cadcorp SIS Control	an introduction to writing custom GIS applications using the Cadcorp SIS ActiveX Control
4 Cadcorp SIS OLE DB provider	instructions for accessing the Cadcorp SIS OLE DB Provider
5 Cadcorp SIS Active Server Component	instructions for writing Internet and Intranet applications using the Active Server Component
6 Cadcorp SIS Map Server	instructions for delivering Cadcorp SIS data on the web, using OpenGIS standards

Chapter	Contains
7 Methods	descriptions of all API methods in alphabetical order
8 Examples	example code showing methods in action
Appendix 1 Availability of methods	a list of methods and their availability in Cadcorp SIS products
Appendix 2 Method summaries	a list of methods arranged in functional groups
Appendix 3 ACOM commands	a list of ACOM commands in Cadcorp SIS
Appendix 4 Cadcorp SIS properties	a list of properties available in Cadcorp SIS
Appendix 5 Global constants	a list of the global constants in Cadcorp SIS
Appendix 6 Index dataset naming conventions	a list of all index dataset naming conventions available in Cadcorp SIS
Appendix 7 Setting up connections using data from OpenGIS Servers	information about setting up Cadcorp SIS with OpenGIS Web Map servers
Appendix 8 ASCII character set	the ASCII character set

The manual assumes a reasonable knowledge of the language(s) you are using to customise Cadcorp SIS, such as Microsoft Visual Basic, C++, VBScript, or HTML.

■ Document conventions

◆ Cadcorp SIS products

In this manual, the names of Cadcorp SIS products are abbreviated as follows:

MV Cadcorp SIS Map Viewer

MM Cadcorp SIS Map Manager

ME Cadcorp SIS Map Editor

MD Cadcorp SIS Map Modeller

OV Cadcorp SIS ActiveX Viewer Control

OM Cadcorp SIS ActiveX Manager Control

OD Cadcorp SIS ActiveX Modeller Control

ASC Cadcorp SIS Active Server Component

◆ Typographic conventions

`Monospaced` type is used for programming code, such as Visual Basic and C(++), HTML source code, and method and property names.

`Oblique monospaced` type is used for placeholders in code and syntax definitions, such as variables.

In the code listings, a space followed by an underscore (_) at the end of a line continues a line of code. For example:

```
GisAddCommand "&Map|&Zoom|Out by 10", "Zoom Out by a factor of 10", _
    "Item", 0, -1, "", ""
```

Chapter 2

Customising with GisLink

- Customising using GisLink 5
- Getting started. 5
- Establishing a connection 6
- Cadcorp SIS commands 7
- Adding custom commands 7
- Adding commands to the Cadcorp SIS pop-up menu 8
- Removing system commands 9
- Running system commands 9
- GisLink Triggers. 10
- GisLink user position input 13
- GisLink debugging 14

■ Customising using GisLink

The Cadcorp SIS Map Manager, Cadcorp SIS Map Editor, and Cadcorp SIS Map Modeller desktop applications can be customised using GisLink to perform particular tasks, to automate some of the built-in functions of the system, or to extend the applications' capabilities in specific ways.

GisLink is a set of methods that make use of the Windows messaging system to allow Microsoft Visual Basic programs to communicate with Cadcorp SIS applications.

You can write GisLink customisations using Visual Basic versions 4.0 (32-bit), 5.0, and 6.0.

The methods in this manual are documented for both GisLink and the Cadcorp SIS Control, although all GisLink calls require a prefix of `Gis`. For example, you use `GisAddCommand` for `AddCommand`.

■ Getting started

◆ The GisLink.bas module file

The GisLink methods and constants available to Visual Basic are contained in an automatically generated Visual Basic module file, with the *.bas extension.

To generate the *.bas file:

1. Run the Cadcorp SIS application (Cadcorp SIS Map Manager, Cadcorp SIS Map Editor or Cadcorp SIS Map Modeller) and choose **Program Window** from the Tools menu.
2. Choose the **Generate Programming File** command from the system menu in the Program Window.

3 Choose the GisLink option and the appropriate Visual Basic version from the drop-down list.
4 Fill in the Filename field, either by typing a filename or choosing one using the Browse button. The file will typically be called GISLINK.BAS, but you can choose any name.
5 Click **OK** to generate the file.

You should now add the generated GisLink.bas module file to your new Visual Basic project.

Cadcorp SIS Map Manager, Cadcorp SIS Map Editor and Cadcorp SIS Map Modeller all contain different levels of functionality, and therefore different GisLink methods. This means the GisLink.bas files created by each of the applications will be different. Your programs will stop with errors if you try to call a Cadcorp SIS Map Editor or Cadcorp SIS Map Modeller method from a GisLink customisation running with Cadcorp SIS Map Manager, or a Cadcorp SIS Map Modeller method when running with Cadcorp SIS Map Editor or Cadcorp SIS Map Manager.

◆ The Visual Basic Startup Form

Visual Basic has the concept of a startup object, which you set by selecting **Properties** from the Project menu in Visual Basic. By default, this is the first form you create in your Visual Basic project, although you can change this.

GisLink customisations should set the *visible* property of the startup form to False, because this form is never displayed, it simply controls the link between Visual Basic and the Cadcorp SIS application.

Your Visual Basic program and Cadcorp SIS communicate using the Windows *handle* of this startup form. Because every form, dialog, and control in Windows has a unique handle, you are able to run several GisLink customisations simultaneously, each having its own channel of communication with Cadcorp SIS.

■ Establishing a connection

The connection between the Visual Basic program and the currently running Cadcorp SIS application is established in the `Form_Load` event of the startup form using the `SetupLink` method:

```
Sub Form_Load()
   If GisSetupLink(hWnd) = 0 Then
      MsgBox “Error connecting to SIS application”
      End
   End If
End Sub
```

The `GisSetupLink` method broadcasts the startup form’s handle (`hWnd`) to the Windows Messaging system. If Cadcorp SIS is running, it will pick up this message and a link between your program and Cadcorp SIS is established.

◆ The ListCaps button

If a connection is successfully established, the Cadcorp SIS application will look for a CommandButton on the startup form with the caption ListCaps (an abbreviation of

ListCapabilities). The name of the button is unimportant, but the caption *must* be ListCaps. If Cadcorp SIS detects this button it will invoke its Click event, running any code you write in this event.

It is in the Click event of this button that you should add the commands, register the groups, and perform any other setup specific to your customisation. You may also choose to remove some commands from the Cadcorp SIS application user interface.

■ Cadcorp SIS commands

Whenever a menu option is selected in Cadcorp SIS Map Manager, Cadcorp SIS Map Editor, and Cadcorp SIS Map Modeller, a *command* is invoked. Each of these commands has a name, which users are normally unaware of. The Program window which you used to generate the Gislink.bas file displays these commands when they are selected, enabling Visual Basic programmers to find the command name and use it within a custom application. All system command names begin with the letters `ACom`. For details on how to call a Cadcorp SIS command from your Visual Basic program, ➲page 9, **Running system commands**

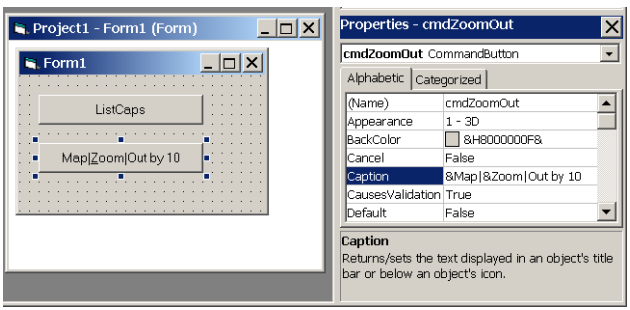
■ Adding custom commands

◆ Adding commands to the Cadcorp SIS menu bar

GisLink customisations can add custom commands to both the main application menu and the local (usually right mouse button) pop-up menu. Custom commands are added by calling the `AddCommand` method. Each custom command to be used by the GisLink customisation must be added in the Click event of the button captioned ListCaps. Each custom command must have a corresponding command button on the start-up form, whose caption exactly matches the *`menu$`* argument given in the `AddCommand` method.

The following code registers the custom command **Out by 10**, which will appear on the Zoom sub-menu of the Map menu of the Cadcorp SIS application.

```
Sub cmdListCaps_Click()
  GisAddCommand "&Map|&Zoom|Out by 10", "Zoom Out by a factor of 10", _
    "Item", 0, -1, "", ""
  GisRelease
End Sub
```



Notice that Map is prefixed with an ampersand (&), which indicates that the following letter is to be underscored and used as the keyboard shortcut. If you want to add com-

mands to the existing Cadcorp SIS menus, the text used in the `GisAddCommand` method must *exactly* match the text which appears on the Cadcorp SIS menu.

Notice that the `GisRelease` method is called after the `GisAddCommand` function. Think of `GisRelease` as being the 'full stop' (period) at the end of the conversation with Cadcorp SIS.

For more information about the `GisAddCommand` method, ➲page 77, **AddCommand (Cadcorp SIS Control)**.

When the user selects **Map>Zoom>Out by 10** from the Cadcorp SIS menu, GisLink will look for a button with the caption `&Map|&Zoom|Out by 10` on your Visual Basic startup form. If it finds such a button, it will invoke its Click event, and perform the code you have written there, for example:

```
Sub cmdZoomOut_Click()
  GisZoomView 10
  GisRelease
End Sub
```

■ Adding commands to the Cadcorp SIS pop-up menu

Commands added to the Cadcorp SIS main menu must always contain the pipe (**|**) symbol to define the hierarchy of the drop-down menu. Commands without this symbol will appear on the local pop-up menu. If you want to create a hierarchy of commands on the local pop-up menu, use the hash (**#**) symbol rather than a pipe (**|**).

```
Sub cmdListCaps_Click()
   GisAddCommand "Redraw", "Redraw the map", "Item", 0, -1, "", ""
   GisAddCommand "Zoom#In", "Zoom in by 2", "Item", 0, -1, "", ""
   GisAddCommand "Zoom#Out", "Zoom out by 2", "Item", 0, -1, "", ""
   GisRelease
End Sub
```

When the user selects **Zoom>In** from the Cadcorp SIS pop-up menu, GisLink will look for a button with the caption `Zoom#In` on your Visual Basic startup form, and invoke the click event:

```
Sub cmdPopupZoomIn_Click()
   GisZoomView 0.5
   GisRelease
End Sub
```

You must call the `GisRelease` method at the end of the `ListCaps_Click`, to return control back to the Cadcorp SIS application user.

■ Removing system commands

Your customisation can remove system commands, to simplify the `Cadcorp SIS` interface presented to the user. System commands are typically removed (and re-added if necessary) in the Load event of the startup form, using the `AllowCommands` method:

```
' Remove all system commands.
GisAllowCommands SIS_COM_NONE, ""

' Remove the zoom in and zoom out commands.
GisAllowCommands SIS_COM_REMOVE, "AComZoomIn2 AComZoomOut"

' Put the zoom in and zoom out commands back.
GisAllowCommands SIS_COM_ADD, "AComZoomIn2 AComZoomOut"

' Allow all commands available for the current SIS application.
GisAllowCommands SIS_COM_ALL, ""
```

■ Running system commands

Although the GisLink API contains many functions to perform operations in Cadcorp SIS applications, it is sometimes easier to invoke one of the system commands than to write Visual Basic code to produce the same result. Any of the commands available to the user can also be invoked through Visual Basic, although some are obviously more useful than others.

System commands fall into two categories:

- one-shot commands, which perform an action without user mouse or positional input, such as **Map>Redraw**)
- callback commands, which require user mouse or positional input, such as **Construct>Geometry 2D>Line**)

One-shot commands are run using the `CallCommand` method. Callback commands are started using the `SwitchCommand` method.

For example, the following code starts by calling a one-shot command, **Redraw**, then starts drawing a line.

```
' Redraw the view by command.
GisCallCommand “AComRedraw”

' Start drawing a line.
GisSwitchCommand “AComLine”

' Release control back to the user.
GisRelease
```

The one-shot redraw command will be started and completed in the duration of the `CallCommand` method. However, the callback command exists beyond the duration of the `SwitchCommand` method, in this case until the user presses Enter or Ctrl-Enter to complete the line, or Escape to quit the command. The progress of callback commands is monitored using *triggers*. ➲page 10, **GisLink Triggers**

The `DoCommand` method can be used for either one-shot or a callback commands, and it will choose the appropriate action. The only exception to this is when you want to invoke a command which displays a modal dialog, such as `AComLayers` (display the

Overlays dialog), `AComExport` (display the Export dialog) and `AComPrintTemplate` (display the Print Template Wizard dialog). Because these commands expect further input from the user, they are callback commands, but they differ from other callback commands in that they have only Succeeded and Failed triggers. In these cases you must use the `SwitchCommand` method.

■ GisLink Triggers

Every command fires triggers that can be monitored by a Visual Basic program.

One-shot commands	Callback commands
Succeeded	Snap
Failed	KeyBack
	KeyEnter
	KeyTab
	End

Here is some typical output from the Program Window (with comments added) when the user draws a line with the **Contruct>Geometry 2D>Line** command, and then uses the local command Convert To Area on the resulting line item:

Program window	Comment
`Trigger AComSelectSlide::End`	Line callback command automatically ends the normal Select command
`Trigger AComLineEx::Snap`	the starting position
`Trigger AComLineEx::Snap`	the second position
`Trigger AComLineEx::Snap`	the third position
`Trigger AComLineEx::Snap`	the fourth position
`Trigger AComLineEx::Snap`	the fifth position
`Trigger AComLineEx::KeyBack`	the fifth position is deleted
`Trigger AComLineEx::KeyEnter`	the user presses the Enter key to create the line item
`Trigger AComLineEx::End`	user presses the Escape key to return to the Select command
`Trigger AComSelectSlide::Snap`	the new line is selected
`Trigger AComMakeArea::Succeeded`	the user selects the Convert to Area one-shot command

◆ Using triggers

To make use of these triggers in your Visual Basic program you will need to **register** them. Triggers can be registered anywhere within your Visual Basic code, and can be unregistered and re-registered as many times as you like. Once a trigger is registered you can write code to monitor it, and respond as necessary. Visual Basic forms and

controls have events such as Click, MouseOver, and so on, which you can write code to respond to. Triggers work in a similar way, except you need to assign the trigger to a command button, and write your code in the Click event of the button.

To monitor the progress of the **Construct>Geometry 2D>Line** command, the `AComLineEx` triggers would need to be registered. When the user starts the command from the Cadcorp SIS menu, or when your program calls the command, you can respond to the triggers as you choose. The following example displays a message box every time the user completes a line by double-clicking the mouse:

First register the trigger in the Form Load event:

```
Private Sub Form_Load()
  If GisSetupLink(hwnd) = 0 Then
    MsgBox "Error connecting to SIS application"
    End
  End If

  GisRegisterTrigger "AComLineEx::DblClick", "LineExDblClick"

End Sub
```

The trigger event, in this case `AComLineEx::DblClick`, is made up of the command name, two colons, then the event type. You can copy and paste trigger events directly into your code from the Cadcorp SIS Program Window. `LineExDblClick` is the caption of a command button on the startup form. Write the code to respond to the trigger in the click event of this button:

```
Private Sub cmdDoubleClick_Click()
  MsgBox "Snappy Snapping!", vbInformation, "Trigger Example"
  GisRelease
End Sub
```

To unregister a trigger, simply register it with an empty string instead of a button caption:

```
GisRegisterTrigger "AComLineEx::DblClick", ""
GisRelease
```

Sometimes you may want a trigger to set a flag, and monitor this flag in a separate section of code. To do this, declare the flags at form level with the Private statement in the Declarations section of the form, and set the flags to True when the trigger is detected by the click event of the command button:

```
Private bAreaEnter As Boolean
Private bAreaDblClick As Boolean

Private Sub cmdAreaDblClick_Click()
  bAreaDblClick = True
End Sub

Private Sub cmdAreaEnter_Click()
  bAreaEnter = True
End Sub
```

```
Private Sub cmdAreaEnd_Click()
   bAreaEnd = True
End Sub
```

Now add a command in the Click event of the ListCaps button, and add a button to the startup form with its caption set to that command:

```
Sub cmdListCaps_Click()
   GisAddCommand "&Example|Draw Area", "Draw a Red Area", "Item", 0, -1, _
     "", ""
   GisRelease
End Sub
```

Register the triggers in the click event of your custom command, monitor the trigger flags, respond accordingly, then unregister the triggers.

```
Sub cmdRedArea_Click()
   ' register the triggers
   GisRegisterTrigger "AcomAreaEx::KeyEnter", "AreaEnter"
   GisRegisterTrigger "AcomAreaEx::DblClick", "AreaDblClick"
   GisRegisterTrigger "AcomAreaEx::End", "AreaEnd"

   ' Start the Area command and release control to the user
   GisDoCommand "AComAreaEx"
   GisRelease

   ' wait for a trigger to set a flag
   Do
      DoEvents
      If bAreaEnter = True Or bAreaDblClick = True Then
         'open the area item, set its _brush$ to Red
         GisOpenSel 0
         GisSetStr SIS_OT_CURITEM, 0, "_brush$", "Red"
         Exit Do
      ElseIf bAreaEnd = True Then
         Exit Do
      End If
   Loop

   ' unregister the triggers, reset the flags,
   ' then end the Area command
   GisRegisterTrigger "AcomAreaEx::KeyEnter", ""
   GisRegisterTrigger "AcomAreaEx::DblClick", ""
   GisRegisterTrigger "AcomAreaEx::End", ""
   bAreaEnter = False: bAreaDblClick = False: bAreaEnd = False
   GisDoCommand "AComSelectSlide"
   GisRelease
End Sub
```

◆ Triggers and deadlocks

GisLink is implemented using Windows messages. The Cadcorp SIS application and Visual Basic take it in turns to process messages. In particular, the Cadcorp SIS application cannot call Visual Basic halfway through processing one of its messages,

because this could cause a message deadlock, hanging your system (or the 16 bit sub-system of Windows 9x and Windows NT if you are using a 16-bit Visual Basic program).

The Cadcorp SIS application queues up triggers while processing one of its messages, and then gives Visual Basic a chance to act on the last trigger it registered. This means that Visual Basic can act on at most one trigger generated during the processing of one Windows message to the Cadcorp SIS application.

■ GisLink user position input

GisLink customisations will often require the user to input a position or positions during a custom command. This will often be as part of a set of procedural actions, eg prompt the user for text, then get a position from the user, then create a Text item using the results, and so on.

This can be achieved using either the `GetPos` or the `GetPosEx` methods. Both `GetPos` and `GetPosEx` operate by using a special system callback command and a Visual Basic event loop. The event loop allows other applications to operate, in particular the Cadcorp SIS application to which the customisation is connected, while maintaining a procedural coding approach within the Visual Basic customisation.

`GetPos` will return only if the user inputs a position using the mouse or by typing in the Position Bar, or if the user presses the Escape key. `GetPosEx` returns with the same actions as `GetPos`, and also if the user presses the Enter or Backspace keys. Both `GetPos` and `GetPosEx` will return if the command ends because the user selects another callback command.

◆ Transparent commands

There is a special subset of callback commands, called transparent commands, which do not end an existing callback command. Transparent commands are typically for view manipulation, such as the zooming and panning commands. All one-shot commands work transparently.

It is possible to use `GetPosEx` to get more than one input position, terminated by pressing the Enter key, the Escape key, or starting another callback command:

```
Dim lResponse As Long
Dim x As Double
Dim y As Double
Dim z As Double

Do
 lResponse = GisGetPosEx(x, y, z)
 Select Case lResponse
  Case SIS_ARG_ENTER
   ' Ending on Enter
   Exit Do
  Case SIS_ARG_ESCAPE
   ' Ending on Escape (this can be caused by starting another,
   ' non-transparent command)
   Exit Do
```

```
    Case SIS_ARG_POSITION
      ' Snap position is in x, y, z
    Case SIS_ARG_BACKSPACE
      ' Backspace key pressed
    End Select
  Loop
```

■ GisLink debugging

◆ Program Window

The Program Window monitors all GisLink method calls, triggers and errors, and outputs information about GisLink customisations connected to the Cadcorp SIS application. Options on the system menu of the Program Window allow you to switch on and off the method calls, triggers, and error output. The Program Window has an Always on Top option that forces it on top of other windows all the time.

When debugging a GisLink customisation, you sometimes need to regain the control from the customisation, particularly when the customisation code has failed to call Release. The Stop Waiting option on the Program Window system menu will immediately return control to the Cadcorp SIS application. In addition pressing Ctrl-Break will return control to the Cadcorp SIS application.

◆ API errors

Any errors generated by an API method are output in the Program Window, and can also be checked in a Visual Basic program. The system variable `_ExecError&` contains the error code from the last API method called. The GisLink.bas file contains a list of the errors that can occur when using GisLink. To query the last error status, use the following code:

```
Dim lError As Long
lError = GisGetInt(SIS_OT_SYSTEM, 0, "_ExecError&")
```

You can query the error directly in the Visual Basic Immediate window by typing:

```
Print GisGetInt(SIS_OT_SYSTEM, 0, "_ExecError&")
```

Chapter 3

Cadcorp SIS Control

- Writing applications using the Cadcorp SIS Control 15
- Cadcorp SIS Control methods 15
- Getting started using Microsoft Visual Basic 6.0 16
- Getting started using Microsoft Visual C++ 6.0 17
- Keyboard focus 19
- Multiple Cadcorp SIS Controls 19
- System settings 20
- Single Document Interface (SDI) 20
- Multiple Document Interface (MDI) 20
- Using the Program Window 20
- Cadcorp SIS Control debugging 21
- On-line help 21
- Cadcorp SIS Control properties 21
- Cadcorp SIS Control commands 24
- Adding custom commands 24
- Running system commands 24
- Cadcorp SIS Control events 25
- Position input 28
- Cadcorp SIS Control licensing 33

■ Writing applications using the Cadcorp SIS Control

The Cadcorp SIS Control Development Module is a 32-bit ActiveX Control (OCX) which you can use from Microsoft Visual Basic 4.0 (32-bit), 5.0 or 6.0, Microsoft Visual C++ 4.0 or 5.0, and other ActiveX Control containers.

The Cadcorp SIS Control provides an Application Programming Interface (API) in the form of Properties, Methods, and Events. Properties can be set to affect the appearance or state of the control, Methods can be called to perform GIS-related functions, and Events can be monitored to retrieve the user's actions.

The Cadcorp SIS Control is licensed, like the Cadcorp SIS applications, using a hardware lock (dongle). The licensing covers both design time, when an application is being written, and run time, when the application is being run. ➲page 33, **Cadcorp SIS Control licensing**

■ Cadcorp SIS Control methods

Most Visual Basic controls provide *methods* to enable the programmer to utilise the functionality of the control. A standard ListBox control in Visual Basic has nine methods, such as `AddItem`, `RemoveItem` and `Clear`. The Cadcorp SIS Control offers a great

deal more functionality than a ListBox, and it has a correspondingly larger number of methods. Methods can be called only at run-time, and allow the programmer to manipulate the control.

The Cadcorp SIS Control offers a large number of methods which carry out mapping-related functions, such as adding data files, displaying data from databases, manipulating the view scale, drawing lines and text and performing spatial searches.

The methods in the Cadcorp SIS Control closely resemble the functions available in GisLink customisations. You will see that not all methods are available to all levels of use of the control. For example, an application which sets the Level property of the control to `SIS_LEVEL_MANAGER` will not be able to use the `CreateExtrusion` method, because that 3D modelling method is specific to the Modeller level of the control.

Definitions of each method can be found in ➲Chapter 7: **"Methods"**.

◆ Adding a Cadcorp SIS Control

The means of adding a Cadcorp SIS Control will be specific to the ActiveX Control container program being used. This manual describes the process of starting a Cadcorp SIS Control project using Visual Basic 6.0 and C++.

A Cadcorp SIS Control design-time licence is required to create applications. Without a licence you may find that you can add the Cadcorp SIS Control component to the project, but you will be unable to create an instance of it on a form. If you attempt to open a project which contains the Cadcorp SIS Control, you will be notified that a licence has not been detected.

➲page 33, **Cadcorp SIS Control licensing**

■ Getting started using Microsoft Visual Basic 6.0

This section assumes some knowledge of Visual Basic. See the Visual Basic documentation for more help on Visual Basic.

◆ SisConst.bas

Many of the Cadcorp SIS Control methods and properties require values as arguments, or return values from function calls. These values are defined as constants with meaningful names. The Cadcorp SIS Control constants available to the Visual Basic programmer are contained in an automatically-generated Visual Basic module file, with the *.bas extension.

To create the SisConst.bas file:

1. Run a Cadcorp SIS application that can be customised using GisLink (Cadcorp SIS Map Manager, Cadcorp SIS Map Editor or Cadcorp SIS Map Modeller) and select the **Tools>Program Window** command.
2. Select the Generate Programming File command from the system menu in the Program Window.
3. Choose the Cadcorp SIS Control option, and the appropriate Visual Basic version from the drop-down list.
4. Fill in the Filename field, either by typing a filename or by choosing one using the Browse button. The file will typically be called SisConst.bas, but you can choose any name.

5 Click OK to generate the file.

The constants file can be distributed and shared by many developers, because its content changes only with new releases of the Cadcorp SIS software.

Below is an example of a section of the Cadcorp SIS constants file:

```
' Cadcorp SIS Control Licence Levels.
Global Const SIS_LEVEL_UNLICENSED = 0
Global Const SIS_LEVEL_MANAGER = 1
Global Const SIS_LEVEL_MODELLER = 2
Global Const SIS_LEVEL_VIEWER = 3
```

If you program in Delphi or other languages, you can open this file in any text editor, where you change the syntax of the constants' declarations to suit your chosen programming language.

◆ Adding the Cadcorp SIS Control to a Visual Basic project

1 Start a new Visual Basic Project.
2 Add the SisConst.bas file, created above, to the project, using **Project>Add File**.
3 Choose the Components command from the Project menu, or from the local menu in the Toolbox window, or press Ctrl-T.

   The list of controls on the Controls tab should contain the Cadcorp SIS Control. If the Cadcorp SIS Control is not in the list, check that it is correctly installed and registered.
4 Find the Cadcorp SIS Control in the list, check the box next to it, and then click OK.
5 The Cadcorp SIS icon should now be visible in the Toolbox. Click this icon to create an instance of the Cadcorp SIS Control on your form. Press the mouse key over the form and hold it down to drag out the Cadcorp SIS Control to the desired size. Release the mouse key to create the Cadcorp SIS Control. The Cadcorp SIS Control draws the Cadcorp SIS icon. Rename the Cadcorp SIS Control to SIS using the Properties Window.
6 Go to the Load event of the form and type the following code:

```
Sis.Level = SIS_LEVEL_MANAGER
Sis.SetAxesPrj "*APrjNatGrid"
Sis.SetViewPrj "*APrjNatGrid"
Sis.CreateBackdropOverlay 0, "GB National Grid"
```

7 Choose Start from the Run menu, or press F5 to start the program. When the program starts, the Cadcorp SIS Control should show the outline of the United Kingdom. You can manipulate the map view with the scroll bars, or the middle mouse button, or the mouse-wheel where available.

■ Getting started using Microsoft Visual C++ 6.0

This section assumes a basic knowledge of Visual C++, the Microsoft Foundation Classes, and C++.

◆ SisConst.h

The Cadcorp SIS Control constants available to Visual C++ are contained in an automatically generated C/C++ header file, with the *.h extension.

To create the SisConst.h file:

1. Run a Cadcorp SIS application that can be customised using GisLink (Cadcorp SIS Map Manager, Cadcorp SIS Map Editor or Cadcorp SIS Map Modeller) and choose the Program Window command from the Tools menu.
2. Choose the Generate Programming File command from the system menu in the Program Window.
3. Choose the Cadcorp SIS Control option, and the C/C++ Header from the drop-down list.
4. Fill in the Filename: field, either by typing a filename or by choosing one using the Browse button. The file will typically be called SisConst.h, but you can choose any name.
5. Click the OK button to automatically generate the file.

◆ Adding the Cadcorp SIS Control to a Visual C++ Project Workspace

1. Create a new Visual C++ Project Workspace using the MFC AppWizard (*.exe) option.
2. On the MFC AppWizard Step 1, choose the Dialog based option. On the MFC AppWizard Step 2, choose the ActiveX Controls option. Press the Finish button to create the template application.
3. Select the Project-Add To Project-Components and Controls command, and choose the Registered ActiveX Controls ‘folder’. The list should contain the Cadcorp SIS Control. If the Cadcorp SIS Control is not in the list, check that it is correctly installed and registered.
4. Find the Cadcorp SIS Control in the list, select it, and then click the Insert button.
5. Click OK in the Confirm Classes, editing the class and filenames first if required.
6. Click the Close button in the Components and Controls Gallery dialog.
7. Select the Resources tab in the Project Workspace.
8. Find and select the application dialog.
   This will be called IDD_*`your_application_name`*_DIALOG.
9. Make the Controls toolbar visible. The Controls toolbar should contain the Cadcorp SIS Control icon. Click this icon to create an instance of the Cadcorp SIS Control on your dialog. Press the mouse key over the dialog and hold it down to drag out the Cadcorp SIS Control to the desired size. Release the mouse key to create the Cadcorp SIS Control. Use the local menu’s Properties command to change the ID of the Cadcorp SIS Control to `IDC_SIS`.
10. Select the View-Class Wizard command, or press Ctrl-W, and choose the Member Variables tab. Select IDC_SIS from the Control IDs list and click the Add Variable button. Append `sis` to the `m_` string in the Member Variable Name: field, and press OK. Click OK on the MFC ClassWizard dialog to accept the changes.
11. Go to the Class View tab of the Project Workspace and double-click on the `OnInitDialog` method of the `C`*`your_application_name`*`Dlg` class.

12 Add the following code after the `// TODO:` comment:

```
m_sis.SetLevel(SIS_LEVEL_MANAGER);
m_sis.SetAxesPrj("*APrjNatGrid");
m_sis.SetViewPrj("*APrjNatGrid");
m_sis.CreateBackdropOverlay(0,"GB National Grid");
```

13 Go to the top of the file and add the following code after the `#include 'stdafx.h'` statement:

```
#include 'Sisconst.h'
// Use the pathname of the file created above.
```

14 Select the Build>Build <*your_application_name*>.exe command, or press F7 to build the program.

15 Select the Build>Execute <*your_application_name*>.exe command, or press F5, to start the program. When the program starts, the Cadcorp SIS Control should show the outline of the United Kingdom. You can manipulate the map view using the scroll bars, or the middle mouse button, or the mouse wheel where appropriate.

■ Keyboard focus

For the cursor and numeric keypad keyboard scrolling to work, the Cadcorp SIS Control must have keyboard focus, so that all keyboard presses are directed to it, rather than to any other control. It is good practice to call `SetFocus` after starting commands in a Cadcorp SIS Control and, in Visual Basic, in the `GotFocus` event which will ensure that keyboard scrolling works after the user clicks in the Cadcorp SIS Control.

■ Multiple Cadcorp SIS Controls

Many instances of the Cadcorp SIS Control can be added to an application, for example, to have a main view and a keymap. In this case, certain settings will be shared between all of the Cadcorp SIS Controls:

- named object libraries
- units
- default co-ordinate system
- licence level
- datasets
- item defaults
- options
- system variables
- named lists
- named seeds
- application-specific local commands
- named tables
- database recordsets

It is also possible, in the same way as in Cadcorp SIS applications, to run interactive viewing commands across multiple Cadcorp SIS Controls. For example, you could

start the **Map>Zoom>Box** or **Map>Pan>Snap** command in one Cadcorp SIS Control, but receive mouse snaps from another, such as a key map. However, unlike the Cadcorp SIS applications, the focus will not be returned to the window that started the command, because the original window could be hidden, or disabled or on another dialog, and so on. It is therefore the responsibility of the application to reset the focus to the original Cadcorp SIS Control, typically in the `CommandAction` event of the key-map control.

Take care with named tables (created using `CreateDbTable`) and database recordsets (created using `DefineRecordSet`). Both of these are common to the Cadcorp SIS Controls in an application. However, when the last Cadcorp SIS Control is deleted, any named table and database recordsets will be destroyed.

■ System settings

The Cadcorp SIS Control does not save any system settings in the Windows Registry because more than one application (written using the Cadcorp SIS Control but requiring different system settings) could be installed on the same computer. You must therefore set all required system settings when the application starts.

■ Single Document Interface (SDI)

Your application can choose to present a SDI user interface instead of being dialog-based.

In Visual Basic, the Cadcorp SIS Control can be used in exactly the same way, but its owning form must be modified to be a SDI child window.

In Visual C++, it is possible to create a Cadcorp SIS Control as a child of the `CView`-derived class that the SDI application uses. If you take this approach, however, the application cannot easily receive Cadcorp SIS Control events. To receive events, the view class must be derived from `CFormView` and the dialog template used in the `CFormView`-derived class should contain the Cadcorp SIS Control. This allows the application to address the Cadcorp SIS Control in the same way as a Cadcorp SIS Control on a dialog.

■ Multiple Document Interface (MDI)

It is possible to create multiple views of the same Cadcorp SIS Control in separate MDI children or in splitter windows, because Cadcorp SIS Controls have the Swd property. ➲page 21, **Cadcorp SIS Control properties**

Setting the Swd property of one Cadcorp SIS Control to that of another Cadcorp SIS Control will make the second Cadcorp SIS Control contain the same overlays and view as the original.

The same restrictions apply as for SDI applications.

■ Using the Program Window

The Program Window is a free-floating window which, in a Cadcorp SIS application, is a child of the main window, and is thus common to all the child windows of the Cadcorp SIS application. In the Cadcorp SIS Control, however, there is no main win-

dow, only one or more instances of the control, which share, among other things like datasets and Named Object Libraries, the Program window. This window will be created as a child of the most appropriate window *at the time the Control is loaded*. It is up to the application programmer to ensure that this parent window will always be available, because hiding it will force its children (including the Program Window) to be hidden.

In practice, it is best to avoid use of the Program Window except for debugging.

■ Cadcorp SIS Control debugging

◆ Program Window

The Program Window monitors all Cadcorp SIS Control method calls and errors. Options are available on the system menu of the Program Window to switch on and off the method calls, triggers, and error output. The Program Window has an Always on Top option that forces it to always appear on top of other windows.

◆ API errors

Any errors incurred by an API method are output in the Program Window, and can also be checked in a Visual Basic program.

All API *subroutines* return an error code, as opposed to *functions* that return a value. This error code will have the same value as the system variable `ASysVarExecError`. Your code can check the value of `ASysVarExecError` after function calls to check that they succeeded.

The `GetErrorString` method can be used to get a textual version of an error number. `GetErrorString(-1)` returns the most recent error message.

The SisConst.bas/SisConst.h file contains a list of the errors that can occur when using the Cadcorp SIS Control.

■ On-line help

Cadcorp SIS dialogs do not have a Help button. Instead they have a ? button on the dialog frame which uses context-sensitive help. The text for this help is built into the Cadcorp SIS Control so no other files are required. However, there are some dialogs, such as those shown by the **Construct>Geometry 2D>Line** and **Construct>Geometry 2D>Area** commands, that do not have a ? button. These dialogs recognise the F1 function key, and they attempt to show the Cadcorp SIS on-line help file named SIS.CHM. This is in the same directory as the Cadcorp SIS Control SIS.OCX file, at the help page for the command. This file should exist, but can be replaced by an application-specific help file. Alternatively, the few commands which require the help file should be avoided.

■ Cadcorp SIS Control properties

Controls can have a number of Properties associated with them. For example, a ListBox has the `BackColor`, `DataSource`, and `MultiSelect` properties among others. Properties can be set at design time, or at run time, and typically set the appearance of a control.

The Cadcorp SIS Control has standard control properties, such as Top, Left, and Visible, and the additional properties listed below. The icons alongside the property name indicate design time (✎), run time, (➔), or both (✎➔).

`BorderBevel`

`CheckNetworkDongle`

`CommandMessage`

`CommandPrompt`

`Display`

`Level`

`ShowWaitCursor`

`Swd`

`TrackMouse`

The `Enabled` property operates on the Cadcorp SIS Control in the same way as other Visual Basic controls, that is, it prevents a user from interacting with the control using the keyboard or mouse. Unlike most other controls, the Cadcorp SIS Control does not change appearance when `Enabled` is set to False.

✎ **BorderBevel**

Syntax `Sis.BorderBevel = Boolean`

Settings either:

True the control is drawn with a surrounding 3D bevel

False the control is drawn without a surrounding 3D bevel

➔ **CheckNetworkDongle**

Syntax `Sis.CheckNetworkDongle = Boolean`

Settings either:

True the control will search the local area network for the presence of a multi-user network dongle

False the application will report a licensing error if a dongle is not detected on the local parallel port of your PC

For information about the `CheckNetworkDongle` property, together with the `Level` property, ➲page 33, **Cadcorp SIS Control licensing**.

➔ **CommandMessage**

sets and gets the current Cadcorp SIS Control command message string

Syntax `Sis.CommandMessage = String`

Notes The message string is the text which would normally appear in the message bar at the bottom of a Cadcorp SIS Map Manager, Cadcorp SIS Map Editor, or Cadcorp SIS Map Modeller screen. Changing the `CommandMessage` will trigger the `MessageChange` event.

CommandPrompt

sets and gets the current Cadcorp SIS Control command prompt string

Syntax `Sis.CommandPrompt = String`

Notes This value can also be set and queried using the `_ArgPrompt$` system variable.

Display

controls the display type of the Cadcorp SIS Control. There are two types of display: Map (the window is a 2D view), and 3D (the window is a 3D OpenGL view). Some methods and commands are valid only in one type of display.

The `Display` property can be set only at design-time, so if your application is to display both 2D and 3D views, you must to place two Cadcorp SIS Controls on the form, and if necessary switch between them using the `Visible` property.

Level

sets and gets the current Cadcorp SIS Control licence level. The Level property controls access to methods and commands.

Syntax `Sis.Level = Integer`

Settings one of:

Constant	Value	Description
`SIS_LEVEL_UNLICENSED`	0	only a small subset of methods and commands are available
`SIS_LEVEL_MANAGER`	1	all the methods and commands of Cadcorp SIS Map Manager are available
`SIS_LEVEL_MODELLER`	2	all the methods and commands are available
`SIS_LEVEL_VIEWER`	3	a subset of the methods and commands of the Manager level are available

For more information about licensing, and the `Level` and the `CheckNetworkDongle` properties, ➲page 33, **Cadcorp SIS Control licensing**.

ShowWaitCursor

shows the wait cursor when the mouse pointer is over the control. The appearance of the cursor will be as set in the Windows Control panel (Mouse/Pointers, Busy pointer).

Syntax `Sis.ShowWaitCursor = Boolean`

Settings either:

True the wait cursor will be displayed

False the pointer cursor will be displayed

Swd

sets or returns the SWD serial number of the Cadcorp SIS Control

Syntax `Sis.swd = Integer`

Example `SisDetail.Swd = SisMain.Swd`

Notes Every Cadcorp SIS Control contains an SWD, which is essentially a list of overlays and a view extent. Each SWD has a unique serial number. Setting the `Swd` property of one Cadcorp SIS Control to the value of the SWD property of another Cadcorp SIS Control will create two views of the same set of overlays. Manipulating the overlays in either Cadcorp SIS Control will then affect the overlays in the other. This is analogous to having two child windows in a Cadcorp SIS application, which are views of the same SWD file. The serial number of an SWD is constant through a single program execution, but should not be stored for use between executions.

Setting the `Swd` property to –1 will make the Cadcorp SIS Control create and reference a new, empty SWD.

➔ TrackMouse

Syntax `Sis.TrackMouse = Boolean`

Settings either:

True	the `MouseTrack` event will be triggered every time the location of the mouse pointer changes, ie as the user moves the mouse
False	the `MouseTrack` event will be disabled

For more details of the `MouseTrack` event, ➲page 25, **Cadcorp SIS Control events**.

■ Cadcorp SIS Control commands

Whenever a menu option is selected in a Cadcorp SIS application, a *command* is invoked. Each of these commands has a name, which users are usually unaware of. The Program Window displays these commands when they are selected, enabling Cadcorp SIS Control programmers to find the name and use it when writing applications.

All system command names begin with the letters `ACom`.

■ Adding custom commands

Cadcorp SIS Control applications have access only the local menu, because no main menu exists. You can add local commands using `AddCommand`:

```
Sis.AddCommand "View area Details", "Display area item details", _
 "Area", 1, 1, "", ""
```

Whenever the user selects a single area item, the local menu now contains the **View Area Details** command. When the user selects this command, the `AppCommand` Cadcorp SIS Control Event will be called, with the menu string (the first argument to `AddCommand`) as the *`comname`* argument. The application can then respond to this event. In this case, it displays a dialog containing information about the selected area item.

■ Running system commands

Although the Cadcorp SIS API contains many functions to perform operations in the Cadcorp SIS Control, it is sometimes easier to invoke one of the system commands than to write application code to produce the same result. Any of the commands available to the user can also be invoked, although some are obviously more useful than others.

In the Cadcorp SIS Control, both one-shot and callback commands (➲Chapter 2: **"Customising with GisLink", Running system commands**, page 9) are run using the `DoCommand` method.

```
' Redraw the view by command.
Sis.DoCommand "AComRedraw"
' Start drawing a line.
Sis.DoCommand "AComLineEx"
```

The one-shot redraw command will be started and completed in the duration of the `DoCommand` method. However, the callback command exists beyond the duration of the call to the `DoCommand` method, in this case until the user presses Enter or Ctrl-Enter to complete the line, or Escape to quit the command. The progress of callback commands is monitored using events.

The `DoCommand` method can be called with either a one-shot or a callback command, and will choose the appropriate action.

■ Cadcorp SIS Control events

An ActiveX Control can specify *events*. For example, a button has a Click event. Events notify the ActiveX Control Container when something important happens in the ActiveX Control, typically when the user interacts with the ActiveX Control.

The Cadcorp SIS Control contains the following events:

```
AppCommand
CommandAction
DatasetItemEdit
LicenceError
MessageChange
MouseTrack
PromptChange
ScaleChange
SelectionChange
Snap
```

Cadcorp SIS Control events are not sent to the ActiveX Control Container when they occur as a result of actions taken by the ActiveX Control Container – when the Cadcorp SIS Control is inside one of its methods. Therefore, it may be necessary to act after calling one of the Cadcorp SIS Control methods, as well as reacting to an event. For example, the `ScaleChange` event will be called if the user zooms using the keyboard or mouse-wheel, but not if the `ZoomView` method is called.

The `LicenceError` event is an exception to this rule, and will be called *immediately* any licence error occurs. Do not call more methods from within the `LicenceError` event, to avoid possible recursion.

❢ AppCommand

called whenever an application command added by `AddCommand` is selected

Declaration

```
Private Sub Sis.AppCommand (ByVal comname As String)
End Sub
```

Argument *comname* STRING

the command name specified in `AddCommand`

CommandAction

called when system command actions take place. This is similar to GisLink triggers.

Declaration

```
Private Sub Sis.CommandAction (ByVal comname As String, ByVal _
  Action As String)
End Sub
```

Arguments

comname STRING

the command class name

action STRING

the command action, one of the following:

`End`	the current callback command has ended
`Failed`	the oneshot command failed
`KeyBack`	the Backspace key has been pressed
`KeyEnter`	the Enter key has been pressed
`KeyTab`	the Tab key has been pressed
`Succeeded`	the oneshot command succeeded

DatasetItemEdit

called when a dataset item is edited

Declaration

```
Private Sub Sis.DatasetItemEdit(ByVal dataset As String, _
   ByVal edit As Integer, ByVal nItems As Long)
End Sub
```

Arguments

dataset STRING

the name of the dataset containing the edited items

edit SHORT INTEGER

the edit action. One of the following:

0 items have been added

1 items have been removed

2 items have been swapped

nItems LONG INTEGER

always 0 in this release

LicenceError

called whenever a runtime licensing error occurs

Declaration

```
Private Sub Sis.LicenceError()
End Sub
```

MessageChange

called whenever the system message changes

Declaration

```
Private Sub Sis.MessageChange()
   lblMessage.Caption = Sis.CommandMessage
End Sub
```

MouseTrack

called whenever the mouse moves in the Cadcorp SIS Control, if the TrackMouse property is set to True. This event can be used in conjuction with the `GetCoordString` method to mimic the behaviour of the Position Bar in Cadcorp SIS applications.

Declaration

```
Private Sub Sis.MouseTrack(ByVal x As Double, ByVal y As Double, _
    ByVal z As Double)
  lblMouseTrack.Caption = Str(x) & ”,” & Str(y) & ”,” & Str(z)
End Sub
```

Arguments

x, y, z DOUBLE

the mouse position

PromptChange

called whenever the system prompt changes

Declaration

```
Private Sub Sis.PromptChange()
  lblPrompt.Caption = Sis.CommandPrompt
End Sub
```

ScaleChange

called whenever the display scale changes, such as if the view is zoomed or the Cadcorp SIS Control window changes size

Declaration

```
Private Sub Sis.ScaleChange(ByVal displayScale As Double)
  lblScale.Caption = Str(displayScale#)
End Sub
```

Arguments

displayScale DOUBLE

the display scale

SelectionChange

called whenever the selection changes

Declaration

```
Private Sub Sis.SelectionChange (ByVal nEditable As Long, ByVal nHittable _
  As Long, ByVal commonClass As String)
   MsgBox "There are "& Str(nEditable) & " Editable Items Selected"
   MsgBox "There are “ & Str(nHittable) & " Hittable Items Selected"
   MsgBox "The most common class is " & commClass
End Sub
```

Arguments

nEditable LONG INTEGER

the number of editable items currently selected

nHittable LONG INTEGER

the number of hittable items currently selected

commonClass STRING

the lowest common denominator class of those selected

Snap

called whenever the user snaps

Declaration

```
Private Sub Sis.Snap(ByVal x As Double, ByVal y As Double, ByVal z As Double)
  lblSnap.Caption = "Position snapped " & Str(x) & "," & Str(y) & Str(z)
End Sub
```

Arguments `x, y, z` DOUBLE
the snap position

The special system named list `snapped` can be used in this event to query any snapped item.

■ Position input

◆ Procedural and event-based approaches

Cadcorp SIS Control applications often require the user to input positions using the mouse as part of an application command or action. This may be part of a set of a procedural actions. For example, your application prompts the user for text, then gets a position from the user, then creates a text item using the results.

This procedural approach conflicts with the Windows Graphical User Interface (GUI) model of dialog input, which allows users to interact with many controls, in any order, before they click the OK or Apply buttons. As a result, the Cadcorp SIS Control uses events to tell the application when the user has interacted with it. This conflicts with the procedural approach because the events happen asynchronously.

To mimic a procedural approach, you must split up the procedure into two parts:

- before a position or positions are required
- after the required position(s) have been supplied

The first part can be started by a button press on the application dialog and would, in the example above, prompt the user for text. After the text is successfully entered, the application would set a flag that signified that a position was being requested. The application would then check for the flag in the Snap event, and, if the flag were set, would perform the second part of the procedural command, eg create a Text item using the previously entered text, before resetting the flag.

This approach does have drawbacks, however: global flags are required to monitor the application state, which other user actions can affect, and global variables are required, to remember the text, for example.

You can avoid some of these problems. For example, eliminate a global variable by getting the position first and *then* prompting the user. Or you could show another dialog to collect the details. Or you could disable the rest of the dialog until a position is entered. However, the dichotomy between the procedural and event-based approaches remains.

◆ GetPos/GetPosEx

GisLink, which has the advantage of using Cadcorp SIS as an application, as opposed to the Cadcorp SIS Control which exists within a foreign environment, has the `GetPos` and `GetPosEx` methods to overcome the procedural versus event problem. The same techniques can be used with the Cadcorp SIS Control using Visual Basic and Visual C++, but are unsupported because other OLE Container applications may not have an equivalent, or their effects may be platform-dependent.

◆ User position input with Microsoft Visual Basic

The following code is a modified `GetPosEx` method, together with its companion `GetArg` method:

```
Function GetPosEx (ByRef SisPos As Sis, x As Double, y As Double, _
  z As Double) As Long
 Dim lTypeArg As Long
 Dim sPos As String
 GetPosEx = SIS_ARG_ESCAPE
 Do
  lTypeArg = GetArg (SisPos)
  GetPosEx = lTypeArg
  Select Case lTypeArg
   Case SIS_ARG_ESCAPE, SIS_ARG_ENTER, SIS_ARG_BACKSPACE
    Exit Function
   Case SIS_ARG_POSITION
    sPos = SisPos.GetStr SIS_OT_SYSTEM, 0, "_ArgPos$")
    If sPos <> "" Then
     SisPos.SplitPos x, y, z, sPos
     Exit Function
    End If
   End Select
 Loop
End Function

Function GetArg (ByRef SisPos As Sis) As Long
' Get the number of arguments user has entered so far.
 Dim lArg as Long

 lArg = sisPos.GetInt (SIS_OT_SYSTEM, 0, "_NumArg&")

' Start argument collecting command in SIS Control.
' AComGetArg uses the _ArgPrompt$ system variable as its prompt.
' It increments the _NumArg& system variable by 1 to indicate that
' it has finished.
' It sets the _TypeArg& system variable to store the user action.
' It sets the _ArgPos$ variable if the action was a snap.
  sisPos.DoCommand "AComGetArg"
  sisPos.SetFocus

  Do
' Let user interact with the dialog. ending if the all forms are closed.
    If DoEvents() = 0 Then End

' See if user has done something yet.
    If sisPos.GetInt (SIS_OT_SYSTEM, 0, "_NumArg&") > lArg Then Exit Do
  Loop

' Find out what user did.
  GetArg = sisPos.GetInt (SIS_OT_SYSTEM, 0, "_TypeArg&")
End Function
```

◆ Visual Basic DoEvents function

The Visual Basic `DoEvents` function used in `GetArg` above allows the user to interact with other parts of the applications as well as the Cadcorp SIS control. A side effect of this is that the user could terminate the whole application while the `GetArg` method is still in its `Do` loop.

The `GetArg` method attempts to cope with this by exiting when the last form is closed, ie when the `DoEvents` method returns 0.

However, this approach will not work if the Cadcorp SIS Control is contained in a sub-form of the main Visual Basic application form. In this case, the sub-form/dialog *must* be prevented from closing while calling `GetArg`. Failure to do this could lead to a crash, because the Cadcorp SIS Control being queried can be deleted before `GetArg` has returned.

A reliable way to prevent this is to have a variable which is set to True at the start of the `GetArg` method, and False at the end, and to use this variable in the `QueryUnload` of the calling form to prevent the form being unloaded. For example:

```
Private Sub Form_QueryUnload (Cancel as Integer, UnloadMode As _
  Integer)
    If bInGetArg Then Cancel = 1
End Sub
```

◆ User position input with Microsoft Visual C++

Getting position input using Visual C++ is more complex than in Visual Basic because of the fundamental differences between a window and a modal dialog. Modal dialogs already have their own event loops that allow them to be used procedurally, and you must take care not to upset this.

◆ GetPos/GetPosEx in dialogs

The following code fragment has been adapted from the `CWnd::RunModalLoop` method, and may require changes in future versions of the Microsoft Foundation Classes:

```
int GetArg(CDialog *pParent,CSis &sisPos)
{
   int nArg=sisPos.GetInt( SIS_OT_SYSTEM,0,"_NumArg&");
   sisPos.DoCommand("AComGetArg");
   sisPos.SetFocus();
   MSG* pMsg=&AfxGetThread()->m_msgCur;
   for (;;)
   {
      if (!pParent->ContinueModal()) return -1;
      // Pump messages while available
      do
      {
         if (!pParent->ContinueModal()) return -1;
         // Pump message, but quit on WM_QUIT
         if (!AfxGetThread()->PumpMessage())
         {
            AfxPostQuitMessage(0);
            return -1;
         }
```

```
            if (!pParent->ContinueModal()) return -1;
            if (sisPos.GetInt(SIS_OT_SYSTEM,0,"_NumArg&")>nArg)
            {
              return sisPos.GetInt(SIS_OT_SYSTEM,0,"_TypeArg&");
            }
        } while (::PeekMessage(pMsg,NULL,NULL,NULL,PM_NOREMOVE));
    }
    return -1;
}
int GetPosEx(CDialog *pParent,CSis &sisPos,double &x,double &y,double &z)
{
    int action=SIS_ARG_ESCAPE;
    while (TRUE)
    {
        action=GetArg(pParent,sisPos);
        switch (action)
        {
            case SIS_ARG_ESCAPE:
            case SIS_ARG_ENTER:
            case SIS_ARG_BACKSPACE:
            {
              return action;
            }
            case SIS_ARG_POSITION:
            {
              CString
               strPosition=sisPos.GetStr(SIS_OT_SYSTEM,0,"_ArgPos$");
              if (!strPosition.IsEmpty())
              {
                sisPos.SplitPos(&x,&y,&z,strPosition);
                return action;
              }
            }
            case -1:
            {
              // Error return.
              return -1;
            }
        }
    }
    return action;
}
```

◆ GetPos/GetPosEx in other windows

Cadcorp SIS Controls that are used in a `CView`-derived class work similarly, but have special code for idle processing and responding to the user quitting the application while in the event loop:

```
int GetArg(CSis &sisPos)
{
    int nArg=sisPos.GetInt(SIS_OT_SYSTEM,0,"_NumArg&");
    sisPos.DoCommand("AComGetArg");
    sisPos.SetFocus();
    CWinThread *pThread=AfxGetThread();
    while (TRUE)
```

```
    {
      MSG msg;
      if (::PeekMessage(&msg,NULL,NULL,NULL,PM_NOREMOVE))
      {
        try
        {
          if (!pThread->PumpMessage())
          {
            AfxPostQuitMessage(0);
            return -1;
          }
        }
        catch (...)
        {
          TRACE0("Error: exception in GetArg - continuing.\n");
        }
      }
      else
      {
        pThread->OnIdle(-1);
      }
      if (::PeekMessage(&msg,NULL,WM_QUIT,WM_QUIT,PM_NOREMOVE)) return -1;
       try
       {
         if (sisPos.GetInt(SIS_OT_SYSTEM,0,"_NumArg&")>nArg) break;
       }
       catch (...)
       {
       return -1;
       }
}
return sisPos.GetInt (SIS_OT_SYSTEM,0,"_TypeArg&");
}
int GetPosEx(CSis &sisPos,double &x,double &y,double &z)
{
  int action=SIS_ARG_ESCAPE;
  while (TRUE)
  {
    action=GetArg(sisPos);
    switch (action)
    {
      case SIS_ARG_ESCAPE:
      case SIS_ARG_ENTER:
      case SIS_ARG_BACKSPACE:
      {
        return action;
      }
      case SIS_ARG_POSITION:
      {
        CString
         strPosition=sisPos.GetStr(SIS_OT_SYSTEM,0,"_ArgPos$");
        if (!strPosition.IsEmpty())
        {
          sisPos.SplitPos(&x,&y,&z,strPosition);
          return action;
        }
```

```
            }
            case -1:
            {
               // Error return.
               return -1;
            }
         }
      }
      return action;
   }
```

Both the Visual Basic and Visual C++ code fragments shown here can be used to get more than one position from the user.

■ Cadcorp SIS Control licensing

The Cadcorp SIS Control is licensed, like the Cadcorp SIS applications, using a hardware lock (dongle). The licensing covers both design time, when an application is being written, and run time, when the application is being run.

In addition, the Cadcorp SIS Control has four built-in licensing levels, controlled by the `Level` property. ➲Chapter 3: **"Cadcorp SIS Control", Cadcorp SIS Control properties**

The default level is `Unlicensed`, at which only a small number of system housekeeping methods and commands are available. The other levels enable more of the methods and commands. The `Unlicensed` level does not require that a licence be present, and is useful for starting an application before making a choice of required level.

The `Level` property cannot be set at design-time leaving the Cadcorp SIS Control unlicensed at startup. If the application is written using Visual Basic, the Cadcorp SIS Control will have been created before the `Form_Load` event, so the `Level` property can be set here. The property can be tested immediately after being set; any discrepancy from the intended value indicates a licence failure. In addition, the `LicenceError` event will be called.

At run-time, the licence is tested from time to time at the start of a method, and whenever the `Level` property is changed. This prevents swamping the licence, and potentially the network, if a network licence is in use. If an error occurs when changing the `Level` property, the existing level will be maintained, but the `LicenceError` event will be called.

◆ Network licensing

The Cadcorp SIS Control supports network licences and local (hardware) locks. A network licence is a hardware lock plugged into a remote machine on the network and controlled by a server program on the remote machine.

Like the Cadcorp SIS applications, it is possible to prevent the Cadcorp SIS Control looking for a network licence using the `_bCheckNetworkDongle&` system option. There is, however, a bootstrapping problem, both at design-time and run-time, because the default value of the system option is to not look for a network hardware lock. The option is set using the `SetInt` method, or the `CheckNetworkDongle` property must be used. The `CheckNetworkDongle` property maps directly on to the system option. If, at run-time, the application wants to use a network licence, set the `CheckNetworkDongle`

property before setting the Level property to Viewer, Manager, or Modeller, which will cause a licence test.

◆ Design time considerations

At design time, the Cadcorp SIS Control will be loaded only when necessary and will be automatically unloaded when not in use, for example, if no form containing the Cadcorp SIS Control is open in Visual Basic. The Cadcorp SIS Control can therefore be loaded and unloaded many times during a design session. If the design time licensing is using a network licence, this can have an impact on performance because of the time taken to find the network licence server on the network. We recommend that design time licences be used on local machines wherever possible.

In addition, the design time licence check happens before the Cadcorp SIS Control is created, and is thus unable to use the `CheckNetworkDongle` property. If, at design-time, a local design time licence is not found, the design time user will be offered the chance to look for a network licence. If the option is taken, and a network design time licence is found, the Cadcorp SIS Control will remember to look for a network licence in future by storing a value in the Windows Registry. This value will be unset only if a future check for a design time network licence fails.

◆ Multiple Cadcorp SIS Controls in an application

You can have more than one Cadcorp SIS Control in an application. Multiple Cadcorp SIS Controls will share the same licence, and, by implication, the same Level. In addition, they will take only one ‘user’ from a network licence. Take care when changing the `Level` property of any one Cadcorp SIS Control, because this will affect all other Cadcorp SIS Controls in the application.

Chapter 4

Cadcorp SIS OLE DB provider

■ Introduction to Object Linking and Embedding for Databases 35
■ Cadcorp SIS OLE DB Provider 35

■ Introduction to Object Linking and Embedding for Databases

Businesses are increasingly finding that they have to build solutions that require information from a variety of data stores. The fact that there are many ways in which they can be accessed creates barriers to developing applications that can bridge both the new and the old technologies.

OLE DB is a low-level programming interface that effectively serves as the foundation of Microsoft's Universal Data Access Strategy. ODBC was designed to access only relational databases, but OLE DB allows applications to access any type of data source, including relational databases.

For more information, visit www.microsoft.com/data/oledb/.

■ Cadcorp SIS OLE DB Provider

The Cadcorp SIS OLE DB Provider is supported only on Microsoft Windows NT4, Windows 2000, and Windows XP, and it requires a Cadcorp SIS dongle. The data returned by the Cadcorp SIS OLE DB Provider is *read only*.

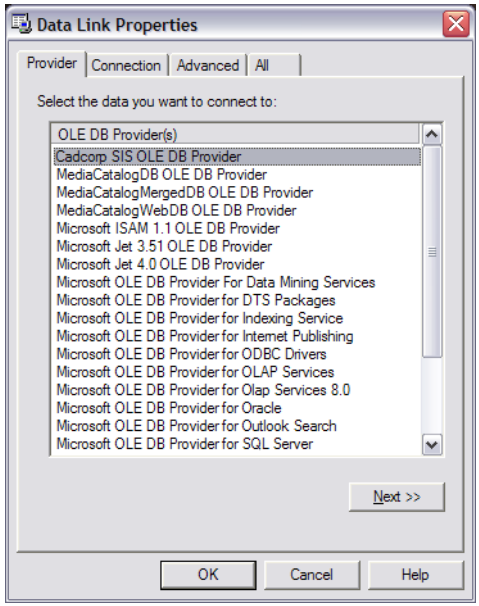
The Cadcorp SIS OLE DB Provider can be used in any application which supports OLE DB data sources, such as:

- Microsoft Excel
- Crystal Reports
- Microsoft Visual Basic V6.0

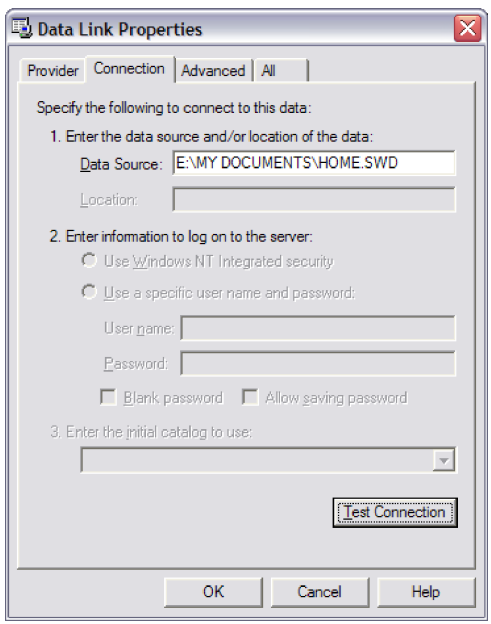
This chapter discusses the use of the Cadcorp SIS OLE DB Provider via Microsoft Visual Basic V6.0 only. A basic understanding of Microsoft ActiveX Data Objects (ADO) technologies is assumed.

To use the Cadcorp SIS OLE DB provider, you must have an ADO connection object that is opened using a connection string. The easiest way to create this connection string is by using a UDL (Universal Data Link) file.

To create a UDL file, create an empty file with the file extension *.udl. When this file is opened in Windows, you are presented with a dialog for generating a connection:



From the Provider tab, select Cadcorp SIS OLE DB Provider, then click on the Next button, and complete the following dialog.



When you have entered all the information, click OK to close the dialog, and then open the file using a text editor such as Notepad. The file contains the connection string.

```
[oledb]
 ; Everything after this line is an OLE DB initstring
 Provider=Cadcorp SIS OLE DB Provider;Data Source=E:\MY DOCUMENTS\HOME.SWD
```

The third line is the connection string required by the ADO connection object.

The following code shows how this connection string is used to open the ADO Connection.

```
 Dim oConnection As ADODB.Connection
 Set oConnection = New ADODB.Connection
 With oConnection
  .Provider = "Cadcorp SIS OLE DB Provider"
  .Properties("Data Source") = "E:\MY DOCUMENTS\HOME.SWD"
  .CursorLocation = adUseClient
  .Open
  If .State = adStateOpen Then
   MsgBox "Successfully connected to: " & .Properties("Data Source")
  End If
 End With
```

The cursor location is required if you want to be able to use recordsets via code. With an open connection, you can get a list of all available overlays using the following code:

```
Dim oRS As ADODB.Recordset

 Set oRS = oConnection.OpenSchema(adSchemaTables)

 Do While Not oRS.EOF
  Debug.Print oRS.Fields("TABLE_NAME").Value
  oRS.MoveNext
 Loop

 oRS.Close
 Set oRS = Nothing
```

When you have identified the overlay you want to open, use the following code to load the overlay data into an ADO recordset:

```
Dim oRS As ADODB.Recordset
 Dim oField As ADODB.Field

 Set oRS = New ADODB.Recordset
 Set oRS = oConnection.Execute("Areas", , adCmdTable)

 Debug.Print "Records: " & oRS.RecordCount
 Do While Not oRS.EOF And Not oRS.BOF
  For Each oField In oRS.Fields
   If TypeName(oField.Value) = "Unknown" Then
```

```
    Debug.Print oField.Name & vbTab & "Unknown"
   Else
    Debug.Print oField.Name & vbTab & oField.Value
   End If
  Next oField
  oRS.MoveNext
 Loop

 oRS.Close
 Set oRS = Nothing
```

Make sure that when you are finished with the connection, you close it:

```
oConnection.Close
```

A sample Microsoft Visual Basic application showing how the provider can be used is included in the Bonus directory on your Cadcorp SIS installation CD.

Chapter 5

Cadcorp SIS Active Server Component

- Introduction 39
- What the ASC does 39
- Requirements 40
- Installation: Windows NT 41
- Installation: Windows XP and 2000 41
- Overview 45
- ASC elements 47
- Building a Cadcorp ASC application 53
- Establish a virtual directory 53
- Generate the files 53
- Adding tools to MAP.ASP 57
- Adding a query function 60
- OpenGIS Web Servers 65

■ Introduction

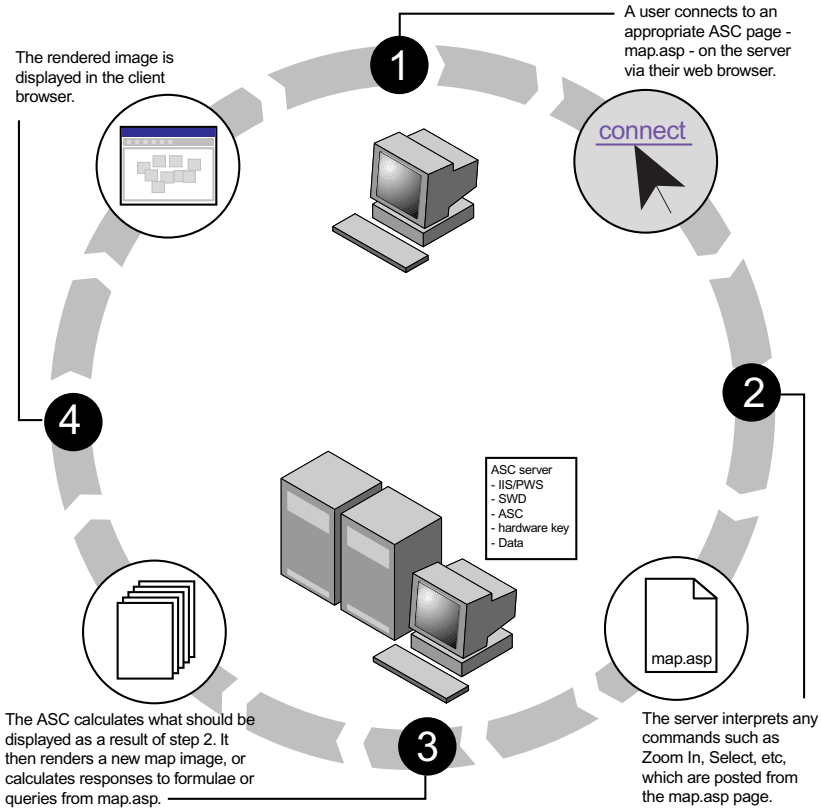
The Cadcorp SIS Active Server Component, or ASC, is designed to allow embed spatial data and GIS functionality to be used within standard internet browsers. It is a 32-bit COM Server side application, which allows a programmer to create tools, usable in Internet Browsers, to perform GIS functions such as Zoom In, Select a Record, and Show the Nearest. The ASC is addressed using the methods described in this manual.

The Cadcorp SIS Active Server Component is a part of the Internet Development Kit, which should be installed only on the developer's computer (the development environment). This is for building ASC applications, which are later transferred to an ASC server, which has a separate ASC licence.

■ What the ASC does

The Cadcorp SIS Active Server Component (ASC) allows graphic and attribute data held in a saved window definition (SWD) to be displayed in an internet browser. It adopts a 'thin' client approach, so that the bulk of processing is performed on the server.

The process of serving an SWD via the ASC is shown in the following diagram:



■ Requirements

You need the following to install, run, and begin developing with the ASC:

- a Cadcorp SIS desktop product capable of generating an SWD with data in it (Cadcorp SIS Map Modeller)
- an ASC licence code and hardware key (found in the Internet Developer Kit)
- a server computer running Microsoft Personal Web Server (Windows 2000) or Microsoft Internet Information Server (Windows NT, XP, or 2000). The server must have the hardware key attached to its parallel port and the user must have administrator privileges, or equivalent.
- a client computer which can access the server on a Local Area Network (LAN) or the World Wide Web (WWW), with Netscape Navigator or Internet Explorer 4.0 or above installed on it

- an HTML editor such as Microsoft FrontPage or Macromedia DreamWeaver. We do not recommend using a text editor for large sites, where coding can become complex.

■ Installation: Windows NT

The first step is to install the ASC on the server.

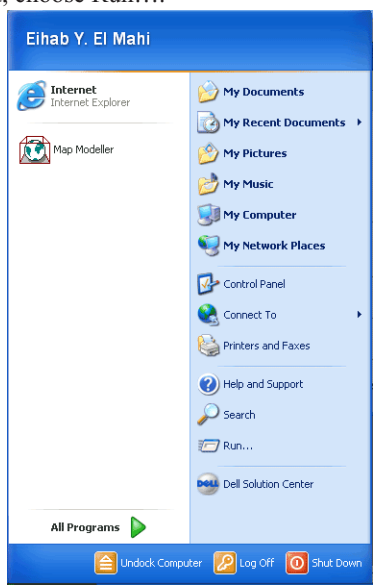
1. Insert the Cadcorp CD-ROM into the appropriate drive on the server.
2. From the list of options on the Cadcorp introduction screen, select the ASC option and follow the Wizard to the end making sure to enter the ASC licence number.
3. Close down and reboot the server.
4. Ensure that the ASC hardware key is connected to the Server's parallel port.

Once the computer has been rebooted, the SisASC.dll needs to be unregistered.

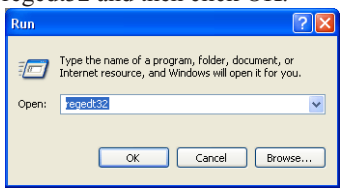
1. Select Run from the Windows Taskbar, then type Regsvr32, browse to the location of the SisASC.dll and then type /u. This unregisters any existing ASC installations. For example:
   `Regsvr32 "C:\Program Files\Cadcorp SIS 5.0\SisAsc.dll" /u`
2. Re-register the SisASC.dll. Repeat the above process without the `/u` at the end of the string. A message will appear informing you that the registration has been successful.

■ Installation: Windows XP and 2000

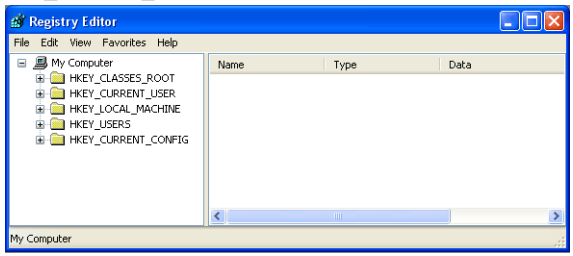
1. From the Start menu, choose Run….



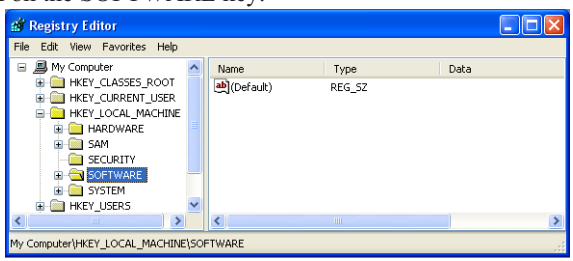
2 When prompted, type regedt32 and then click OK.



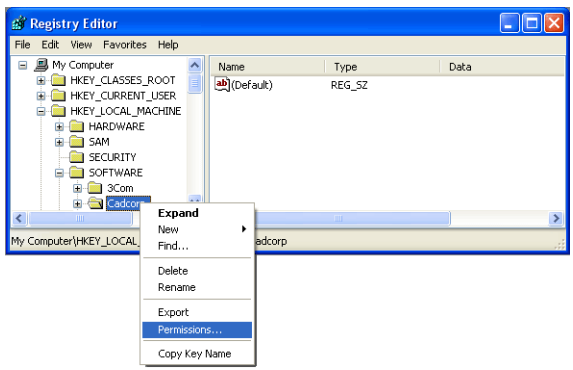
3 Expand HKEY_LOCAL_MACHINE.



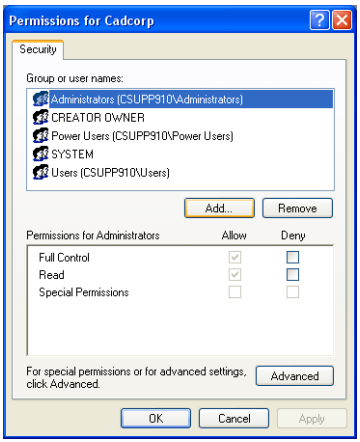
4 Right-click on the SOFTWARE key.



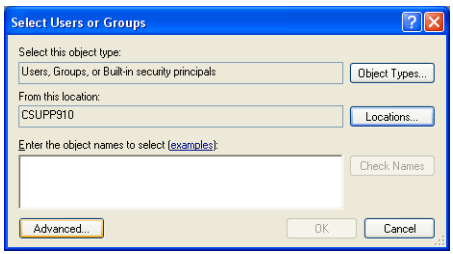
5 Select Permissions….



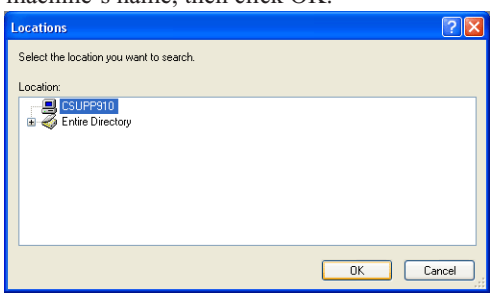
6 Click Add….



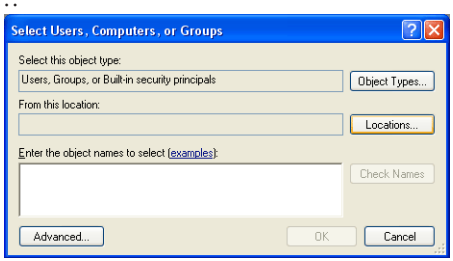
7 Click Locations…



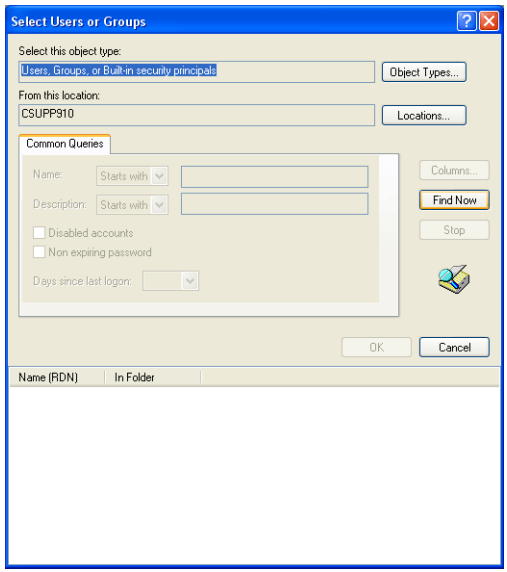
8 Highlight your machine's name, then click OK.



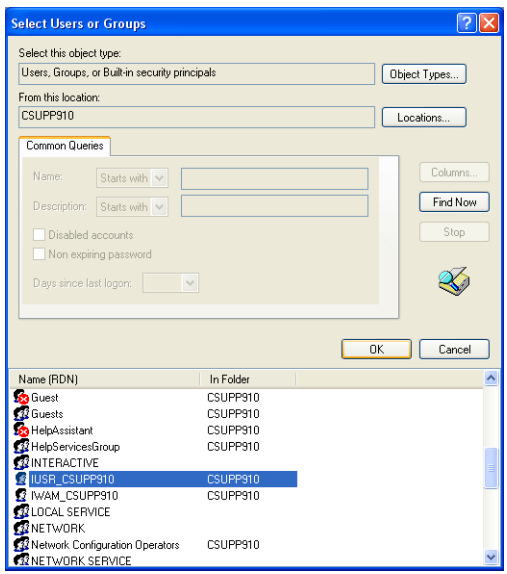
9 Click Advanced…



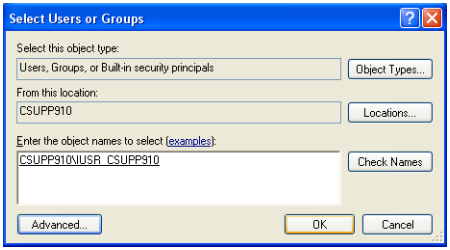
10 Click Find Now.



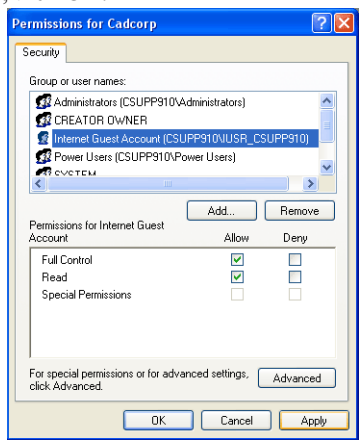
11 Select IUSR_*MachineName* and then click OK.



12 Click OK.



13 Highlight the IUSR (Internet Guest Account) and allow Full Control to this account. Click Apply, then OK.



■ Overview

Use the ASC by writing code within Active Server Pages, which call ASC methods. An Active Server Page (ASP) is an HTML page that includes one or more scripts (small, embedded programs) that are processed on a Microsoft Web server before the page is sent to the user. An ASP is somewhat similar to a server-side include or a common gateway interface (CGI) application, in that all involve programs that run on the server, usually tailoring a page for the user. Typically, the script in the web page at the server uses input received as the result of the user's request for the page to access data from a database and then builds or customises the page on the fly before sending it to the requestor.

ASP is a feature of the Microsoft Internet Information Server (IIS), but, since the server-side script is just building a regular HTML page, it can be delivered to almost any browser. You can create an ASP file by including a script written in VBScript or JScript in an HTML file, or by using ActiveX Data Objects (ADO) program statements in the HTML file. You name the HTML file with the *.asp file suffix. Microsoft recommends the use of the server-side ASP rather than a client-side script, where there is actually a choice, because the server-side script will result in an easily displayable HTML page. Client-side scripts (for example, with JavaScript) may not work as intended on older browsers (source www.Whatis.com).

By calling Cadcorp SIS methods, the programmer gains the ability to carry out a large range of functions from creating a simple map image to allowing the user to pan, zoom and query spatial data and related databases through a web browser.

The ASC interprets these methods and code and renders an image (normally a JPEG) which is then sent to the client browser.

The following code shows how to use a method in the ASC. In this case the SIS method `ZoomView` is being called when a button called ZOOMIN on a Form on an ASP is clicked. This will zoom in on the map image.

```
If Request.Form("ZOOMIN")<>"" Then
    oSis.ZoomView 0.5
end if
```

By manipulating methods, you can develop complex and powerful web-enabled applications. Next, we explain the other elements you need to begin writing your own applications.

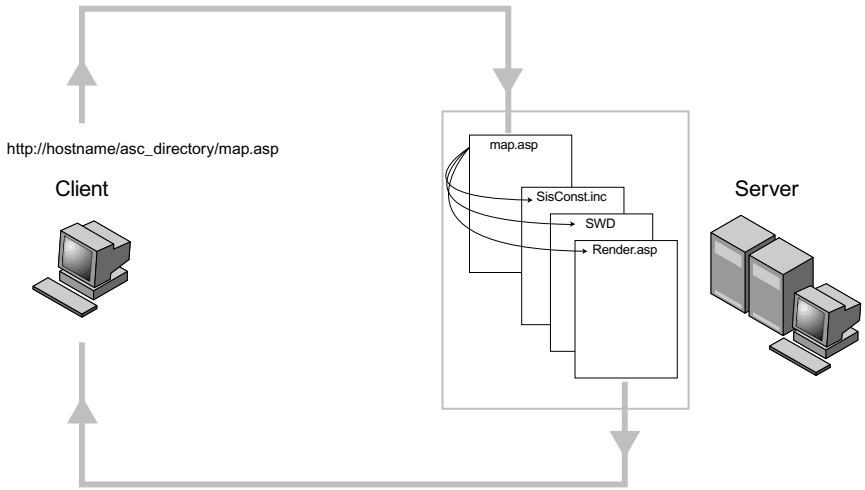
■ ASC elements

You need the following five files to implement the ASC:

Created with Cadcorp software	Created with an HTML editor
saved window definition (SWD) a collection of data, and information describing it, such as line colour, thematic mapping, and symbols	`Global.asa` session variables and subroutines

`SisConst.inc`
a constant file generated from the Program Window's local menu
	`Map.asp`
	the page which the user interacts with the ASC via a FORM (this does not have to be called Map.asp)
	`Render.asp`
	the page which Map.asp calls in order to send a new image to the client browser

The files should all be in a common folder, which will be treated as a Virtual Directory by a Personal Web Server or Internet Information Server:



◆ 1: a saved window definition (SWD)

The aim of the ASC is to allow the user to serve the contents of a specific SWD file. It is this file which the ASC methods act on. For example, if a piece of code is written for the ASC which allows the user to make a layer visible or invisible by clicking on a button on a browser, it will refer to the layers in the SWD and change its properties accordingly. Remember, the SWD points to the location of the data, it does not store the data internally, unless it is an internal file. This is important to remember, because data might not follow the same directory structure on the web server.

◆ 2: Global.asa

This file can be used to handle application-wide events. For example, you can use the Global.asa file to automatically execute one subroutine whenever a visitor arrives at your web site, and another subroutine whenever a visitor leaves. It is good practice to use one Global.asa file per application.

The Global.asa file can contain subroutines that handle the following types of events:

`Session_OnStart`	This event occurs whenever a new visitor requests the first page contained within the Active Server Pages application.
`Session_OnEnd`	This event occurs when a user session ends. By default, a user session ends after the user does not request a page for more than 20 minutes.
`Application_OnStart`	This event occurs when a user requests a page from an Active Server Page application for the first time. Typically, the Application_OnStart event is triggered after your web server is restarted or after the Global.asa file is modified.
`Application_OnEnd`	This event is triggered when your web server shuts down. It is the last event called after all of the Session_OnEnd events are triggered.

The Global.asa file for Cadcorp SIS ASC is as follows.

Listing 5.1 global.asa file

```
<body bgcolor="#FFFFFF">
<font face="Arial"><b> </b></font>
<SCRIPT LANGUAGE=VBScript RUNAT=Server>
 Sub Application_OnStart
 End Sub
</SCRIPT>
<SCRIPT LANGUAGE=VBScript RUNAT=Server>
 Sub Application_OnEnd
 End Sub
</SCRIPT>
<SCRIPT LANGUAGE=VBScript RUNAT=Server>
Sub Session_OnStart
 ' Store the image size.
   Session("xSize")=200
   Session("ySize")=200
' Make sure that new users start on the correct page of
' GettingStarted (map.asp).
   startPage="/map.asp"
   currentPage=Request.ServerVariables("SCRIPT_NAME")
   If strcomp(currentPage,startPage,1) Then
      Response.Redirect("/cadcorpSIS" & startPage &
        Request.QueryString())
   End If
 End Sub
</SCRIPT>

<SCRIPT LANGUAGE=VBScript RUNAT=Server>
 Sub Session_OnEnd
```

```
 End Sub
</SCRIPT>
' End of example
```

In this example, the image size of the rendered image is set as 200 pixels square. The image can be set to a maximum of 1024 pixels by 1024 pixels:

```
' Store the image size.
  Session("xSize")=200
  Session("ySize")=200
```

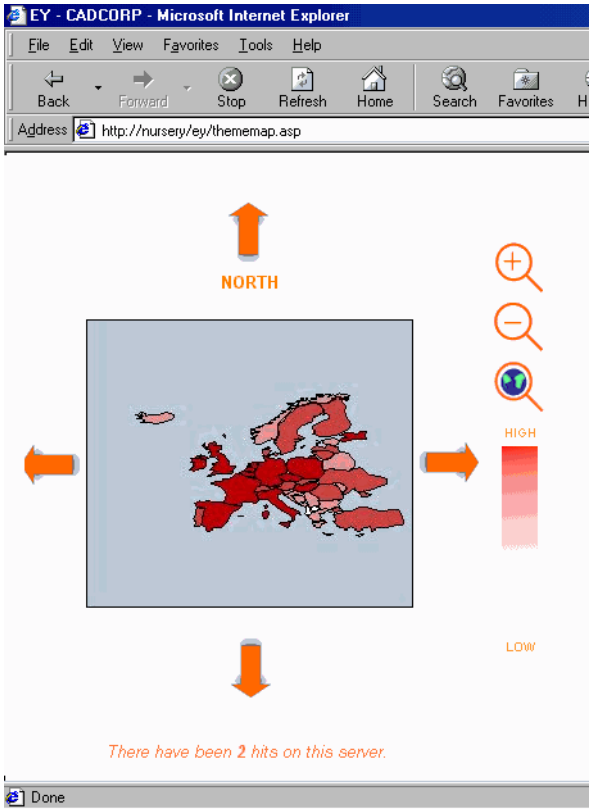
This number should correspond to the size specified in the render.asp file.

The start page is set to be map.asp (this is where the ASC will return the rendered image to) which is in a folder on the web server called cadcorpSIS.

◆ 3: Map.asp

The ASC needs a place for user interaction to occur. The ASC image appears on this page. Buttons and tools which are designed for use with the ASC will also appear on this page. Such tools may be a Zoom In button, an Identify Image or Query dialog. The page does not necessarily need to be called map.asp. In the example given in the next chapter, and illustrated on the following page, a page called thememap.asp has been constructed.

On the page are Pan tools, Zoom In and Zoom Out tools and a Zoom to Full Extent tool. In the centre of the page is a JPEG image, which is passed from the SWD via the ASC.



The simplest page will have just the rendered image on it. (Remember, the ASC is a toolkit.) Earlier we mentioned that the ASC serves an image which is generated from data in an SWD.

The following code (➲page 50, **Listing 5.2 map.asp file**) illustrates how an SWD named ptalbot.swd would be used on a page named map.asp.

Listing 5.2 map.asp file

```
<html>
<head>
<title>CADCORP</title>
</head>
<!-- #include file = "SisConst.inc" -->

<%
' VB Script
' Initialise the ASC, load the SWD and pass to a page called
' render.asp
 If Not IsObject(Session("Sis")) Then
 ' Create the SIS Active Server Component
 Set oSis=Server.CreateObject("SisASC.SisASC.6")
```

```
 oSis.LoadSwd Server.MapPath("ptalbot.swd")
 Set Session("Sis")=oSis
End If

Set oSis= Session("Sis")
%>

<body>
<FORM METHOD="POST" ACTION="map.asp" name="mapform">
<input type="image" name="Map" src="Render.asp" width="200"
 height="200" align="middle">
</FORM>
<!-- Whole form -->
</body>
</html>
```

This code consists of two major sections.

The first is the part between the script tags <% and %>. This is VB Script which initialises the ASC code. The script sits between the header and the body, as with any other ASP. The line:

`Set oSis=Server.CreateObject("SisASC.SisASC.6"),`

sets the version of the ASC to Cadcorp SIS version 5. This is necessary to configure the licence number and hardware key. In this case, the SWD ptalbot is used. This is set on the line:

`oSis.LoadSwd Server.MapPath("ptalbot.swd")`

The session variable `oSIS` is also set:

`Set Session("Sis")=oSIS`

For any ASC method to work, it must be preceded by this variable. For example, in GIS Link or in the Active X component, methods would be labelled, `GISZoomExtent`. By setting the variable as value `oSIS`, the same method would be written as:

`oSIS.ZoomExtent`

The second part of the code is the HTML body, which contains a form, between the `<FORM>` and `</FORM>` tags. The form is where the image rendered by the Cadcorp SIS ASC is actually located. It is also the means by which the user communicates with the ASC to get information and render a new image from the SWD.

The form is a way for Active Server Pages to communicate from the client (the user's Internet Browser) to the server. In this case, any communications from the form called `mapform` are POSTed to the server. The response or ACTION occurs on a page called map.asp. This can be set to be any ASP, but for the page which contains our map image, it should generally be set to itself.

The last part of the form is where our map image is displayed. As you can see, the image is set to be 200 x 200 pixels in size, corresponding to the values we set in the script. The image is called Map and is drawn from a page named render.asp, dicussed next.

◆ 4: Render.asp

The render.asp file contains a single line of code:

```
<% Session("Sis").Render 200,200,"image/jpeg"%>
```

This code takes the session variable `SIS` and then renders an image 200 pixels square, which corresponds to the sizes which were specified in the map.asp file above. The image is set to the JPEG format.

◆ 5: SisConst.Inc

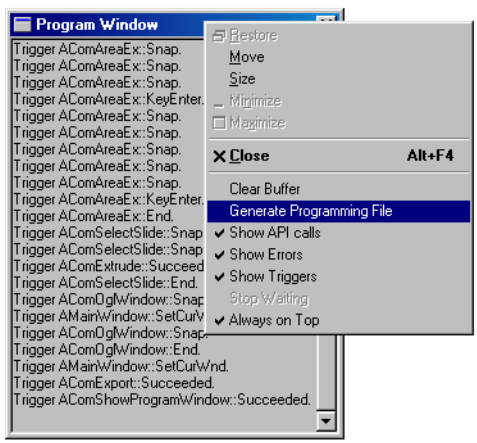
This include file is required by the ASC, because it contains the information required by the server to process the ASC methods. In the map.asp page you may have seen that the file is actually referred to in the script:

```
<html>
<head>
<title>CADCORP</title>
</head>
<!-- #include file = "SisConst.inc" -->
<%
' VB Script
' Initialise the ASC, load the SWD and pass to a page called
' render.asp etc
```

The contents of the file allow constants to be stored so that the methods within scripts written by programmers can be interpreted by the ASC. A sample of the content is shown below.

```
' Formula calculations.
Const SIS_CALCULATE_COUNT = 0
Const SIS_CALCULATE_SUM = 1
Const SIS_CALCULATE_AVERAGE = 2
' Co-ordinate types.
Const SIS_COORD_CARTESIAN = 0
Const SIS_COORD_SPHERICAL = 1
```

The file is generated from Cadcorp SIS desktop products by clicking the local menu of the title bar of the Program Window.



■ Building a Cadcorp ASC application

This section describes how to build a simple Cadcorp SIS ASC application. The application will allow the end user to Zoom In, Zoom Out, Pan, Switch overlays on and off and find attributes in an overlay table.

The main stages of the process are as follows:

1. Establish a virtual directory on the server.
2. Create the files:
   the SWD
   SisConst.Inc
   Render.asp
   Map.asp (including ASC tools)

You will need the Microsoft Personal Web Server (PWS) or the Internet Information Server (IIS) already installed on the server.

■ Establish a virtual directory

➲Chapter 6: **"Cadcorp SIS Map Server", Establish a virtual directory**, page 71

■ Generate the files

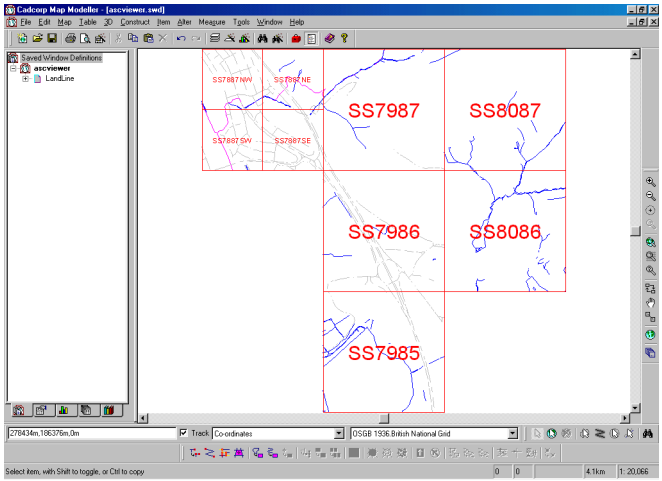
◆ Saved window definition

Saved window definition (SWD) files are created in Cadcorp SIS desktop products, ie Cadcorp SIS Map Modeller, Cadcorp SIS Map Editor, or Cadcorp SIS Map Manager. An SWD is a user-defined view of a series of data files known as overlays. The SWD is saved by choosing Save from the File menu in Map Modeller. The SWD will store details relating to the location, appearance, and properties of the overlays. By default, the SWD which is used by the ASC will appear as it did at the last save.

For our sample ASC application, we will load in `Land-line.SWD` into Cadcorp SIS Map Modeller. This SWD is provided in the sample data supplied with Cadcorp SIS. The data is 1:1250 Ordnance Survey data of the Port Talbot area in South Wales.

Load the SWD by choosing **File>Open** and browsing to the sample data directory (make sure you have installed it from the Cadcorp SIS CD-ROM). Select `Landline.SWD`.

You will be presented with a map window which appears as follows:

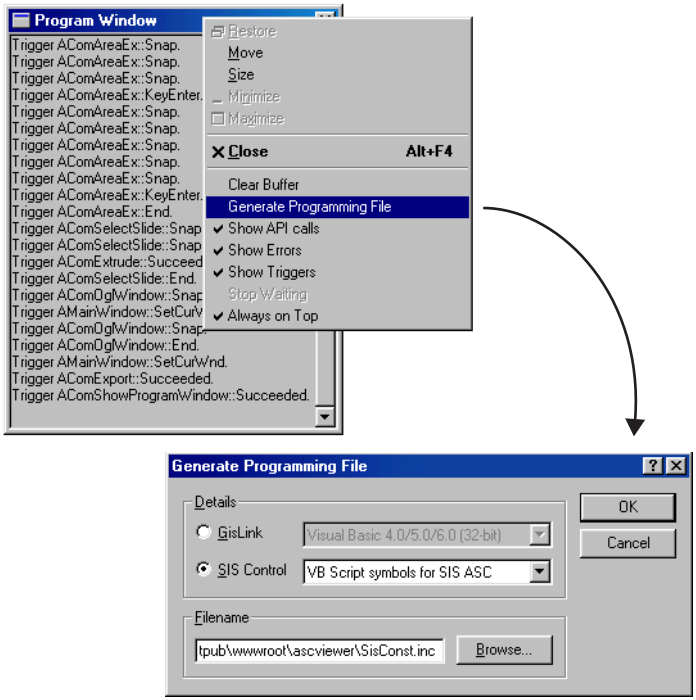


Select **Save As** from the File menu, and browse to the ascviewer folder (Virtual Directory) you created in Stage 1. Call the file `ascviewer.swd`. We will refer to this file in our HTML code later on.

◆ SisConst.inc

This file is essential for the ASC methods to work. It must be generated from a Cadcorp SIS desktop product and then saved into the directory in which the ASC application will run from, such as the ascviewer folder.

To generate the file, display the Program Window, then choose Generate Programming File from the title bar's local menu.



Select the SIS Control, ASC option. Browse to the folder where your virtual directory is located. Call the file SisConst.inc.

◆ Render.asp

Construct the render page in an HTML editor. It consists of one line of code, which specifies the size of the image returned from the ASC. Make sure it is the same size in all the relevant pages. Save to the **ascviewer** directory.

Listing 5.3 RENDER.ASP

```
<% Session("Sis").Render (Session("xSize"),Session("ySize"),"image/jpeg"%>
```

If an alternative image file type is required, use one of the following instead of `"image/jpeg"`: `"image/gif"`, `"image/png"`, `"image/x-MS-bmp"`, or `"image/x-png"`.

◆ Map.asp

This is the element of the site where you create your on-line GIS application. In this example, we will first add in the ASC image. Next, we will add in some viewing tools, and finally we will create a query mechanism.

The first job is to initialise the ASC on the page. In your HTML editor, type the following code and save it as Map.asp in your ascviewer folder.

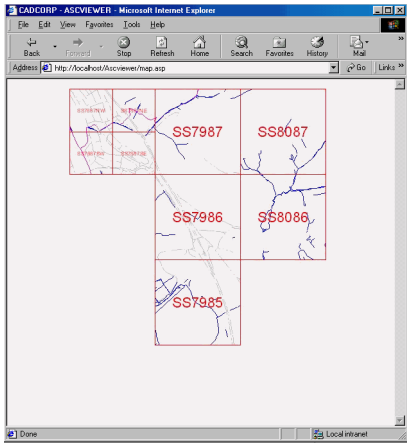
Listing 5.4 MAP.ASP

```
<html>
<head>
<meta http-equiv="Content-Language" content="en-gb">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>CADCORP - ASCVIEWER</title>
<base target="_self">
</head>
<!-- #include file = "SisConst.inc" -->
<%
 ' load swd and pass to renderer

    ' Store the image size.
     Session("xSize")=400
     Session("ySize")=400

    If Not IsObject(Session("Sis")) Then
     ' Create the SIS Active Server Component
     Set oSis=Server.CreateObject("SisASC.SisASC.6")
     oSis.LoadSwd Server.MapPath("ascviewer.swd")
     Set Session("Sis")=oSis
    End If

    Set oSis = Session("Sis")

    strX=Request.Form("Map.X")
    strY=Request.Form("Map.Y")
%>
<body leftmargin="0" link="#333399" bgcolor="#FFFFFF">
<form METHOD="POST" ACTION="map.asp" name="mapform">
<div align="center"><font face="Arial">
<input type="image" name="Map" src="Render.asp"
width="<%=Session("xSize")%>"   height="<%=Session("ySize")%>"
align="middle"></font>
</div>
</form>
<!-- Whole form -->
</body>
</html>
```

Save this code in the ascviewer directory. In your web server software (PWS/ IIS), make sure that the server is running. Now, in Internet Explorer, browse to the folder in which you have saved your ASC files:

http://localhost/ascviewer/map.asp

The following should appear in your browser:



If the above does not appear, retrace your steps and make sure that all the files listed are resident in your virtual directory and that the file names are correct.

Make sure that;

- the SWD is named ascviewer.swd and is referred to correctly in map.asp
- the include file SisConst.inc is in the Virtual Directory and is referred to in map.asp
- the Server object in Map.asp is set to version 6:
  `oSis=Server.CreateObject("SisASC.SisASC.6")`

■ Adding tools to MAP.ASP

The example above is good for serving a single view of an SWD or data. The next stage is to allow the user the ability to browse the data more interactively.

◆ Adding Zoom tools

We will first add Zoom In and Zoom Out tools to map.asp. To do this, we call ASC methods using VBscript. Open your HTML editor and add the following two lines to the FORM section of your page.

Listing 5.5 Adding zoom buttons to form in Map.asp

```
<body leftmargin="0" link="#333399" bgcolor="#FFFFFF">
<form METHOD="POST" ACTION="map.asp" name="mapform">
 <div align="center">
  <p><font face="Arial"><b>
   <input type="image" name="Map" src="Render.asp" width="400"
    height="400" align="middle">
   </b></font></p>
  <p>
   <input type ="submit" name="ZOOMIN" value="Zoom In">
   <input type ="submit" name="ZOOMOUT" value="Zoom Out">
```

```
   </p>
  </div>
 </form>
<!-- Whole form -->
</body>
</html>
```

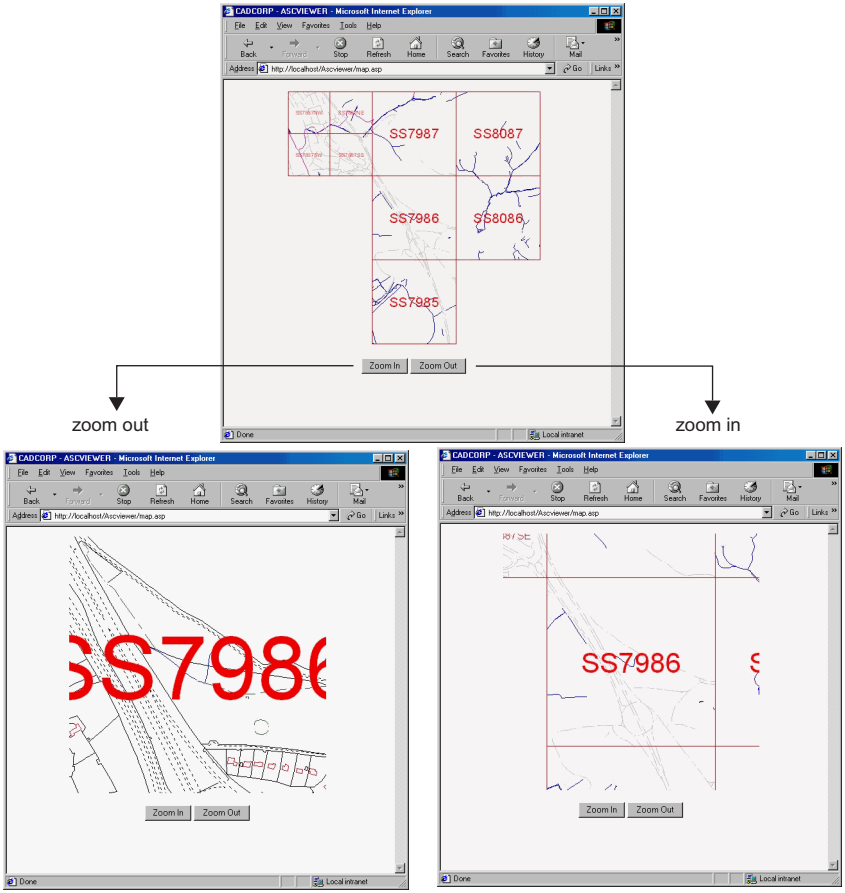
This will add two buttons to FORM on the Map.asp page. Add the lines shown below to the script portion of Map.asp.

Listing 5.6 Adding to script section of Map.asp

```
<%
 ' load swd and pass to renderer
     If Not IsObject(Session("Sis")) Then
      ' Create the SIS Active Server Component
      Set oSis=Server.CreateObject("SisASC.SisASC.6")
      oSis.LoadSwd Server.MapPath("ascviewer.swd")
      Set Session("Sis")=oSis
     End If

     Set oSis = Session("Sis")
     strX=Request.Form("Map.X")
     strY=Request.Form("Map.Y")
  If Request.Form("ZOOMIN") <>"" Then
     oSis.ZoomView 0.5
  End if
  If Request.Form("ZOOMOUT") <>"" Then
     oSis.ZoomView 1.5
  End if
%>
```

Once the user clicks on the buttons, the rendered image will Zoom In or Out of the SWD.

The page will appear as follows (note the location of the tool buttons):



◆ Adding Pan tools

The next items to add are Pan tools. These will allow the user to move the map image North, South, East and West. Instead of using Form Button Objects, this time we will use images to act as the means of moving the map image. Each of the images will move the map a given amount North, South, East, or West.

The first thing to do is to add the four Form Image objects to the HTML. The following example adds four images called north.jpg, south.jpg, east.jpg, and west.jpg to the page. The rendered image is also now placed in a table to assist the layout.

Within the FORM element on the HTML code, the following will create an Image object for the Pan South command. This uses an image called south.jpg which is stored in the ascviewer folder.

Listing 5.7 Image object for the Pan South command

```
<input type="image" border="0" name="PANSOUTH" src="south.jpg" width="40"
height="18">
```

Within the script section of the page, the following code will respond to the user selecting the PANSOUTH object and move the centre of the map image to the south accordingly.

Listing 5.8 Pan button action

```
If Request.Form("PANSOUTH.x")<>"" Then
  oSis.SplitExtent x1,y1,z1,x2,y2,z2,oSis.GetViewExtent
  yd=(y2-y1)/4
  oSis.GetViewExtent x1,y1-yd,z1,x2,y2-yd,z2
End if
```

Obviously this needs to be repeated for the other three directions.

Try adding a **button** which allows the user to zoom to the Full Extent of the SWD. As a hint, here are the elements you will require.

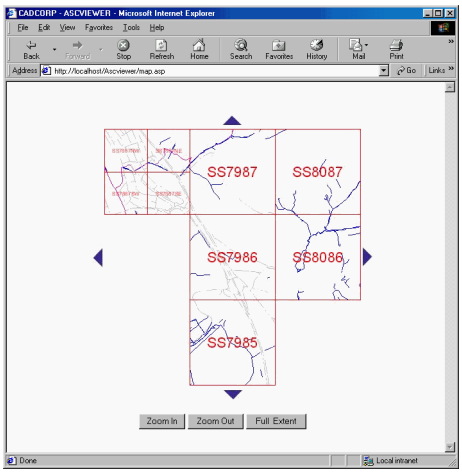
Listing 5.9 HTML

```
<input type="submit" name="FULLEXTENT" value="Full Extent">
```

Listing 5.10 VB Script

```
If Request.Form("FullExtent") <>"" Then
 oSis.ZoomExtent
end if
```

The map.asp page should now look like this:



■ Adding a query function

We now have all the basic viewing elements a non-expert GIS user may require. As an example of a more advanced programming call, the map.asp user is now going to be able to select items in the map image, and have information displayed about them on

the screen. In this case, because we are using Ordnance Survey Landline data it would be helpful for users if text appeared when they clicked on the map telling them what the item was (Building Outline, General Line Detail, County Boundary and so on). In the following example, the object property `_DESC$` will be passed to a string variable, `strDescription`, which can then be printed in the HTML code using the line:

```
<% response.write strDescription %>
```

Add the following into the script element of the page.

Listing 5.11 Checking for user click

```
If strX <>"" and strY<>"" then
   oSis.GetViewPos x1, y1, z1, strX, strY,400, 400
   oSis.OpenClosestItem x1, y1, z1, 1000000, "V", ""
   strDescription = "You selected: " & oSis.GetStr(SIS_OT_CURITEM,
     0,  "_DESC$") & "<p>"
End if
```

The code checks to see if the user has clicked on the map image (`if strX <>”” and strY <>””`). It then gets the position of the cursor and translates it into the real world co-ordinates which are held in the SWD (`oSis.GetViewPos`). The code then selects the nearest point to the cursor (`oSis.OpenClosestItem`) and the property is passed on to the user (`Sis.GetStr`).

The final element to add is a button to toggle data on and off. This may be useful if an ASC application deals with many overlays or datasets. To toggle data on and off we will need to use ASP Session variables. These variables will be stored in memory regardless of the fact that the browser is refreshed: the ASP Session will remember that a layer is on or off, and will alter it accordingly when the user presses the Data toggle button.

To set the Session variables, add the following subroutine to the script.

Listing 5.12 Setting session variables

```
Sub Session_OnStart
  Session("data") = False
  Session("zi") = 0
End Sub
```

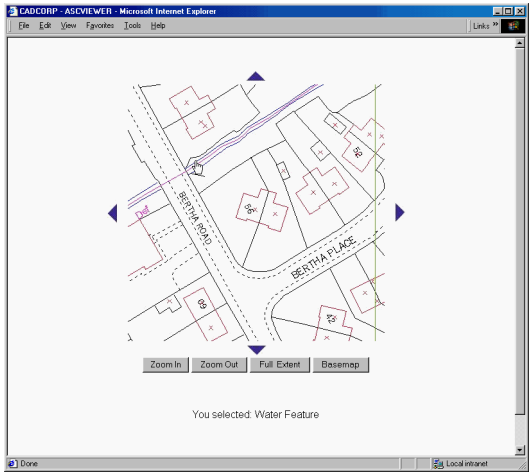
The variables `Session("data")` and `Session("zi")` act as counters in a loop which checks to see if the data (the first overlay in the SWD in this case) is on or off. It then turns it off, if it is on, and vice versa. The code is shown below. You will need to add a button called DATATOGGLE to the form.

Listing 5.13 DATATOGGLE code

```
If Request.Form("DATATOGGLE")<>"" Then
 If Session("data")= false and Session("zi")= 0 then

 oSis.SetInt SIS_OT_OVERLAY,0,"_status&",SIS_VISIBLE
 Session("data")= true
 Session("zi")= 1
 End if
If Session("data")= true and Session("zi")= 0 then
     oSis.SetInt SIS_OT_OVERLAY,0,"_status&", SIS_INVISIBLE
     Session("data")= false
     Session("zi")= 1
```

```
End if
Session("zi")= 1
    End if
    Session("zi") = 0
   End if
```

The finished application should now look like this:



Listing 5.14 The ASCVIEWER Map.asp code

```
<html>
<head>
<meta http-equiv="Content-Language" content="en-gb">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>CADCORP - ASCVIEWER</title>
<base target="_self">
</head>
<!-- #include file = "SisConst.inc" -->
<%
Sub Session_OnStart
 Session("zones") = False
 Session("zi") = 0
End Sub
'-------------------------------------
 'load swd and pass to renderer
If Not IsObject(Session("Sis")) Then
'Create the SIS Active Server Component
 Set oSis=Server.CreateObject("SisASC.SisASC.6")
 oSis.LoadSwd Server.MapPath("ascviewer.swd")
 Set Session("Sis")= oSis
End If
Set oSis= Session("Sis")

strX=Request.Form("Map.X")
strY=Request.Form("Map.Y")
'---------------------------------------------
If Request.Form("DATATOGGLE")<>"" Then
```

```
If  Session("data")= false  and  Session("zi")= 0 then
 oSis.SetInt SIS_OT_OVERLAY,0,"_status&",SIS_VISIBLE
 Session("data")= true
 Session("zi")= 1
End if
If  Session("data")= true  and  Session("zi")= 0 then
     oSis.SetInt SIS_OT_OVERLAY,0,"_status&",SIS_INVISIBLE
     Session("data")= false
     Session("zi")= 1
    End if
    Session("zi") = 0
End if
'-----------------------------------------------
If strX <>""  and strY<>"" then
 oSis.GetViewPos x1, y1, z1, strX, strY,400, 400
 oSis.OpenClosestItem x1, y1, z1, 1000000, "V", ""
 strDescription = "You selected: " & oSis.GetStr(SIS_OT_CURITEM, 0, "_DESC$")
& "<p>"
 strClass = oSis.GetStr(SIS_OT_CURITEM, 0, "_class$")
End if
'-----------------------------------------------
If Request.Form("ZOOMIN") <>"" Then
 oSis.ZoomView 0.5
End if

If Request.Form("ZOOMOUT") <>"" Then
 oSis.ZoomView 1.5
End if
If Request.Form("FullExtent") <>"" Then
 oSis.ZoomExtent
End if
If Request.Form("PANNORTH.x")<>"" Then
 oSis.SplitExtent x1,y1,z1,x2,y2,z2,oSis.GetViewExtent
 yd=(y2-y1)/4
 oSis.SetViewExtent x1,y1+yd,z1,x2,y2+yd,z2
End if

If Request.Form("PANEAST.x")<>"" Then
 oSis.SplitExtent x1,y1,z1,x2,y2,z2,oSis.GetViewExtent
 xd=(x2-x1)/4
 oSis.SetViewExtent x1+xd,y1,z1,x2+xd,y2,z2
End if

If Request.Form("PANSOUTH.x")<>"" Then
 oSis.SplitExtent x1,y1,z1,x2,y2,z2,oSis.GetViewExtent
 yd=(y2-y1)/4
 oSis.SetViewExtent x1,y1-yd,z1,x2,y2-yd,z2
End if

If Request.Form("PANWEST.x")<>"" Then
 oSis.SplitExtent x1,y1,z1,x2,y2,z2,oSis.GetViewExtent
 xd=(x2-x1)/4
 oSis.SetViewExtent x1-xd,y1,z1,x2-xd,y2,z2
 end if
%>
```

```
<body leftmargin="0" link="#333399" bgcolor="#FFFFFF">
<form METHOD="POST" ACTION="map.asp" name="mapform">
<div align="center">
<p>  </p>
<table width="61%" border="0" cellspacing="0" cellpadding="0">
<tr>
<td width="50"> </td>
<td width="656">
<div align="center">
<input type="image" border="0" name="PANNORTH" src="north.jpg" width="40"
height="18">
</div>
</td>
<td width="96"> </td>
</tr>

<tr>
<td width="50">
<div align="right">
<input type="image" border="0" name="PANWEST" src="west.jpg" width="18"
height="40">
<div>
</td>
<td width="656">
<div align="center"><font face="Arial"><b>
<input type="image" name="Map" src="Render.asp" width="400" height="400">
</b></font></div>
</td>
<td width="96">
<div align="left">
<input type="image" border="0" name="PANEAST" src="East.jpg" width="18"
height="40">
</div>
</td>
</tr>

<tr>
<td width="50"> </td>
<td width="656">
<div align="center">
<table width="86%" border="0" cellspacing="0" cellpadding="0">
<tr>
<td>
<div align="center">
<input type="image" border="0" name="PANSOUTH" src="south.jpg" width="40"
height="18">
</div>
</td>
</tr>
<tr>
<td>
<div align="center">
<input type="submit" name="ZOOMIN" value="Zoom In">
<input type="submit" name="ZOOMOUT" value="Zoom Out">
<input type="submit" name="FULLEXTENT" value="Full Extent">
```

```
<input type="submit" name="DATATOGGLE" value="Basemap">
</div>
</td>
</tr>
<table>

<p> </p>
<%response.write strDescription%>
</div>
</td>
<td width="96"> </td>
</tr>
</table>
<p>  </p>
<p>  </p>
</div>
</form>
<!-- Whole form -->
</body>
</html>
```

■ OpenGIS Web Servers

◆ The Web Map Server, Web Feature Server, Web Terrain Server, Gazetteer, and the ASC

Cadcorp SIS ASC can also serve as a client for OpenGIS Web Map Servers. These servers can potentially be located anywhere, and their data can be accessed via software which implements the appropriate OpenGIS Consortium (OGC) specifications. The ASC can also act as an OpenGIS Server, so that any client application (eg internet browsers or Cadcorp SIS) can access the data contained on these servers.

When using the Web Map Server, data can be returned as a raster image, or, when using the Web Feature Server, as vectors. To access this functionality, several methods are exposed to the programmer in the ASC which require parameters to be set. This is similar to the way the Render method works.

For more information, see www.opengis.org.

The Cadcorp SIS ASC methods `WMS()`, `WTS()`, `WFS()` and `Gazetteer()` all take a single string as argument. This string is the 'query string' of the HTTP request that is sent to the W?S server, ie the part of the Uniform Resource Locator (URL) after the question mark.

To clarify this, here is an example of its usage. Suppose you have an Active Server Page (ASP) named WFS.asp with the following content:

```
========== WFS.asp ============
<%
  Sis.WFS Request.QueryString
%>
==============================
```

WFS requests are sent to the page WFS.asp, for example through the URL:

http://www.someserver.net/scripts/WFS.asp?request=GetCapabilities&version=1.0.0&service=wfs

The query string is:

`"request=GetCapabilities&version=1.0.0&service=wfs"`

The result of the request is directly written into the ASP Response object, similar to the `Sis.Render` method.

It would also be possible to hard code the query string in, for example, WMS.asp:

```
========== WMS.asp ============
<%
  Sis.WMS
"request=GetMap&service=wms&version=1.1.0&format=image/png&width=400&
height=300&layers=...&srs=...&bbox=..."
%>
==============================
```

in which case WMS.asp would always return the same picture.

It is incorrect to send a WFS request to a WMS server, and vice versa; and similarly with the other servers. If this occurs, you will get an error.

WFS (`service=wfs`) and Gazetteer (service=gaz) support the requests (*request=... parameter value*) `GetCapabilities`, `DescribeFeatureType`, and `GetFeature`.

WMS (`service=wms`) supports the requests `GetCapabilities` and `GetMap`.

WTS (`service=wts`) supports the requests `GetCapabilities` and `GetView`.

The exact definition of all the parameters allowed in the query string can be found on the OpenGIS website.

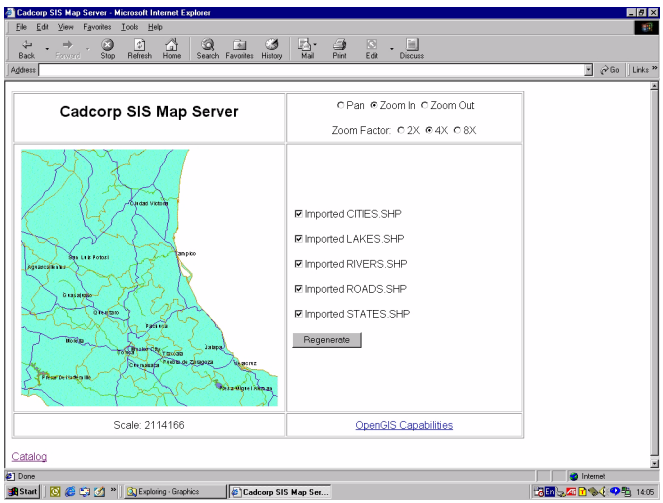
Chapter 6

Cadcorp SIS Map Server

- Cadcorp SIS Map Server. 67
- Components of Cadcorp SIS Map Server 68
- Requirements 68
- The API requests 69
- Establish a virtual directory 71
- Adding content to your Map Server. 72
- Creating the HTML file 72

■ Cadcorp SIS Map Server

The Cadcorp SIS Map Server gives you the ability to serve SWD files on the web quickly and easily. You can use Cadcorp SIS to create maps which can then be easily distributed around an organisation and its customers. The following picture shows the Cadcorp SIS Map Server in action, showing an SWD in a web page, and featuring Pan, Zoom, and Layer inclusion.



The Cadcorp SIS Map Server has been developed around the Web Map Overlay Specification. This was defined by the OpenGIS Consortium, through its GIS industry-wide Web Mapping Testbed (WMT) initiative. It defines an open use of geospatial data on the web. The aim is to allow web users to access web servers and display geographical data, regardless of the differences in GIS data formats or GIS software.

◆ The capabilities of the Cadcorp SIS Map Server

The Cadcorp SIS Map Server allows a web server to perform the following tasks:

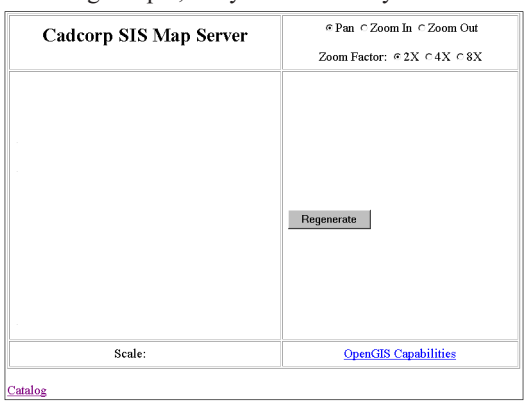
- display a map in any standard browser without the need for plugins
- display in a browser a list of available SWDs stored on the server
- display a map window of a SWD with basic information about the content of the file, such as the list of overlays within it and its scale. It also provides the possibility to perform basic zoom and pan commands.
- provide information in XML format about the SWDs found, with additional information, such as their name, current extents, scale and projections

Cadcorp SIS Map Server is essentially a simple entry-level web product. To create sophisticated web applications that can exploit the full functionality of Cadcorp SIS, use Cadcorp ASC (Active Server Component).

■ Components of Cadcorp SIS Map Server

Cadcorp SIS Map Server consists of the following components:

- Cadcorp SIS Map Modeller, to create the map and save it as a SWD
- the dynamic link library SisIsapi.dll, the engine of the Map Server. This is created using the Internet Server Application Program Interface (ISAPI). It allows a user to access a web server using any standard web browser, request map information, get that information back and see it displayed on their web page.
- a standard HTML page, to present your served map on a web page. We supply the following sample, but you can create your own.



■ Requirements

You need the following to install, run, and begin developing with the Cadcorp SIS Map Server:

- Cadcorp SIS Map Modeller, to generate a saved window definition (SWD) with data in it
- licence code and hardware key

- a server computer running Microsoft Internet Information Server (Windows NT Server, Windows 2000, and XP). The server must have the hardware key attached to its parallel port and the user must have administrator privileges, or equivalent.
- a client computer which can access the server on a Local Area Network (LAN) or the World Wide Web (WWW), with Netscape Navigator or Internet Explorer 4.0 or above installed on it
- an HTML editor such as Microsoft FrontPage or Macromedia Dreamweaver. We do not recommend using a text editor for large sites, where coding can become complex.

◆ Web server software

Internet Information Service must be set up on your machine. This product allows you to perform web server administration tasks necessary in order to set up a Cadcorp SIS Map Server.

We recommend that appopriate server hardware and software be used when deploying Cadcorp SIS Map Server (and Cadcorp SIS ASC). Suitable software would include Microsoft Internet Information Server (IIS) for Windows NT, 2000, and XP.

■ The API requests

The Cadcorp SIS Map Server responds to the following requests:

Request	Response
`GetMap`	provides clients of the with pictures of maps
`SwdCatalog`	provides an HTML page that lists the available SWDs on the server
`BasicPanZoom`	provides an HTML page with a map and basic viewing controls
`GetCapabilities`	provides clients an XML file that describes the current capabilities: the SWDs that can be served, what formats they can be served in, and related information.

◆ GetMap request

An example of this request is:

```
http://Server_address/SisISAPI.dll?Request=map&WMTVER=1.0.0&SRS=EPSG:27700
&LAYERS=LANDLINE:0,LANDLINE:1&BBOX=278000,187500,278500,188000&WIDTH=400&H
EIGHT=400&FORMAT=JPG
```

The parameters to be passed are:

`http://server_address/SisIsapi.ddl`	the URL prefix of the server where the SisISAPI.dll file resides
`?Request=map`	describes the requests
`&WMTVER=1.0.0`	the version of the request

`&SRS=srs_identifier`	Spatial Reference System: a text parameter that defines a horizontal coordinate reference system code. This is generally defined through an EPSG (European Petroleum Survey Group) numeric identifier.
`&LAYERS=layer_list`	comma separated list of one or more map layers
`&BBOX= xmin, ymin, xmax, ymax`	a bounding box defined as a set of comma-separated values. They must define an are that is contained within the `BBOX` as specified in the Capabilities XML for each layer in the LAYERS list.
`&WIDTH=width`	width in pixels of the map picture
`&HEIGHT=height`	height in pixels of the map picture
`&FORMAT=format`	output format of the map (valid formats include: `GIF`, `JPEG`, `PNG`, and `TIFF`)

◆ SwdCatalog request

An example of this request is:

```
http://Server_address/SisISAPI.dll?SwdCatalog
```

The only parameter to be passed is the request name, SwdCatalog.

◆ BasicPanZoom request

An example of this request is:

```
http://Server_address/SisISAPI.dll?BasicPanZoom&swd=LANDLINE&overlaymask=-
1&width=400&height=400&bbox=278000,187500,278500,188000
```

The parameters to be passed are:

`http://server_address/SisIsapi.ddl`	the URL prefix of the server where the SisISAPI.dll file resides
`Request=BasicPanZoom`	type of request
`&SWD=overlay`	which SWD to display
`&OVERLAYMASK=integer`	which overlays are drawn
`&WIDTH=width`	width in pixels of the map picture
`&HEIGHT=height`	height in pixels of the map picture
`&BBOX= xmin, ymin, xmax, ymax`	the bounding box of the map view
`&MODE=mode`	the mouse click operation: 1 (pan), 2 (zoom in), or 3 (zoom out)
`&ZOOM=factor`	the zoom factor (2, 4, or 8)

◆ GetCapabilities request

An example of this request is:

`http://Server_address/SisISAPI.ddl?GetCapabilities`

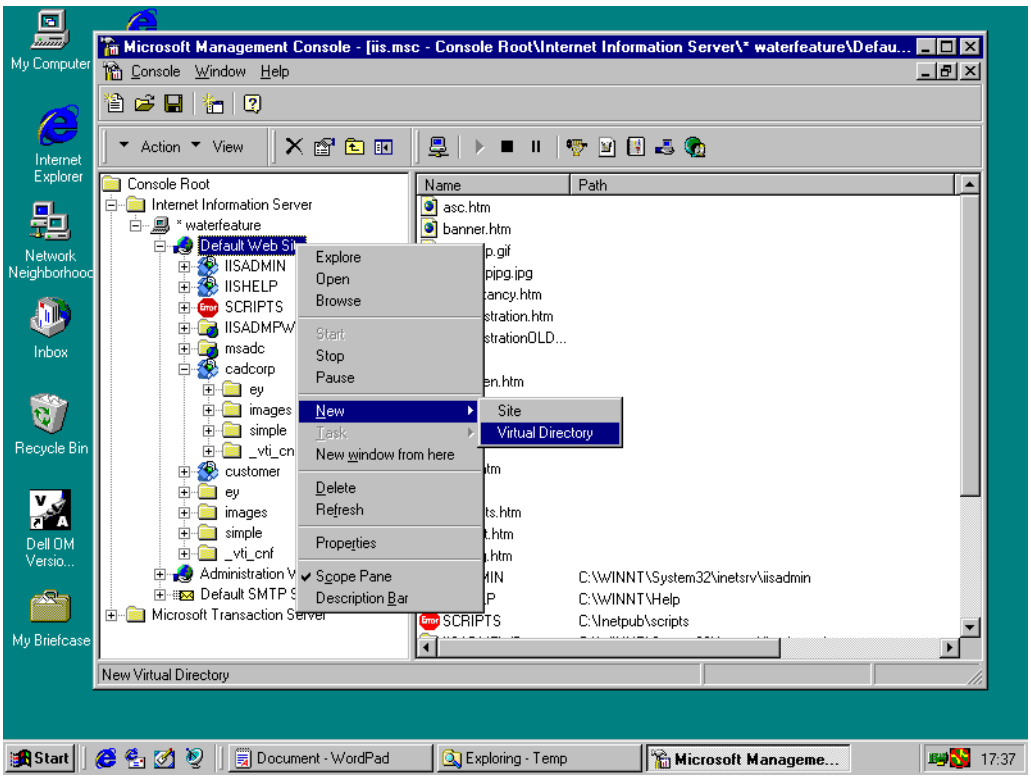
The only parameter to be passed is the request name, `GetCapabilities`.

■ Establish a virtual directory

The first part of getting a site up and running is to create a Virtual Directory on the web server. A virtual directory is an alias that points to a physical directory on the server hard drive where all your maps (SWD files) are located. When an internet user sends a request to the map server to access the SWD files in the virtual directory, the server knows, thanks to the virtual directory, that it has to respond by making accessible the SWD files contained within the Cadcorp SIS Map Server directory.

Create a folder on your server's hard-disk and call it ascviewer. Next, go into the IIS Management Console (Windows NT, XP, or 2000).

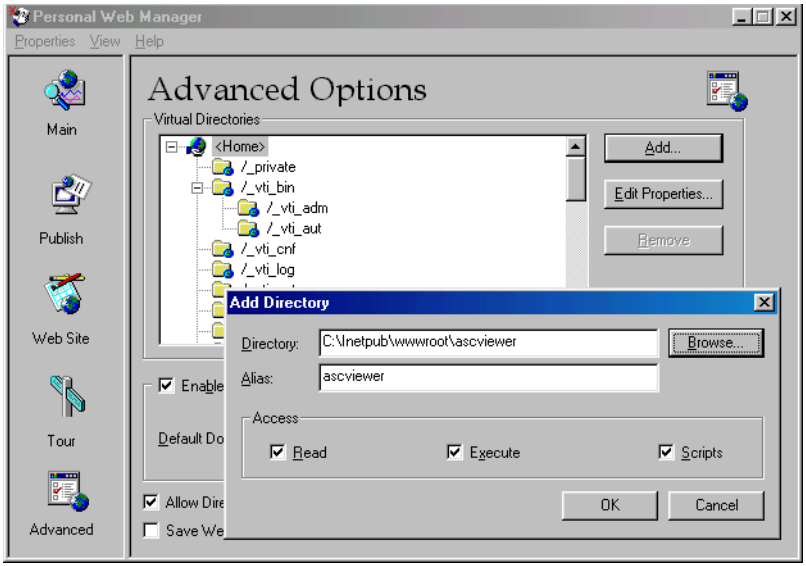
Select New>Virtual Directory:



Browse to the folder you named ascviewer and give the alias value the same name. The alias is the name the user will type in to get to the folder, eg:

http://server/alias/map.asp

When you create the Virtual Directory, make sure that you allow Read, Execute and Scripts by ticking in the check boxes – ASPs will not work unless you do.



Once you have completed creating your virtual directory, stop and then restart the web server in IIS.

■ Adding content to your Map Server

Put the following file and folders into the physical directory:

- the SisIsapi.dll file that is shipped with Cadcorp SIS. Copy this across from the Cadcorp SIS installation folder, typically C:\Program Files\Cadcorp SIS V6.0\.
- the Libraries and Plugins folders that are installed on your machine when Cadcorp SIS is installed. Again, copy these across from the Cadcorp installation folder, typically C:\Program Files\Cadcorp SIS V6.0\Libraries\ and C:\Program Files\Cadcorp SIS V6.0\Plugins\.
- the SWDs files that are to be made available through the Map Server requests. Those files that may be linked by the SWDs need not be in this same physical directory.
- the HTML file that contains the requests that can be submitted by the clients. This is described in the following section.

■ Creating the HTML file

The following example HTML file (➲page 73, **Listing 6.1 DEFAULT.HTM**) contains all the requests explained in the previous section. The file has been called default.html. By default, Personal Web Server will serve the file default.html when a request is made to the server. For example, if the alias is SISmapServer, and the physical directory contains the file default.html, when an Internet client uses the URL

http://*Server_Name*/SISmapServer, the user will automatically see the content of this file. A URL such as http://*Server_Name*/SISmapServer/default.htm is not needed.

The example HTML file assumes that a physical directory has been created, and that it contains three SWDs: Canada, Landline and Mexico. When the file is requested, it will display the name of the four requests that have been explained above, with a short description for each one, namely: `SwdCatalog`, `BasicPanZoom`, `GetCapabilities` and `GetMap`.

The requests are sent to the Map Server using the hypertext anchor HTML element `<A>` and its attribute `HREF`. This is a common way of sending a request to a server and the result is a URL query string. For example:

```
<A href="SisIsapi.dll?SwdCatalog">SwdCatalog</A>
```

sends the server the following request when the user clicks on SwdCatalog:

```
http://Server_Name/SISMapServer/SisIsapi.dll?SwdCatalog.
```

Effectively, the way of sending the appropriate request to the server is specifying SisIsapi.dll followed by a question mark and the name of the request and its parameters as the `HREF` attribute for the `<A>` tag.

Listing 6.1 DEFAULT.HTM

```
<HTML>
<HEAD>
<TITLE>Cadcorp SIS Internet Map Server</TITLE>
</HEAD>
<BODY>
<TABLE>
 <TR>
  <TD> <A href="SisIsapi.dll?SwdCatalog">SwdCatalog</A>
   <P>
    This method will display a list of SWDs.
</TD>
 </TR>
 <TR>
  <TD>
   <P>
    <A
href="SisISAPI.dll?BasicPanZoom&amp;swd=LANDLINE&amp;overlaymask=-1&amp;wi
dth=400&amp;height=400&amp;bbox=278000,187500,278500,188000">BasicPanZoom<
/A>
   <P>
    This request will display a map window with functions to zoom in/out.
and Pan
</TD>
 </TR>
 <TR>
  <TD>
   <P>
    <A href="SisISAPI.dll?GetCapabilities">GetCapabilities</A>
   <P>
This request will list the current capabilities in xml format, describing all
found swds and their overlays and what the current extents are.
</TD>
 </TR>
 <TR>
  <TD>
```

```
    <P>
     <A
href="SisISAPI.dll?Request=map&amp;WMTVER=1.0.0&amp;SRS=EPSG:4326&amp;LAYE
RS=CANADA:0,CANADA:1,CANADA:2,CANADA:3&amp;BBOX=-141.003,18.0584,-52.6203,
106.441&amp;WIDTH=400&amp;HEIGHT=400&amp;FORMAT=JPG">GetMap</A>
    <P>
     This request will display a map in a browser.
 </TR>
</TABLE>
</BODY>
</HTML>
```

Chapter 7

Methods

■ **Introduction** 75
■ **Alphabetical list of methods** 76
■ **Notes** 219

■ Introduction

This chapter describes all the methods available in Cadcorp SIS. They are listed in alphabetical order.

Syntax examples for methods which are available to both GisLink and Cadcorp SIS Control applications show the `Gis` prefix. Programmers using the Cadcorp SIS Control should omit this prefix and use the control name prefix. For example:

GisLink syntax	`GisCreateLabelTheme (`*`formula`*`)`
Cadcorp SIS Control syntax	`Sis.CreateLabelTheme (`*`formula`*`)`

Methods which are available in only one programming environment show syntax examples specific to the environment. Methods which provide a significant difference in function in these environments are listed twice, with specific syntax examples and notes.

◆ Data types

In the method descriptions, arguments have the following data types.

`SHORT INTEGER` a data type that holds integer variables stored as 2-byte whole numbers in the range –32 768 to 32 767. The Integer data type is also used to represent enumerated values.

`LONG INTEGER` a 4-byte integer ranging in value from –2 147 483 648 to 2 147 483 647

`DOUBLE` a data type that holds double-precision floating-point numbers as 64-bit numbers in the range –1.79769313486232E308 to –4.94065645841247E–324 for negative values; 4.94065645841247E–324 to 1.79769313486232E308 for positive values.

`STRING` a data type consisting of a sequence of contiguous characters that represent the characters themselves rather than their numeric values. A String can include letters, numbers, spaces, and punctuation.

The String data type can store fixed-length strings ranging in length from 0 to approximately 63 000 characters and dynamic strings ranging in length from 0 to approximately 2 billion characters.

■ Alphabetical list of methods

❍ Activate

Activate the SIS Control.

Syntax `SIScontrolName.Activate ( )`

Example `SIS.Activate`
makes the Cadcorp SIS control `SIS` the active control

Notes In Microsoft Visual C++ applications, you should use this method instead of the normal ActiveX method `SetFocus`. Microsoft Visual Basic applications can use the `SetFocus` method.

Available OV, OM, OD

❍ ActivateWnd

Activate a window by its number, ie make the window the currently active child window, and its SWD the current SWD.

Syntax `GisActivateWnd ( number )`

Arguments *number* SHORT INTEGER
the number of the window to activate (starting at 0)

Example `GisActivateWnd 2`
makes the third window the current window

Available MM, ME, MD

❍ AddCommand (Cadcorp SIS Control)

Add an application-defined command to the menu with restrictions on use. Cadcorp SIS Control applications can add only to the local menus.

Syntax

```
ControlName.AddCommand ( menu, help, clsName, min, max, filter,
locus )
```

Arguments

menu STRING

the menu command string. Applications which use the Cadcorp SIS Control will have the AppCommand event called with this string as an argument. To add to an existing menu, this string must exactly match the menu text and hash (#) characters to signify a sub-menu. Keyboard shortcuts are not supported on the local menu.

help STRING

the prompt to be displayed

clsName STRING

the class or group name this command is to be associated with. If set to `"Item"` or `""`, the command will be available for all item classes.

min SHORT INTEGER

the minimum number of items which must be selected to make this command valid:

–1 hittable and editable items are valid

0 no items have to be selected

1 only editable items are valid

max SHORT INTEGER

the maximum number of items which may be selected to enable this command:

–1 no limit

If both *min* and *max* are set to 0 and the command is a local command, the command will appear on the local menu only when no items are selected.

filter STRING

optionally specifies a named filter which all selected items must pass for the command to be available

locus STRING

optionally specifies a named locus which all selected items must fall within for the command to be available

Example

```
SIS.AddCommand "Display Property Details", _
  "Displays the details of the current Selected Property", "Point", 1, _
  1, "Properties", "Spatial Search"
```

the local menu Display Property Details will be available whenever a single point item which matches the filter Properties, but is also within the locus Spatial Search, is selected

Available OV, OM, OD

❍ AddCommand (GisLink)

Add an application-defined command to the menu, with restrictions on use.

Syntax

```
GisAddCommand ( menu, help, clsName, min, max, filter, locus )
```

Arguments *menu* STRING

the menu command string. GisLink customisations must have a button on the main form with this string as the Caption. To add to an existing menu, this string must exactly match the menu text, including ampersand (&) characters preceding any underlined keyboard shortcut and pipe (|) characters to signify a sub-menu.

If this string does not include a pipe symbol, the command will be added to the local, right-mouse menu. Sub-menus on the local menu are denoted by the hash (#) symbol.

Keyboard shortcuts are not supported on the local menu.

help STRING

the prompt to be displayed in the prompt panel of the main window

clsName STRING

the class or group name this command is to be associated with. If set to `"Item"` or `""`, the command will be available for all item classes.

min SHORT INTEGER

the minimum number of items which must be selected to make this command valid:

–1 hittable and editable items are valid

0 no items have to be selected

1 only editable items are valid

max SHORT INTEGER

the maximum number of items which may be selected to enable this command. If max is –1, there is no limit. If both *min* and *max* are set to 0, and the command is a local command, the command will appear on the local menu only when no items are selected.

filter STRING

an optional named filter which all selected items must pass for the command to be available

locus STRING

an optional named locus which all selected items must fall within for the command to be available

Example

```
GisAddCommand "Display Property Details", _
  "Displays the details of the current Selected Property", "Point", 1, _
  1, "Properties", "Spatial Search"
```

whenever a single point item, which matches the filter Properties, but is also within the locus Spatial Search, is selected, the local menu Display Property Details will be available

Available MM, ME, MD

❍ AddToList

Add the current open item to a named list. If the list does not exist, it will be created.

Syntax `GisAddToList ( )`

Arguments *list* STRING

the named list to which to add the current open item

Available MM, ME, MD, OV, OM, OD, ASC

❍ AllowCommands (GisLink)

Add or remove commands from the menu. To find the ACom equivalent for a menu command, turn on the Program Window using the command **Tools>Program Window**, select the command, and the ACom equivalent will be displayed in the window.

Syntax `GisAllowCommands ( flag, listcom )`

Arguments *flag* SHORT INTEGER

`SIS_COM_ALL`	allow all commands
`SIS_COM_ADD`	add the command(s) specified in *listcom*
`SIS_COM_REMOVE`	remove the command(s) specified in *listcom*
`SIS_COM_NONE`	disallow all commands

listcom STRING
a list of commands separated by space, comma, tab, or newline characters

Example `GisAllowCommand SIS_COM_REMOVE, "AComLayers, AComCopy, AComPaste"`
removes the Overlays, Copy, and Paste commands from the main and local menus

Notes If you wish the user to have access to only a few commands, you can remove all commands using `SIS_COM_REMOVE` and then, using `SIS_COM_ADD`, add back the allowed commands.

Commands removed in this way disappear from the menu system (main and local). The corresponding toolbar icon, if displayed, will be disabled (greyed). The commands can still be invoked using the `CallCommand` and `SwitchCommand` methods.

Available MM, ME, MD

❍ AllowCommands (Cadcorp SIS Control)

Add or remove commands from the menu. To find the ACom equivalent for a menu command, turn on the Program Window (**Tools>Program Window**), select the command, and the ACom equivalent will be displayed in the window.

Syntax `ControlName.AllowCommands ( flag, listcom )`

Arguments *flag* SHORT INTEGER

`SIS_COM_ALL`	allow all commands
`SIS_COM_ADD`	add the command(s) specified in *listcom*
`SIS_COM_REMOVE`	remove the command(s) specified in *listcom*
`SIS_COM_NONE`	disallow all commands

listcom STRING
a list of commands, separated by space, comma, tab, or newline characters

Example `SIS.AllowCommands SIS_COM_REMOVE, "AComLayers, AComCopy, AComPaste"`
removes the commands Overlays, Copy, and Paste from the default local menu

Notes If you want the user to have access to only a few commands, you can remove all commands, using `SIS_COM_REMOVE`, and then, using `SIS_COM_ADD`, add back the allowed commands. Commands removed in this way simply disappear from the menu, but can still be run using other API methods.

Available OV, OM, OD

❍ BezierTo

Draw a Bezier curve from the current drawing position. The curve is appended to the current line sequence, started by the last `MoveTo`, and extended using `BulgeTo`, `LineTo`, or this method.

Syntax
```
GisBezierTo ( x1, y1, z1, x2, y2, z2, x3, y3, z3 )
```

Arguments

x1, y1, z1 DOUBLE
the position of the first Bezier control point

x2, y2, z2 DOUBLE
the position of the second Bezier control point

x3, y3, z3 DOUBLE
the position of the end point of the Bezier curve

Notes The current drawing position will moved to the end of the curve, ie *x3*, *y3*, *z3*, after calling this method.

Available ME, MD, OD, ASC

❍ BulgeTo

Draw an arc from the current drawing position, through an angle, around a centre point. The arc is appended to the current line sequence, started by the last `MoveTo`, and extended using `BezierTo`, `LineTo`, or this method.

Syntax
```
GisBulgeTo ( angle, x, y )
```

Arguments

angle DOUBLE
the arc angle, in radians

x, y DOUBLE
the co-ordinates of the end point

Notes The current drawing position will move to the end of the arc after calling this method.

Available MM, ME, MD, OM, OD, ASC

❍ CallCommand

Call a non-interactive, or one-shot command (a command that does not require user intervention with the mouse) in the current SWD.

Syntax
```
GisCallCommand ( comname )
```

Arguments

comname STRING
the command to call

Example
```
GisCallCommand "AComRedraw"
```

Available MM, ME, MD

❍ CanDoCommand

Check whether or not a command can be executed. The result depends on the type of the current window and the selection within the current SWD. In Cadcorp SIS Control, the current licence level is also checked.

Syntax `rv = GisCanDoCommand ( comname )`

Arguments *comname* STRING
the command to check

Returns SHORT INTEGER

True the command can be executed

False the command cannot be executed

Example `Available = SIS.CanDoCommand ("AComExtrude")`
returns false if the licence level is Manager, true if the licence level is Modeller

Available MM, ME, MD, OV, OM, OD

❍ ChangeFeatureFilter

Include or Exclude feature code from a named feature filter.

Syntax `GisChangeFeatureFilter ( filter, fcode, flag )`

Arguments *filter* STRING
the named feature filter to edit, previously created using `CreateFeatureFilter`

fcode SHORT INTEGER
the feature code whose information is to be changed. Use 0 to specify all feature codes in the feature filter.

flag SHORT INTEGER

`SIS_FEATUREEXCLUDE` exclude from filter

`SIS_FEATUREINCLUDE` include in filter

Each of these flags may have the modifier `SIS_FEATURECASCADE` added to them, to apply the flag to all of the children of the specified feature code.

Examples
```
GisChangeFeatureFilter "LandLine", 1, SIS_FEATUREEXCLUDE
```
excludes the feature 1 from the filter LandLine
```
GisChangeFeatureFilter "Land-LineTest", 10102, SIS_FEATUREEXCLUDE _
  + SIS_FEATURECASCADE
```
excludes the feature code 10102 (Buildings), and its children, from the feature filter Land-LineTest

Available MM, ME, MD, OM, OV, OD, ASC

❍ ChangeLocusTestMode

Modify the test mode of a named locus.

Syntax `GisChangeLocusTestMode ( locus, geomTest, geomMode )`

Arguments *locus* STRING
the named Locus whose test mode is to be modified

geomTest SHORT INTEGER
the geometry test to use ➲page219, **Geometry tests**

geomMode SHORT INTEGER
the geometry test mode to use:

`SIS_GM_ORIGIN`	items whose origin (always a single point) must pass the testing method with the selected item
`SIS_GM_EXTENTS`	items whose extents (always a rectangle) must pass the testing method with the selected item
`SIS_GM_GEOMETRY`	items whose geometry must pass the testing method with the selected item

Example `ChangeLocusTestMode "Scheme20", SIS_GT_CONTAIN, SIS_GM_GEOMETRY`

Available MM, ME, MD, OM, OD, ASC

❍ ChangePrjUnits

Copy a named Transverse Mercator projection, changing the units, and replacing any existing projection with the same name.

Syntax `GisChangePrjUnits ( prjOut, prjln, mode, dSize )`

Arguments *prjOut* STRING
the named projection to create or replace

prjIn STRING
the named projection to copy

mode SHORT INTEGER
the units of *dSize*. This parameter must be set to 0, so that *dSize* is specified in metres.

dSize DOUBLE
the size of one projection unit

Example `GisChangePrjUnits "NatGrid10", "*APrjNatGrid", 0, 10`
create a new projection NatGrid10, based on the projection *APrjNatGrid, where the new unit is 10 times larger than the original.

Notes This routine can be used to adjust the unit size of Transverse Mercator projections, typically for use with external data sources, which are defined using m, cm, inches, and so on.

Available MM, ME, MD, OM, OD, ASC

❍ ChangeValueListFilter

Include or exclude a list of values from a value list filter.

Syntax `GisChangeValueListFilter ( filter, flag, listvar )`

Arguments *filter* STRING
the named value-list filter to change previously created using `CreateValueListFilter`

flag SHORT INTEGER

`SIS_FILTERRESET`	reset the value in the filter
`SIS_FILTERADD`	add to the filter
`SIS_FILTERREMOVE`	remove from the filter

listval STRING

a list of values, separated by spaces, commas, tabs, or newlines, to add or remove

Example `GisChangeValueListFilter "Rate Values", SIS_FILTERRESET, "10, 30, 50"`

Notes Value-list filters work only on integer properties, ie those with names ending in &.

Available MM, ME, MD, OM, OD, ASC

❍ CleanLines

Clean up line items, removing repeated vertices, and so on.

Syntax `GisCleanLines ( list, tolerance, options )`

Arguments *list* STRING

the named list containing the line items to be cleaned. Upon completion the named list will contain all the remaining line items.

tolerance DOUBLE

the tolerance to use. Line segments whose length is less than the tolerance value will be removed. Specify 0.0 to prevent deleting vertices which are close together.

options SHORT INTEGER

`SIS_CLEAN_LINE_NONE`	delete line segments only if shorter than tolerance
`SIS_CLEAN_LINE_REMOVE_0`	remove vertices in the middle of a straight line section
`SIS_CLEAN_LINE_REMOVE_180`	remove vertices which are causing spikes in the line
`SIS_CLEAN_LINE_REMOVE_SELF`	remove sections of the line between self intersections

Add together the options `SIS_CLEAN_LINE_REMOVE_0`, `SIS_CLEAN_LINE_REMOVE_180` and `SIS_CLEAN_LINE_REMOVE_SELF` to perform several types of cleaning at once.

Example `GisCleanLines "LinesFound", 1, SIS_CLEAN_LINE_NONE`

cleans all the lines in the list, deleting all the segments that have a length shorter that the tolerance

Available ME, MD, OD, ASC

❍ CloseDataset

Close a dataset.

Syntax `GisCloseDataset ( filename )`

Arguments *filename* STRING

the dataset to close

Example `GisCloseDataset "c:\projects\Planning.bds"`

Available MM, ME, MD, OM, OD, ASC

❍ CloseIndexDatasetTile

Close a named dataset tile within an index dataset.

Syntax `GisCloseIndexDatasetTile ( nDataset, tilename )`

Arguments *`nDataset`* LONG INTEGER
the serial number of the index dataset ➲page 222, **Serial numbers**

`tilename` STRING
the tile to close

Example `GisCloseIndexDatasetTile 1,"TQ1234"`

Available MM, ME, MD, OM, OD, ASC

❍ CloseItem

Close the current open item, stopping it being current.

Syntax `GisCloseItem ( )`

Notes If the current open item is a group, the graphics will be locked to the cursor for the user to place with two screen snaps (position and alignment).

Available MM, ME, MD, OV, OM, OD, ASC

❍ CombineFilter

Create a named filter by combining two named filters using a Boolean operation, replacing any existing filter with the same name.

Syntax `GisCombineFilter ( filterOutput, filter1, filter2, mode )`

Arguments *`filterOutput`* STRING
the named filter to create or replace

`filter1` STRING
the first filter to combine or an empty string (see Notes)

`filter2` STRING
the second filter to combine or an empty string (see Notes)

`mode` SHORT INTEGER
the Boolean operation to use.

`SIS_BOOLEAN_AND`	items must be allowed by both filters
`SIS_BOOLEAN_OR`	items must be allowed by at least one of the filters
`SIS_BOOLEAN_XOR`	items must be allowed by exactly one of the filters
`SIS_BOOLEAN_DIFF`	items must be allowed by *`filter1`*, but not by *`filter2`*

Example `GisCombineFilter "Search", "Planning", "Address", SIS_BOOLEAN_ADD`

Notes The new filter works by making and storing a copy of the two old filters. Any changes to the existing filters after calling this method will not affect the new filter.

Either *`filter1`* or *`filter2`* can be empty strings (but not both). An empty string for a filter means no filter (ie no items are excluded). Therefore, by using an empty string for *`filter1`* with the `SIS_BOOLEAN_DIFF` mode the effect of *`filter2`* can be reversed.

Available MM, ME, MD, OM, OD, ASC

❍ CombineLists

Combine two named lists using a Boolean operation, returning the answer in a third named list.

Syntax `GisCombineLists ( listOutput, list1, list2, mode )`

Arguments *listOutput* STRING
the named list resulting from the Boolean operation. The *listOutput* argument can be the same as either *list1* or *list2*, to re-use an existing named list.

list1, list2 STRING
the named lists to combine

mode SHORT INTEGER
the Boolean operation to use

SIS_BOOLEAN_AND	add items which are in both *list1* and *list2*
SIS_BOOLEAN_OR	add items which are in *list1* or *list2*
SIS_BOOLEAN_XOR	add items which are in *list1* or *list2*, but not in both
SIS_BOOLEAN_DIFF	add items which are in *list1* but not in *list2*

Example `GisCombineLists "Combination", "Buildings","Gardens", SIS_BOOLEAN_AND`

Available MM, ME, MD, OM, OD, ASC

❍ CombineLocus

Create a named locus by combining two named loci using a Boolean operation, replacing any existing locus with the same name.

Syntax `GisCombineLocus ( locusOutput, locus1, locus2, mode )`

Arguments *locusOutput* STRING
the named locus to create or replace

locus1 STRING
the first locus to combine or an empty string (see Notes)

locus2 STRING
the second locus to combine or an empty string (see Notes)

mode SHORT INTEGER
the Boolean operation to use:

SIS_BOOLEAN_AND	the items' centroid must be in both locus objects
SIS_BOOLEAN_OR	the items' centroid must be in at least one of the original locus objects
SIS_BOOLEAN_XOR	the items' centroid must be in exactly one of the original locus objects
SIS_BOOLEAN_DIFF	the items' centroid must be in *locus1*, but not in *locus2*

Notes The new locus works by making and storing a copy of the two old locus objects. The new locus will not support any advanced testing modes, but instead will use the testing modes of the two constituent locus objects. Any changes to the existing loci after calling this method will not affect the new locus.

Either `locus1` or `locus2` may be empty strings (but not both). An empty string for a locus means no locus (ie no items are excluded). Therefore, by using an empty string for `locus1` with the `SIS_BOOLEAN_DIFF` mode, the effect of `locus2` can be reversed.

Available ME, MD, OD, ASC

❍ CompactDataset

Discard all undo actions and defragment the memory used by a dataset. The current open item, if any, will be closed, and the current selection list will be emptied before compacting the dataset.

Syntax `GisCompactDataset ( nDataset )`

Arguments `nDataset` LONG INTEGER

the serial number of the dataset to be compacted ➲page 222, **Serial numbers**

Available MM, ME, MD, OV, OM, OD, ASC

❍ Compose

Compose the current window, in preparation for using `PlacePrintTemplate` or `CreatePhoto` on another window.

Syntax `GisCompose ( )`

Available MM, ME, MD, OV, OM, OD, ASC

❍ Copy

Copy the items in a named list to the clipboard, optionally deleting the existing item (Cut instead of Copy).

Syntax `GisCopy ( list, bDelete )`

Arguments `list` STRING

the named list whose items are to be copied

`bDelete` SHORT INTEGER

True	delete the items after copying (ie Cut)
False	leave the existing items behind (ie Copy)

Example `GisCopy "FoundItems", False`

copies all the items in the list `FoundItems` to the clipboard

Available MM, ME, MD, OV, OM, OD

❍ CopyFeatureCode

Copy an existing feature code into the currently loaded feature table. The feature table must be loaded for editing using `LoadFeatureTable`.

Syntax `GisCopyFeatureCode ( fcodeTo, fcodeFrom, ftable )`

Arguments `fcodeTo` SHORT INTEGER
the feature code to be added

`fcodeFrom` SHORT INTEGER
the feature code to be copied

`ftable` STRING
the feature table from which to copy the feature code. Use `""` to copy a feature code in the currently loaded feature table.

Example `GisCopyFeatureCode 1, 1, "Landline"`
copies the feature code 1 from the feature table LandLine into the currently loaded feature table

Available ME, MD, OD, ASC

❍ CopyListItems

Copy the items in a named list to the default overlay.

Syntax `GisCopyListItems ( list )`

Arguments `list` STRING
the named list whose items are to be copied

Available MM, ME, MD, OM, OD, ASC

❍ CopyThemeComponent

Copy an existing theme component into the currently loaded theme. Use `LoadTheme` to load a theme for editing.

Syntax `GisCopyThemeComponent ( componentTo, componentFrom, theme )`

Arguments `componentFrom` SHORT INTEGER
the component to copy into

`componentTo` SHORT INTEGER
the component to copy from

`theme` STRING
the theme from which to copy the component. Use `""` to copy a component in the currently loaded theme.

Example `GisCopyThemeComponent 1, 1, "Population"`

Notes Several types of theme consist of several components, eg blocks in a Bar Charts theme, slices in a Pie Charts theme, and so on. Each of these components has its own properties. Theme component properties are set and queried using the `SIS_OT_THEMECOMPONENT` constant. Theme component indices run from zero to one less than the number of theme components.

Available MM, ME, MD, OM, OD, ASC

❍ CreateAreaFromLines

Create an area item(s) from the line item(s) in a named list, optionally deleting the line items after creating the area item(s).

Syntax `GisCreateAreaFromLines ( list,bDelete,createOption )`

Arguments *list* STRING

the named list containing the line items to be used in the area creation. If the method succeeds, the named list will contain the area items created.

bDelete SHORT INTEGER

True delete the line items after area creation

False leave the existing line items behind

createOption SHORT INTEGER

SIS_AREA_ONE_TO_ONE	create an area for every line in the named list
SIS_AREA_MANY_TO_ONE	create one area using all of the line items in the named list
SIS_AREA_DISJOINT	create disjoint area items where a line in the list which is wholly contained within another line in the list will become a hole, or island, within the containing line

Example `GisCreateAreaFromLines "ListFound", True, SIS_AREA_MANY_TO_ONE`

creates an area item for each line item in the list ListFound

Available MM, ME, MD, OM, OD, ASC

❍ CreateAssembly

Create an assembly item from the items in a named list.

Syntax `GisCreateAssembly ( list, x, y, z, shape, a, s )`

Arguments *list* STRING

the named list containing the items to be added to the assembly

x, y, z DOUBLE

the position of the assembly

shape STRING

the shape, or symbol, to be used at the hook point of the assembly

a, s DOUBLE

the angle, in radians, and scale of the assembly

Example `GisCreateAssembly "Lines", 123, 5, 237, 0, "GXF.005", 1,273, 1250`

Notes If a group is open, graphics are added to the group, otherwise a new item is created.

Available ME, MD, OD, ASC

❍ CreateBackdropOverlay

Create a new overlay which uses a named item as a backdrop.

Syntax `GisCreateBackDropOverlay ( pos, backdrop )`

Arguments *pos* SHORT INTEGER

the position in the overlays list at which to insert the overlay. If this argument specifies a position in the existing overlays, the new overlay will not replace the existing overlay at the given position, but will shuffle any other overlays down the list.

backdrop STRING
the named item to use as a backdrop

Example `GisCreateBackDropOverlay 0 "GB National Grid"`

Available MM, ME, MD, OV, OM, OD, ASC

❍ CreateBarTheme

Create a new Bar Chart theme. After editing the theme properties, use `StoreTheme` to save the theme in a named object library.

Syntax `GisCreateBarTheme ( nBlock )`

Arguments *nBlocks* SHORT INTEGER
the number of blocks in the Bar Chart, in the range 1 to 256

Available MM, ME, MD, OV, OM, OD, ASC

❍ CreateBds

Create an empty BDS (Base DataSet) file. This call will fail if the BDS file already exists.

Syntax `GisCreateBds ( filename )`

Arguments *filename* STRING
the name of the new BDS file. The new dataset is created in the current attached directory, unless a full path name is given. The new dataset can be added to the current SWD using `InsertDataset`.

Available MM, ME, MD, OM, OD, ASC

❍ CreateBitmap

Create a bitmap item.

Syntax `GisCreateBitmap ( filename, x1, y1, x2, y2, bLinked, bStretch )`

Arguments *filename* STRING
the name of the bitmap to be opened

x1, y1, x2, y2 DOUBLE
the extents within which the bitmap item will be placed

bLinked SHORT INTEGER

True create a link to the filename

False copy the contents of the filename

bStretch SHORT INTEGER

True stretch the bitmap item to fill the whole of the given extents

False maintain the aspect ratio of the bitmap item within the given extents

Example `GisCreateBitmap ("c:\temp\bitmap.bmp", 10, 10, 20, 20, False, False)`

Notes This API method respects the axes angle setting: the x, y, z values are interpreted within the axes and all new items created will align to the axes angle. If a group is open, graphics are added to the group, otherwise a new item is created.

Available MM, ME, MD, OM, OD, ASC

Example
```
GisCreateBitmap "d:data\photo.bmp", 100, 100, 500, 500, True, True
```

❍ CreateBlock

Create a named block from the items in a named list, replacing any existing block of the same name.

Syntax
```
GisCreateBlock ( list, blk, x, y, z )
```

Arguments

list STRING
the named list containing the items to be inserted into the block

blk STRING
the named block to create or replace

x, y, z DOUBLE
the hook point of the block

Example
```
GisCreateBlock "BlockList", NewBlockName", 10, 10, 0
```

Available ME, MD, OD, ASC

❍ CreateBoolean

Create an area item by combining existing area items. This method uses only area, QZone, and polygon items. The type of item created is dependent on the type of the items being combined. If a group is open, graphics are added to the group, otherwise a new item is created.

Syntax
```
GisCreateBoolean ( list, boolop )
```

Arguments

list STRING
the named list containing the area items to be combined

boolop SHORT INTEGER
➲page221, **Boolean tests**

Example
```
GisCreateBoolean "Overlap Areas", SIS_BOOLEAN_AND
```
creates a new item which is the result of the overlapping items in the list Overlap Areas

Available ME, MD, OM, ASC

❍ CreateBoundary

Create an item from the boundary of the current open item.

Syntax
```
GisCreateBoundary ( )
```

Notes The boundary of an item is always one dimension less than the item itself:

- the boundary of an area item, which is two-dimensional, is a line item
- the boundary of a non-closed line item, which is one-dimensional, is a multi-point item made up of two points (the start and end point of the line)

- the boundary of a point item, which has no dimensions, is nothing

If a group is open, graphics are added to the group, otherwise a new item is created.

Available ME, MD, OD, ASC

❍ CreateBoxLabel

Create a special label item which has a line pointing to a labelled location. Label text is like box text, as it is created in real world units, and when printed maintains its actual proportions to the surrounding graphics. This method respects the axes angle setting. This means that the x, y, and z values are interpreted within the axes and all new items creates will align to the axes angle.

Syntax `GisCreateBoxLabel ( x1, y1, z1, h, text, x2, y2, z2 )`

Arguments

x1, y1, z1 DOUBLE
the hook point of the label item

h DOUBLE
the height in metres of the label item

text STRING
the text of the label item

x2, y2, z2 DOUBLE
the position to draw the label line to

Example `GisCreateBoxLabel 10, 10, 0, 3, "Box Label Text", 20, 20, 0`

Notes If a group is open, graphics are added to the group, otherwise a new item is created.

Available MM, ME, MD, OM, OD, ASC

❍ CreateBoxText

Create a box text item in real world units.

Syntax `GisCreateBoxText ( x, y, z, h, text )`

Arguments

x, y, z DOUBLE
the hook point of the box text item

h DOUBLE
the height in metres of the box text item

text STRING
the text of the box text item

Example `GisCreateBoxText 10, 10, 0, 5, "BoxText"`

Notes When printed it maintains its actual proportions to the surrounding graphics. This API method respects the axes angle setting. This means that the x, y, z are interpreted within the axes and all new items created will align to the axes angle. If a group is open, graphics are added to the group, otherwise a new item is created.

Available MM, ME, MD, OM, OD, ASC

❍ CreateBufferFromItems

Create an area or QZone item surrounding the items in a named list.

Syntax `GisCreateBufferFromItems ( list, radius, resolution )`

Arguments *list* STRING
the named list containing the items around which the area or QZone will be created

radius DOUBLE
the buffer radius around each item

resolution DOUBLE
the resolution in metres of the QZone. Use 0.0 for a smooth area item, or a positive number for a QZone item.

Example `GisCreateBufferFromItems "Conservation", 100, 1`

Available ME, MD, OD, ASC

❍ CreateBufferLocusFromItems

Create a named buffer locus surrounding items in a named list. It replaces any existing locus with the same name.

Syntax `GisCreateBufferLocusFromItems ( list, bDelete, locus, radius, resolution )`

Arguments *list* STRING
the named list containing the items around which the locus will be created

bDelete SHORT INTEGER

True delete the items after locus creation

False leave the existing items behind

locus STRING
the named locus to create or replace

radius DOUBLE
the buffer radius around each item

resolution DOUBLE
the resolution in metres of the locus. Use 0.0 for a smooth area locus or a positive number for a QZone locus.

Example `GisCreateBufferLocusFromItems "Conservation", False, "Conservation",100,1`

Available ME, MD, OD, ASC

❍ CreateCircle

Create a circular area item.

Syntax `GisCreateCircle ( x, y, z, r )`

Arguments *x, y, z* DOUBLE
the centre of the circle

r DOUBLE
the radius of the circle (in metres)

Notes If a group is open, graphics are added to the group, otherwise a new item is created.

Available MM, ME, MD, OM, OD, ASC

❍ CreateCircleLocus

Create a named circular locus, replacing any existing locus with the same name.

Syntax `GisCreateCircleLocus ( locus, x, y, radius )`

Arguments *locus* STRING
the named locus to create or replace

x, y DOUBLE
the centre of the circle locus

radius DOUBLE
the radius of the circle locus

Example `GisCreateCircleLocus "RoadScheme", 106723, 187222, 500`

Available ME, MD, OD, ASC

❍ CreateClassTreeFilter

Create a named class tree filter, a filter based on the class of item replacing any existing filter with the same name.

Syntax `GisCreateClassTreeFilter ( filter, formula )`

Arguments *filter* STRING
the named filter to create or replace

formula STRING
a class tree formula of the form *–Class +SubClass1 –SubClass2 +SubClass3*. For example, `-Item +Line +Area` will match only line and area items, and their sub-classes.

Example `GisCreateClassTreeFilter "NotPolygons", "+Item -SeedArea"`
creates a filter for all items excluding polygons

Notes The item class names in the formula argument, which reflect the Cadcorp SIS C++ class names, are not necessarily the same as those that appear in the Cadcorp SIS user interface, which are translatable. In particular, the polygon and chain item classes should be specified as SeedArea and SeedChain respectively. The class name to use in this method is stored in the `_class$` item property. The translatable class name is stored in the `_classLocal$` property.

Available MM, ME, MD, OM, OD, ASC

❍ CreateCombinedFilter

Create a named combined class/property filter, a filter that combines the class of an item and a property formula. It replaces any existing filter with the same name.

Syntax `GisCreateCombinedFilter ( filter )`

Arguments *filter* STRING
the named filter to create or replace

Notes Use `SetCombineFilterClause` to set the value of this filter.

Available MM, ME, MD, OM, OD, ASC

❍ CreateContourTheme

Create a new Contour Theme. After editing the theme properties, use `StoreTheme` to save the theme in a named object library.

Syntax `GisCreateCountourTheme ( hMajor, hMinor )`

Arguments *hMajor, hMinor* DOUBLE
the height of the major and minor axes

Example `GisCreateCountourTheme 100, 50`

Available MD, OD, ASC

❍ CreateConvexHull

Create the smallest possible item with convex geometry, which contains the current open item.

Syntax `GisCreateConvexHull ( )`

Notes If a group is open, graphics are added to the group, otherwise a new item is created.

Available ME, MD, OD, ASC

❍ CreateDataSourceOverlay

Insert a dataset into the current SWD, which will fetch data from non-file data source.

Syntax `GisCreateDataSourceOverlay ( pos, clsDataSources, params )`

Arguments *pos* SHORT INTEGER
the position in the overlays list at which to insert the overlay. If this argument specifies a position in the existing overlays, the new overlay will not replace the existing overlay at the given position, but will shuffle any other overlays down the list.

clsDataSource STRING
the classname of the data source to use

params STRING
optional parameters used to configure the data source

Notes For a data source using the Oracle Spatial Object-Relational model, the *clsDataSource* argument should be `"OracleSpatialDataset"`. The *params* argument contains the connection string and SQL WHERE clause information to the Oracle server in the following format:

`"server = xxxx,user = xxx,password = xxx ,where = ""column_name = value"" "`

An optional WHERE clause can be added to retrieve partial datasets.

Example

```
Dim sConnection As String
' set connection and WHERE clause parameters
sConnection = "server = sis,User=me,password=secret, _
  layer= TEST_HAMPS_MERG,where = ""parcel_ref = '15/043/0013_SU27363183'"""

' Method call
GisCreateDataSourceOverlay 0, "OracleSpatialDataset", sConnection
```

Available MM, ME, MD, OV, OM, OD, ASC

❍ CreateDbBlobOverlay

Create an overlay which views Blobs stored in a database.

Syntax

```
GisCreateDbBlobOverlay ( pos, prj, rs, fmt, nfBlob, nfId, nfVer, nfSr,
nfSlMin, nfSlMax, span )
```

Arguments

pos SHORT INTEGER

the position number in the overlays list to insert the overlay. If this argument specifies a position in the existing overlays, the new overlay will not replace the existing overlay at the given position, but will shuffle any other overlays down the list.

prj STRING

the named projection of the stored item Blobs

rs STRING

a named recordset previously created using `DefineRecordset`

fmt SHORT INTEGER

the format of the item Blob strings:

`SIS_BLOB_SIS`	Cadcorp SIS format
`SIS_BLOB_OGIS_WKB`	OpenGIS Well-Known-Binary format
`SIS_BLOB_OGIS_WKT`	OpenGIS Well-known-Text format

nfBlob SHORT INTEGER

the index in the recordset columns argument of the item Blob string column. This column must exist. Column indices start from 0.

nfId SHORT INTEGER

the index in the recordset columns argument of the item ID column. A value of –1 will make Cadcorp SIS generate the item IDs automatically.

nfVer SHORT INTEGER

the index in the recordset columns argument of the item version column. A value of –1 indicates that no version information is being supplied. Column indices start from 0.

nfSr SHORT INTEGER

the index in the recordset columns argument of the item spatial reference column. A value of –1 indicates that no spatial reference is being supplied. Column indices start from 0.

nfSlMin SHORT INTEGER

the index in the recordset columns argument of the item minimum scale threshold column. A value of –1 indicates that no minimum scale threshold is being supplied.

This parameter is currently ignored. Use –1 for future compatibility.

nfSlMax SHORT INTEGER

The index in the recordset columns argument of the item maximum scale threshold column. A value of –1 indicates that no maximum scale threshold is being supplied.

This parameter is currently ignored. Use –1 for future compatibility.

span DOUBLE

the span used in the spatial reference (see `GetSpatialReference`)

Example

```
GisCreateDbBlobOverlay 0, "*APrjNatGrid", "rsBlobs", SIS_BLOB_SIS, 3, 0, _
  1, 2, -1, -1, 2000000
```

creates a View Blobs overlay at position 0 in the overlays list.

`SIS_BLOB_SIS`	a constant defined in GisLink.bas (and SisConst.bas)
`3`	the binary data (the graphics) are in column 3 of the table
`0`	the item ID is in column 0 of the table
`1`	the item version is in column 1 of the table
`2`	the spatial reference is in column 2 of the table
`-1`	(max and min scale thresholds cannot be set in the current release)
`2000000`	The spatial reference string encodes a position and a radius which together describe an extents circle. The span used when calculating a spatial reference must be big enough to cover all the possible co-ordinates. A smaller span will give spatial references with a finer resolution. The spatial reference does not affect the accuracy or resolution of the positions of the item it is associated with, only the accuracy of whether or not the item is loaded in a particular view. The worst that can happen with a coarse resolution is that extra items are loaded.

When creating an overlay in this way, it is essential that you know the structure of the database table, so you can set the correct arguments for the `DefineRecordset` and `CreateDbBlobOverlay` methods.

Available MM, ME, MD, OM, OD, ASC

❍ CreateDbOverlay

Create an overlay which stores editable Blobs in a database.

Syntax

```
GisCreateDbOverlay ( pos, dialect, bTransact, connect, tableItem,
prj, span )
```

Arguments

pos SHORT INTEGER

the position in the overlays list at which to insert the overlay. If this argument specifies a position in the existing overlays, the new overlay will not replace the existing overlay at the given position, but will shuffle any other overlays down the list.

dialect SHORT INTEGER

the dialect to use. This is currently ignored, but may be used in future.

bTransact SHORT INTEGER

True use short transactions for updating linked items (eg topology)

False do not use transactions

connect STRING

the connection string ➲page 222, **Connecting to databases**

tableItem STRING

the name of the table containing item information, ie Blob, spatial reference, etc. Many different tables may be used for different datasets within a single database.

prj STRING
the named projection of the stored item Blobs

span DOUBLE
the span used in the spatial reference (see ➲page 154, **GetSpatialReference**)

Example `GisCreateDBOverlay 1, 0, True, "", "BlobTable", "*AprjNatGrid", 2000000`
creates a DBOverlay at position 1 in the current list of overlays, which uses short transactions and prompts for connection strings, storing items in a table called BlobTable using *AprjNatGrid projection in a 2 000 000m span

Available MM, ME, MD, OM, OD, ASC

❍ CreateDbPointOverlay

Create an overlay which views records stored in a database provided each record has two fields containing X and Y co-ordinates.

Syntax
```
GisCreateDbPointOverlay ( pos, prj, rs, aclass, nfx, nfy, nfld, nfSr,
nfSIMin, nfSLMax, span )
```

Arguments

pos SHORT INTEGER
the position in the overlays list at which to insert the overlay. If this argument specifies a position in the existing overlays, the new overlay will not replace the existing overlay at the given position, but will shuffle any other overlays down the list.

prj STRING
the named projection of the stored point co-ordinates

rs STRING
a named recordset previously created using `DefineRecordset`

aclass STRING
the class of item to create: point or text. Text items can be created by aliasing any string column in the named recordset to `_text$`. Each ttext item created will then get its text from the aliased column.

nfX SHORT INTEGER
the index in the recordset columns argument of the x co-ordinate column. This column must exist. Column indices start from 0.

nfY SHORT INTEGER
the index in the recordset columns argument of the y co-ordinate column. This column must exist. Column indices start from 0.

nfId SHORT INTEGER
the index in the recordset columns argument of the item ID column. A value of –1 makes Cadcorp SIS generate the item IDs automatically. Column indices start from 0.

nfSr SHORT INTEGER
the index in the recordset columns argument of the item spatial reference column. A value of –1 indicates that no spatial reference is being supplied. Column indices start from 0.

nfSlMin SHORT INTEGER
the index in the recordset columns argument of the item minimum scale threshold column. A value of –1 indicates that no minimum scale threshold is being supplied. (This parameter is currently ignored, use –1 for future compatibility.)

nfSlMax SHORT INTEGER
the index in the recordset columns argument of the item maximum scale threshold column. A value of –1 indicates that no maximum scale threshold is being supplied. (This parameter is currently ignored, use –1 for future compatibility.)

span DOUBLE
the span used in the spatial reference (➲page 154, **GetSpatialReference**)

Example
```
GisCreateDBPointOverlay 1, "*APrjNatGrid", "Planning", "Point", 1, 2, 3, _
  -1, -1, -1, 2000000
```
creates an overlay at position 1 on the overlays list, in the given projection, using recordset Planning, display as points, X index is 1, Y index is 2 (positions of the columns containing co-ordinate information in the column argument used to define the recordset), ID index is 3, no spatial reference, minimum and maximum view scale, within a span of 2 000 000m

Available MM, ME, MD, OM, OD, ASC

❍ CreateDbTable

Create a named table which views data from a database, replacing any existing table with the same name. Named tables are stored only in project workspace (Cadcorp SIS) files, not in named object library (NOL) files.

Syntax
```
GisCreateDbTable ( table, rs, linkfields, bReadOnly )
```

Arguments

table STRING
the named table to create or replace

rs STRING
a named recordset previously created using `DefineRecordset`. Use an empty string to delete any existing table of this name (the rest of the arguments are ignored).

linkfields STRING
a comma-delimited list of columns which specify how items are 'joined' to the table. Each of the columns must have been included in the previously defined recordset. This argument has been superseded by extra arguments to the `Table(`*`table, columns, props`*`)` formula, and should be used only for backwards compatibility.

bReadOnly SHORT INTEGER
this argument is ignored at present, and should be set to True

Example
```
GisCreateDBTable "Planning_Records", "Planning", "ID", True
```

Available MM, ME, MD, OM, OD, ASC

❍ CreateDisplacement

Create a displacement item, prior to doing a rubber sheet operation.

Syntax
```
GisCreateDisplacement ( x1, y1, z1, x2, y2, z2 )
```

Arguments

x1, y1, z1 DOUBLE
the position to displace from

x2, y2, z2 DOUBLE
the position to displace to

Available ME, MD, OD, ASC

❍ CreateDotTheme

Create a new Dot Density theme, based on a formula. After editing the theme properties, use `StoreTheme` to save the theme in a named object library.

Syntax `GisCreateDotTheme ( formula )`

Arguments *formula* STRING
the formula to use for calculating dot densities

Example `GisCreateDotTheme "(R0to9&/Presid&)*100"`
display dots representing the percentage of children age 0 to 9 who were resident in each of the census areas

Available MM, ME, MD, OV, OM, OD, ASC

❍ CreateDoubleBoolean

Execute a combination of Boolean operations.

Syntax `GisCreateDoubleBoolean ( list1, boolop1, list2, boolop2, boolop3 )`

Available *list1* STRING
the named list for the first Boolean operation

boolop1 SHORT INTEGER
the Boolean operator for the first Boolean operation

list2 STRING
the named list for the second Boolean operation

boolop2 SHORT INTEGER
the Boolean operator for the second Boolean operation

boolop3 SHORT INTEGER
the Boolean operator for the third Boolean operation, which operates on the two items resulting from the first two Boolean operations

Example
```
GisCreateDoubleBoolean "Land", SIS_BOOLEAN_AND, "Road", SIS_BOOLEAN_AND,
SIS_BOOLEAN_OR
```
creates an item which is the result of all the common parts in Land and the common parts in Road added together

Notes The combination of operations is similar to the following sequence of calls, but will handle any failures in the intermediate steps:
```
EmptyList "listTemp"
CreateBoolean list1, boolop1
AddToList "listTemp"
CreateBoolean list2, boolop2
AddToList "listTemp"
CreateBoolean "listTemp", boolop3
OpenList "listTemp", 0
DeleteItem
OpenList "listTemp", 1
DeleteItem
EmptyList "listTemp"
```
➲page221, **Boolean tests** for valid values for *boolop1*, *boolop2*, and *boolop3*

Available ME, MD, OM, ASC

❍ CreateDrapeBitmap

Create a named bitmap item from the current view, which is suitable for draping in the 3D window. This item will be saved into the current named object library.

Syntax `GisCreateDrapeBitmap ( name )`

Arguments *namestring* STRING
the name of the bitmap item to be created

Example `GisCreateDrapeBitmap ("AerialPhoto")`
saves the currently displayed aerial photograph as a bitmap item in the current named object library. The view to be saved may contain raster and/or vector data.

Notes The bitmap item may be draped over a 3D surface in the 3D window using the `DrapeBitmap` method.

Available MD, OD, ASC

❍ CreateEllipse

Create an elliptical area item. This API method respects the axes angle setting. This means that the x, y, and z values are interpreted within the axes and all new items created will align to the axes angle.

Syntax `GisCreateEllipse ( x1, y1, x2, y2 )`

Arguments *x1, y1, x2, y2* DOUBLE
the rectangular extents of the ellipse

Notes If a group is open, graphics are added to the group, otherwise a new item is created.

Available MM, ME, MD, OV, OM, OD, ASC

❍ CreateExtrudeTheme

Create a new Extrude theme, based on a formula. After editing the theme properties, use `StoreTheme` to save the theme in a named object library.

Syntax `GisCreateExtrudeTheme ( formula )`

Arguments *formula* STRING
the formula to use when evaluating the extrusion height

Example `GisCreateExtrudeTheme "(R0to9&/Presid&)*100"`
apply an extrusion factor to each census area, representing the percentage of children aged 0 to 9 who were resident. The extruded areas can be viewed in the 3D window.

Available MD, OD, ASC

❍ CreateExtrusion

Create a surface item by extruding current open area or line items.

Syntax `GisCreateExtrusion ( height )`

Arguments *height* DOUBLE
the height to which to extrude the current open area or line item (in metres)

Notes If a group is open, graphics are added to the group, otherwise a new item is created.

Available MD, OD, ASC

❍ CreateFeatureFilter

Create a named filter based on a named feature table, replacing any existing filter with the same name.

Syntax `GisCreateFeatureFilter ( filter, ftable )`

Arguments *`filter`* STRING
the named filter to create or replace

`ftable` STRING
the named feature table on which to base the feature filter

Example `GisCreateFeatureFilter "MyLand-Line", "Feature Filter.Land-Line"`
creates a feature filter called MyLand-Line based on the existing filter Feature Filter.Land-Line
`GisCreateFeatureFilter "My-Land-Line", "MyLandLineTable"`
creates a feature filter based on the contents of the feature table MyLandLineTable

Available MM, ME, MD, OV, OM, OD, ASC

❍ CreateFlowTheme

Create a new Flow theme, based on the gradient of surfaces.

Syntax `GisCreateFlowTheme ( )`

Notes After editing the theme properties, use `StoreTheme` to save the theme in a named object library.

Available MD, OD, ASC

❍ CreateFormulaGrid

Create a grid item by combining named grid items using a formula.

Syntax `GisCreateFormulaGrid ( formula )`

Arguments *`formula`* STRING
the formulae involving grid items which have previously been saved in a named object library. For example, if grid items have been saved under the names g1 and g2, a new grid item can be created by using the formula `g1 + g2`.

Available MD, OD, ASC

❍ CreateGraduatedTheme

Create a new Graduated theme. After editing the theme properties, use `StoreTheme` to save the theme in a named object library.

Syntax `GisCreateGraduatedTheme ( formula )`

Arguments *`formula`* STRING
the formula to use when evaluating the symbol height, which must evaluate to a number

Available MM, ME, MD, OV, OM, OD, ASC

❍ CreateGraticule

Create a graticule item using the current open photo item.

Syntax `GisCreateGraticule ( x1,y1, x2, y2 )`

Arguments *x1, y1, x2, y2* DOUBLE
the rectangular extents of the graticule. This will typically be the extents of the associated photo item.

Notes This method respects the axes angle setting. This means that the x, y, and z values are interpreted within the axes and all new items created will align to the axes angle.

Available MM, ME, MD, OM, OD, ASC

❍ CreateGridFromQZone

Create a grid item from current open QZone items.

Syntax `GisCreateGridFromQZone ( )`

Available MD, OD, ASC

❍ CreateGroup

Create an empty group item using a previously registered group class. All graphics created after calling this function but before calling `CloseItem`, `PlaceGroup`, `Release` or `UpdateItem` will be part of this group.

Syntax `GisCreateGroup ( groupType )`

Arguments *groupType* STRING
the type, or class, of group to create

Example `GisCreateGroup "Station"`

Notes Before this command is used, the group class must be registered using `RegisterGroupType`, which is typically called only once, at the start of an application.

If an empty string is given, the group will be automatically exploded when placed, leaving the component items ungrouped.

Available MM, ME, MD, OM, OD, ASC

❍ CreateGroupFromItems

Create a group item from the items in a named list, optionally deleting the items in the named list. The current axes will be used as the hook point.

Syntax `GisCreateGroupFromItems ( list, bDelete, groupType )`

Arguments *list* STRING
the named list containing the items to be grouped

bDelete SHORT INTEGER

True delete the items after grouping

False leave the existing items behind

groupType STRING

the type, or class, of group to create; the group class registered using `RegisterGroupType`

Example `GisCreateGroupfromItems "StationList", True, "Station"`

creates a group made up from the items in the list StationList. These items are then deleted.

Available MM, ME, MD, OM, OD, ASC

❍ CreateIndexCoverage

Create tile items covering extents, using one of the standard index dataset naming conventions.

Syntax `GisCreateIndexCoverage ( list, tilename, namer, x1, y1, x2, y2 )`

Arguments *list* STRING

the named list in which to store any items created

tilename STRING

the path name of one of the tiles being indexed

namer STRING

the file naming convention to use. This parameter is needed only if the tilename is not sufficient for Cadcorp SIS to unambiguously determine the tile's naming convention. ➲Chapter 6:**"Index dataset naming conventions",Introduction**,page419 for the available options

x1, y1, x2, y2 DOUBLE

the extents of the coverage

Example

```
GisCreateIndexCoverage "Tiles", "c:\data\tfSS7893.ntf", _
  "ANtfNamer", 277000, 184000, 282000, 189000
```

Available MM, ME, MD, OM, OD, ASC

❍ CreateIndexOverlay

Create an index overlay, optionally creating gateway items for each tile found.

Syntax `GisCreateIndexOverlay ( pos, tilename, namer, flag )`

Arguments *pos* SHORT INTEGER

the position in the overlays list at which to insert the overlay. If this argument specifies a position in the existing overlays, the new overlay will not replace the existing overlay at the given position, but will shuffle any other overlays down the list.

tilename STRING

the path name of one of the tiles being indexed

namer STRING

the tile naming convention to be used. This parameter is needed only if the tilename is not sufficient for Cadcorp SIS to unambiguously determine the tile's naming conven-

tion. ➲Chapter 6: **"Index dataset naming conventions", Introduction**, page 419 for the available options

flags SHORT INTEGER

`SIS_INDEX_OUTLINES`	draw the outline of each tile found
`SIS_INDEX_LABELS`	label each tile found with the tile name
`SIS_INDEX_PYRAMID`	make the naming convention find related tiles in a pyramid

You can add `SIS_INDEX_OUTLINES`, `SIS_INDEX_LABELS`, and `SIS_INDEX_PYRAMID` together.

Example

```
GisCreateIndexOverlay 1, "c:\data\tfSS7887SW.ntf", "ANtfNamer", _
  SIS_INDEX_PYRAMID + SIS_INDEX_OUTLINES + SIS_INDEX_LABELS
```

inserts a pyramid indexed overlay, in position 1, displaying gateways and labels

Available MM, ME, MD, OV, OM, OD, ASC

❍ CreateIndividualTheme

Create a new Individual Values theme. After editing the theme properties, use `StoreTheme` to save the theme in a named object library.

Syntax

```
GisCreateIndividualTheme ( formula, nValues )
```

Arguments

formula STRING
the formula to use for matching values

nValues SHORT INTEGER
the number of individual values in the theme, in the range 2 to 16 383

Example

```
GisCreateIndividualTheme "Ppres91&<50", 256
```

Available MM, ME, MD, OM, OD, ASC

❍ CreateInsert

Create an insert item using a named block item.

Syntax

```
GisCreateInsert ( x, y, z, blk, a, s )
```

Arguments

x, y, z DOUBLE
the position of the insert item

blk STRING
the block item to which the insert item refers

a, s DOUBLE
the angle, in radians, and scale of the insert item

Example

```
GisCreateInsert 10, 10, 0, "Circle", 1.273, 1250
```

Available MM, ME, MD, OM, OD, ASC

❍ CreateInternalOverlay

Create an internal overlay.

Syntax

```
GisCreateInternalOverlay ( overlay, pos )
```

Arguments *overlay* STRING
the name of the internal overlay

pos SHORT INTEGER
the position in the overlays list at which to insert the overlay. If this argument specifies a position in the existing overlays, the new overlay will not replace the existing overlay at the given position, but will shuffle any other overlays down the list.

Example
```
GisCreateInternalOverlay "Bridges", 5
```

Available MM, ME, MD, OV, OM, OD, ASC

❍ CreateIsoRoute

Create a multi-line item (or a line item) covering all connected places which can be reached from a position, within a given cost. When the cost is related to time, this query is often called an isochrone.

Syntax
```
GisCreateIsoRoute ( list, x, y, z, r, isoVal, formula, filter,
locusNoGo )
```

Arguments *x, y, z* DOUBLE
the position to start from

r DOUBLE
the maximum distance from the start point to linear geometry. The topological algorithm will spread out from the closest item found. The distance from the point to the closest item is not included in the cost calculation. Ideally, the start point should be on an item.

isoVal DOUBLE
the maximum cost to incur during route finding

formula STRING
the formula, or simple property, to use in the route finding calculation as the 'cost' of a link item. For example, using the simple property Length will find the shortest route, and using the formula `_length#/Speed#` will, if each link has a user-defined `Speed#` property, find the quickest route. Any formula can be used, although, if a string formula is used, it must be a string representation of a numeric value.

filter STRING
optionally specify a named filter, which all items must pass to be considered as part of a route

locusNoGo STRING
optionally specify a named locus through which no route may pass. The named locus used will normally have its testing mode set to exclude any items which cross it, using a call similar to the following:
```
CreateLocusFromItem("locus",SIS_GT_INTERSECT,SIS_GM_GEOMETRY)
```

Example
```
GisCreateIsoRoute "Routes", 2000, 1500, 10, 30, _
  "_length#/((30*5280/3.2808)/60)", "links", "NoGo"
```

Notes This method can find a route over any geometry, not just link/node topology, by specifying an empty string for the formula argument. In this case, length is used as the cost.

Available ME, MD, OD, ASC

❍ CreateItem

Create an item from a Blob string. If a group is open, graphics are added to the group, otherwise a new item is created.

Syntax `GisCreateItem ( blob, prj, fmt )`

Arguments

blob STRING
the stored item Blob string

prj STRING
the named projection of the stored item Blob

fmt SHORT INTEGER
the format of the stored item Blob

`SIS_BLOB_SIS`	Cadcorp SIS format
`SIS_BLOB_OGIS_WKT`	OpenGis Well-known-Text format

Example `GisCreateItem blob, "*APrjNatGrid", SIS_BLOB_SIS`
creates an item on the current overlay using the value of blob

Notes When modifying an item which is stored as a blob in a database, it is good practice to create a temporary version of the item on an editable overlay, and only commit the changes to the database, having got the new Blob string using `GetBlob`, when the user is satisfied with the changes. This allows any changes to other columns in the database table to be done in the same transaction.

Available MM, ME, MD, OV, OM, OD, ASC

❍ CreateItemB

Create an item from a Blob data.

Syntax `GisCreateItem ( blob, projection, fmt )`

Arguments

blob BINARY LARGE OBJECT
the stored item Blob data, stored in an array of bytes or in a stream

projection STRING
the named projection of the stored item Blob

fmt SHORT INTEGER
the format of the stored item Blob:

`SIS_BLOB_SIS`	Cadcorp SIS format
`SIS_BLOB_OGIS_WKB`	OpenGis Well-Known Binary format
`SIS_BLOB_OGIS_WKT`	OpenGIS Well-Known-Text format

Example `GisCreateItem blob, "*APrjNatGrid", SIS_BLOB_SIS`
creates an item on the current overlay using the value of blob

Notes If a group is open, graphics are added to the group, otherwise a new item is created. When modifying an item which is stored as a Blob in a database, it is good practice to create a temporary version of the item on an editable overlay, and only commit the changes to the database, having got the new Blob string using `GetBlob`, when the user is satisfied with the changes.

This allows any changes to other columns in the database table to be done in the same transaction.

Available OM, OD, ASC

❍ CreateItemFromLocus

Create an item from a named locus.

Syntax `GisCreateItemFromLocus ( locus )`

Arguments *locus* STRING
the named locus to use

Notes If a group is open, graphics are added to the group, otherwise a new item is created.

Available ME, MD, OD, ASC

❍ CreateKeyMap

Create a keymap item. This API method respects the axes angle setting, so the x, y, and z values are interpreted within the axes, and all new items created will align to the axes angle.

Syntax `GisCreateKeyMap ( x1, y1, x2, y2, list, backdrop, view )`

Arguments *x1, y1, x2, y2* DOUBLE
the rectangular extents of the keymap

list STRING
the named list containing the photo items to be associated with the key map

backdrop STRING
the named item to draw as a backdrop in the key map

view STRING
the named view to draw in the key map. If the key map does not have a named view, it will draw the backdrop around the extents of all of the associated photo items.

Example
```
GisCreateKeyMap 22051, 333313, 516257, 662285, "Photo", _
  "GB National Grid", "*KM.UK"
```

Available MM, ME, MD, OM, OD, ASC

❍ CreateLabelTheme

Create a new Label theme. After editing the theme properties, use `StoreTheme` to save the theme in a named object library.

Syntax `GisCreateLabelTheme ( formula )`

Arguments *formula* STRING
the formula to use

Example `GisCreateLabelTheme "_area#"`

Available MM, ME, MD, OV, OM, OD, ASC

❍ CreateLineText

Create a line text item using the current open line item.

Syntax `GisCreateLineText ( text )`

Arguments *`text`* STRING
the text of the line text item

Notes If a group is open, graphics are added to the group, otherwise a new item is created.
To adjust appearance of text, alignment, or spacing, modify the appropriate item property.

Available MM, ME, MD, OM, OD, ASC

❍ CreateLinkFilter

Create an empty named link filter, replacing any existing filter with the same name.

Syntax `GisCreateLinkFilter ( filter, idlist )`

Arguments *`filter`* STRING
the named filter to create or replace

`idlist` STRING
a space-separated list of item ID numbers to include in the link filter

Example `GisCreateLinkFilter "ItemIds", 1 6 23`

Available MM, ME, MD, OM, OD, ASC

❍ CreateListFromOverlay

Create a named list of all of the items on a given overlay.

Syntax `GisCreateListFromOverlay ( pos, list )`

Arguments *`pos`* SHORT INTEGER
the position in the overlays list of the overlay from which to fill the named list

`list` STRING
the named list to fill with overlay items

Notes If you want only those items which match a filter, use `ScanOverlay`.

Available MM, ME, MD, OV, OM, OD, ASC

❍ CreateListFromSelection

Create a named list of the currently selected items.

Syntax `GisCreateListFromSelection ( list )`

Arguments *`list`* STRING
the named list to fill with the currently selected items

Available MM, ME, MD, OM, OD, ASC

❍ CreateLocusFromItem

Create a named locus in a named object library from the current open item, replacing any existing locus with the same name.

Syntax `GisCreateLocusFromItem ( locus, geomTest, geoMode )`

Arguments *locus* STRING
the named locus to create or replace

geomTest LONG INTEGER
the geometry test to use ➲page219, **Geometry tests**

geomMode SHORT INTEGER
the geometry test mode to use:

`SIS_GM_ORIGIN`	items whose origin (always a single point) must pass the testing method with the selected item
`SIS_GM_EXTENTS`	items whose extents (always a rectangle) must pass the testing method with the selected item
`SIS_GM_GEOMETRY`	items whose geometry must pass the testing method with the selected item

Example `GisCreateLocusFromItem "Scheme20", SIS_GT_CONTAIN, SIS_GM_GEOMETRY`

Available MM, ME, MD, OM, OD, ASC

❍ CreateNorthPoint

Create a north point item using the current open photo item.

Syntax `GisCreateNorthPoint ( x, y, z, shape, s )`

Arguments *x, y, z* DOUBLE
the position of the north point

shape STRING
the name of the north point item

s DOUBLE
the scale of the north point

Example `GisCreateNorthPoint 10, 10, 0, "NorthPoint1", 1`

Available MM, ME, MD, OM, OD, ASC

❍ CreateOpenGisSqlOverlay

Create an overlay using an OpenGIS conformant database. The overlay uses the OpenGIS SQL 92 Database dataset.

Syntax `GisCreateOpenGisSqlOverlay ( pos, connect, ftable, gtable )`

Arguments *pos* SHORT INTEGER
the position in the overlays list at which to insert the overlay. If this argument specifies a position in the existing overlays, the new overlay will not replace the existing overlay at the given position, but will shuffle any other overlays down the list.

connect STRING
the database connection string ➲page 222, **Connecting to databases**

ftable STRING
the OpenGIS feature table. This is not the same as a Cadcorp SIS feature table.

gtable STRING
the OpenGIS geometry table

Example `GisCreateOpenGisSqlOverlay 2 connect, "Resources", "Geometry"`

Available MM, ME, MD, OM, OD, ASC

❍ CreatePhaseOverlay

Create a new phase of an existing overlay.

Syntax `GisCreatePhaseOverlay ( oldpos, newpos )`

Arguments *oldPos* SHORT INTEGER
the position in the overlays list of the overlay being phased

newPos SHORT INTEGER
the position in the overlays list at which to insert the new phase overlay. If this argument specifies a position in the existing overlays, the new overlay will not replace the existing overlay at the given position, but will shuffle any other overlays down the list.

Notes Phase overlays refer to the same data as their original overlay but can have different styles, scale thresholds, filters, and so on set on them. The underlying data in the original overlay is not copied, merely referenced again.

Available MM, ME, MD, OM, OD, ASC

❍ CreatePhoto

Create a photo item in the current window, filling it with the previously composed window.

Syntax `GisCreatePhoto ( x1, y1, x2, y2 )`

Arguments *x1, y1, x2, y2* DOUBLE
the rectangular extents of the photo item

Notes This method respects the axes angle setting, so the x, y, and z values are interpreted within the axes and all new items created will align to the axes angle.

Available MM, ME, MD, OM, OD, ASC

❍ CreatePieTheme

Create a new Pie Chart theme.

Syntax `GisCreatePieTheme ( nSlices )`

Arguments *nSlices* SHORT INTEGER
the number of slices in the pie chart, in the range 1 to 256

Notes After editing the theme properties, use `StoreTheme` to save the theme in a named object library.

Available MM, ME, MD, OV, OM, OD, ASC

❍ CreatePoint

Create a point item.

Syntax `GisCreatePoint ( x, y, z, shape, a, s )`

Arguments *x, y, z* DOUBLE
the position of the point

shape STRING
the shape of the point

a, s DOUBLE
the angle, in radians, and scale of the point

Notes If a group is open, graphics are added to the group, otherwise a new item is created.

Example `GisCreatePoint 10, 10, 0, "New Point Symbol", 0, 1`

Available MM, ME, MD, OV, OM, OD, ASC

❍ CreatePropertyFilter

Create a named property filter, based on a property formula. It replaces any existing filter with the same name.

Syntax `GisCreatePropertyFilter ( filter, formula )`

Arguments *filter* STRING
the named filter to create or replace

formula STRING
the property formula, eg `_closed&=0`

Example `GisCreatePropertyFilter "Closed Items", "_closed&=0"`

Available MM, ME, MD, OV, OM, OD, ASC

❍ CreateQZoneFromGrid

Create a QZone item from the cells in the current grid item which are between two values.

Syntax `GisCreateQZoneFromGrid ( v1, v2 )`

Arguments *v1, v2* DOUBLE
the range of values. Grid cells in this range will be included in the resulting QZone item. The resolution of the resulting QZone depends on the size of the grid cells.

Available MD, OD, ASC

❍ CreateRangeTheme

Create a new Range theme. After editing the theme properties, use `StoreTheme` to save the theme in a named object library.

Syntax `GisCreateRangeTheme ( formula, nRanges )`

Arguments *formula* STRING
the formula to use for matching values, which must evaluate to a number

nValues SHORT INTEGER
the number of ranges in the theme, in the range 2 to 15

Example `GisCreateRangeTheme "Rge18&", 10`

Available MM, ME, MD, OV, OM, OD, ASC

❍ CreateRectangle

Create a rectangular area item. This API method respects the axes angle setting. This means that the x, y, and z values are interpreted within the axes and all new items creates will align to the axes angle.

Syntax `GisCreateRectangle ( x1, y1, x2, y2 )`

Arguments *x1, y1, x2, y2* DOUBLE
the rectangular extents of the area item

Notes If a group is open, graphics are added to the group, otherwise a new item is created.

Available MM, ME, MD, OV, OM, OD, ASC

❍ CreateRectLocus

Create a named rectangular locus, replacing any existing locus with the same name.

Syntax `GisCreateRectLocus ( locus, x1, y1, x2, y2 )`

Arguments *locus* STRING
the named locus to create or replace

x1, y1, x2, y2 DOUBLE
the rectangular extents of the locus

Available ME, MD, OD, ASC

❍ CreateReliefTheme

Create a new Relief theme. After editing the theme properties, use `StoreTheme` to save the theme in a named object library.

Syntax `GisCreateReliefTheme ( colset )`

Arguments *colset* STRING
the colourset on which to base the Relief theme

Available MM, ME, MD, OV, OM, OD, ASC

❍ CreateRubberSheet

Create a rubber sheet item from the displace items in a named list.

Syntax `GisCreateRubberSheet ( list )`

Arguments *list* STRING
the named list which must contain three or more displacement items

Available ME, MD, OD, ASC

❍ CreateScaleBar

Create a scale bar item using the current open photo item.

Syntax `GisCreateScaleBar ( x, y, z, shape, a )`

Arguments

x, y, z DOUBLE
the position of the scale bar

shape STRING
the shape of the scale bar

a DOUBLE
the angle, in radians, of the scale bar

Available MM, ME, MD, OM, OD, ASC

❍ CreateScatterGrid

Create a grid item from the hook points of the items in a named list.

Syntax `GisCreateScatterGrid ( list, formula, mode, ox, oy, oz, cx, cy )`

Arguments

list STRING
a named list which contains the items from which to make a scatter grid. The hook point of each item is used to seed a value in the grid item. The items will typically be point items, but any item class may be used. The items are not edited by this operation, and therefore can be from a read-only dataset.

formula STRING
a formula which is evaluated for each item in the named list. This parameter is ignored for the `SIS_SCATTER_GRID_COUNT` mode.

mode SHORT INTEGER
the method used to calculate each grid cell value:

`SIS_SCATTER_GRID_INTERPOLATE`	the cell value is the weighted average for the formula evaluated on the three closest items
`SIS_SCATTER_GRID_CLOSEST`	the cell value is equal to the formula evaluated on the closest item
`SIS_SCATTER_GRID_SUM`	set the value of each grid cell to the sum of the formula evaluated for each item which is inside the cell
`SIS_SCATTER_GRID_COUNT`	count the number of items in each grid cell

ox, oy, oz DOUBLE
the origin of the grid item. This is not the bottom-left-hand corner of the grid, but is a position around which the grid will position itself. Use (0.0,0.0,0.0) to make a grid aligned to the current axes.

cx, cy DOUBLE
the x and y size of each grid cell in metres. The maximum size of the grid item is 1000 by 1000 cells. The created grid will always cover the listed items, so if a very small cell size is specified, this routine can fail.

Example

```
GisCreateScatterGrid "ItemList", "Formula", SIS_SCATTER_GRID_SUM, 0, 0, _
  0, 10, 10
```

Available MD, OD, ASC

❍ CreateSurface

Create a surface item from the current open area item.

Syntax `GisCreateSurface ( )`

Available MD, OD, ASC

❍ CreateText

Create a point text item.

Syntax `GisCreateText ( x, y, z, text )`

If a group is open, graphics are added to the group, otherwise a new item is created.

x, y, z DOUBLE
the position of the text item

text STRING
the text to be created

Available MM, ME, MD, OV, OM, OD, ASC

❍ CreateThiessen

Create Thiessen area items from the hook points of the items in a named list.

Syntax `GisCreateThiessen ( listOutput, list, bClipToCurItem )`

Arguments *listOutput* STRING
the list of Thiessen area items

list STRING
the list of items whose hook points will be used to create the Thiessen areas

bClipToCurItem SHORT INTEGER

True	use current item as a clipping boundary for the Thiessen areas, eg County line
False	do no use current item as a clipping boundary

Notes If a group is open, graphics are added to the group, otherwise a new item is created.

Example `GisCreateThiessen "OutList", "Points", True`

Available MD, OD, ASC

❍ CreateTin

Create a Triangular Irregular Network (TIN) from the hook points of the items in a named list. A TIN is a special type of surface item.

Syntax `GisCreateTin ( list )`

Arguments *list* STRING
the list of items whose hook points will be used to create the TIN

Notes If a group is open, graphics are added to the group, otherwise a new item is created.

Available MD, OD, ASC

❍ CreateTopoTheme

Create a new Topology theme which displays a schematic layout of a topological network. After editing the theme properties, use `StoreTheme` to save the theme in a named object library.

Syntax `GisCreateTopoTheme ( )`

Available ME, MD, OD, ASC

❍ CreateValueListFilter

Create an empty named value-list filter, based on the value of an integer property (&). It replaces any existing filter with the same name.

Syntax `GisCreateValueListFilter ( filter, propertyName )`

Arguments *filter* STRING
the name of the value-list filter to create or replace. If the property to be used is `_id&`, use `CreateLinkFilter`.

propertyName STRING
the property whose values will be compared by the filter

Example `GisCreateValueListFilters "URN Numbers", "URN$"`

Available MM, ME, MD, OM, OD, ASC

❍ DefineNolDatum

Create a named geoid datum using the standard seven Bursa-Wolf parameters to modify WGS84, replacing any existing datum with the same name.

The named datum is used when defining projections.

Syntax `GisDefineNolDatum ( datum,re,rp,dx,dy,dz,ex,ey,ez,m,pm )`

Arguments *datum* STRING
the named datum to create or replace, eg WGS 84

re, rp DOUBLE
the equatorial and polar radius of the ellipsoid, specified in metres

dx, dy, dz DOUBLE
the shifts to apply to the ellipsoid, specified in metres

ex, ey, ez DOUBLE
the rotational adjustments about the X, Y, and Z axes, specified in minutes of arc

m DOUBLE
the correction scale factor, specified in parts per million. Use 0.0 for no scale correction.

pm DOUBLE
the Prime Meridian, specified in radians. At present, this parameter is ignored and must be specified as 0.0. All other values are ignored.

Example
```
GisDefineNolDatum "USER90",6378206,6356584,-4,-102, -129,-0.2570, _
  0.3410, -0.0880, 3.7230, 0
```

Available MM, ME, MD, OM, OD, ASC

❍ DefineNolItem

Store the current open item in a named object library, replacing any existing item with the same name.

Syntax `GisDefineNolItem ( item )`

Arguments *`item`* STRING
the named item to create or replace

Available ME, MD, OD, ASC

❍ DefineNolItemFromLocus

Store a named locus in a named object library as a named item, replacing any existing item with the same name.

Syntax `GisDefineNolItemFromLocus ( item, locus )`

Arguments *`locus`* STRING
the named locus to be stored

`item` STRING
the named item to create or replace

Example `GisDefineNolItemFromLocus "Shema10", Talbot"`

Available ME, MD, OD, ASC

❍ DefineNolObject

Create a named object from an implicit string, replacing any object with the same name. This method is used to create named brushes, coloursets, and pens. Implicit strings can be queried from existing named objects using `GetImplicitNolObject`.

Syntax `GisDefineNolObject ( aclass, name, implicit )`

Arguments *`aclass`* STRING
the class of named object to be created:

`ABrush`	brush
`AColourset`	colourset
`APen`	pen

`name` STRING
the named object to create or replace

`implicit` STRING
the implicit string which defines the object, as follows.

Brushes

Implicit brushes are defined as follows:

B_ht_r:g:b_r2:g2:b2

where:

B	brush
h	hatching pattern: S for solid; H for hollow; or one of +, X, |, –, or /
t	transparency: T for transparent, O for opaque
r:g:b	red, green, and blue values from 0 to 255
r2:g2:b2	red, green, and blue values from 0 to 255 which define background colour (optional)

For example, `B_SO_128:0:128` displays areas in solid, opaque, purple.

In addition, hatching patterns, shapes, and bitmaps can be appended to the implicit brush string, as follows:

B_ht_r:g:b_r2:g2:b2_gapx_gapy_angle_hatchX_hatchY_pen_shape_bitmap

where:

gapx,gapy	the gaps between hatch lines, or between shape and bitmap repetitions, in 1/100ths of a millimetre
angle	the hatch angle, in 1/10ths of a degree
hatchX,hatchY	Boolean flags controlling whether or not hatch lines are drawn
pen	the hatch line pen
shape	the repeated shape drawn at the hatch angle
bitmap	the repeated bitmap, scaled to fit into (*gapX*, *gapY*) but ignoring the hatch angle

Coloursets

Implicit coloursets are defined as follows:

C_x_{nv,v1:r1:g1:b1,v2:r2:g2:b2,...}

where:

C	colourset
x	the method of colour definition: H for HLS (Hue, Luminosity, and Saturation), or R for RGB (Red, Green, and Blue)
nv	the number of colour-value pairs in the colourset
v1...n	the values
rn:gn:bn	HLS values or RGB values

For example, C_R_{4,0.25:64:0:0,0.50:128:0:0,0.75:192:0:0,1.0:255:0:0} would create a colourset with 4 values ranging from 0.0 to 1.0, displayed in shades of red.

Pens

Implicit pens are defined as follows:

P_style_r:g:b_t_o

where:

P	pen
style	the line style: `solid`, `null`, `dot`, `dash`, `dashdot`, or `dashdotdot`
r:g:b	red, green, and blue values from 0 to 255
t	thickness in 1/100ths of a millimetre
o	offset in 1/100ths of a millimetre (optional)

For example, `P_DASH_128:0:128_0_0` would display lines as purple dashed lines.

In addition, dot-dash patterns and shapes can be appended to the implicit pen string as follows:

{*shpHead*, *shpTail*, *nt*, *l*, *g*, *s*, ...*f* }

where:

shpHead	shape at the head of the line, or a space
shpTail	shape at the tail of the line, or a space
nt	the number of (*l*,*g*,*s*) triples which follow
l,g,s	the line length, gap length, and shape (or a space)
f	flag for fitted pattern: 0 for no or 1 for yes

Example `GisDefineNolObject "ABrush", "Parish", "B_XT_100:50:50_50:50:50"`

Available ME, MD, OD, ASC

❍ DefineNolPrintTemplate

Define a named print template from the current window contents, replacing any existing print template with the same name.

Syntax `GisDefineNolPrintTemplate ( pTemplate )`

Arguments *ptemplate* STRING
the named print template to create or replace

Available MM, ME, MD, OM, OD, ASC

❍ DefineNolPrjLatLon

Create a named (Latitude,Longitude) projection, replacing any existing projection with the same name.

Syntax `GisDefineNolPrjLatLon ( prj, lat, lon, datum, bDeg )`

Arguments *prj* STRING
the named projection to create or replace

lat, lon DOUBLE
the projection origin, in degrees (see Notes)

datum STRING
the named geodetic datum on which to base the projection. This can be any named datum previously created or loaded from a named object library, eg WGS 84.

bDeg SHORT INTEGER

True store co-ordinates in degrees

False store co-ordinates in radians

Example `GisDefineNolPrjLatLon "NewProj", 0.0, 0.0, "USER90", 0`

Notes A (0.0,0.0) origin should normally be used when matching a projection to external data. If the projection is going to be used for creating new data, using a local (lat,lon) origin near the centre of the expected extents allows spatial references to be used with a smaller span, thus increasing the spatial reference resolution.

Available MM, ME, MD, OM, OD, ASC

❍ DefineNolPrjTm

Define a named Transverse Mercator projection, replacing any existing projection with the same name.

Syntax `GisDefineNolPrjTm ( prj, lat, lon, datum, f0, cx, cy, cz, tometre )`

Arguments *prj* STRING
the named projection to create or replace

lat, lon DOUBLE
the projection origin, in degrees

datum STRING
the named geoid datum on which to base the projection. This can be any named datum previously created using `DefineNolDatum`, or loaded from a named object library eg WGS 84.

f0 DOUBLE
the scale on Central Meridian

cx, cy, cz DOUBLE
the position of the false origin

tometre DOUBLE
the conversion from projection units to metres

Example `GisDefineNolPrjTm "UserTM", 0, 0, "UserGO", 0, 99966, 500000, 0, 0, 1`

Available MM, ME, MD, OM, OD, ASC

❍ DefineNolShape

Define a named shape from the items in a named list, replacing any existing shape with the same name.

Syntax `GisDefineNolShape ( shape, list, x, y, s )`

Arguments `shape` STRING
the named shape to create or replace

`list` STRING
the named list containing the items which make up the shape

`x, y, z` DOUBLE
the hook point of the shape

`s` DOUBLE
the scale of the shape

Example `GisDefineNolShape "TrigPoint", "Trig", 50, 100, 0, 2`

Available ME, MD, OD, ASC

❍ DefineNolView

Define a named view from the view in the current window, replacing any existing view with the same name.

Syntax `GisDefineNolView ( view )`

Arguments `view` STRING
the named view to create or replace

Available MM, ME, MD, OM, OD, ASC

❍ DefineRecordset

Define a named recordset, for use with databases, replacing any existing recordset with the same name.

Syntax `GisDefineRecordset ( rs, connect, tables, columns, aliases, sqlwhere )`

Arguments `rs` STRING
the named recordset to create or replace

`connect` STRING
the database connection string ➲page 222, **Connecting to databases**

`tables` STRING
a comma-delimited list of tables which contain the columns referred to in the columns argument

`columns` STRING
a comma-delimited list of columns which contain the data which will be available when the named recordset is used

`aliases` STRING
a comma-delimited list of aliases for the columns referred to in the columns argument

`sqlwhere` STRING
an optional SQL WHERE expression, eg:
`(Table.StatusColumn = 'Pending' Or Table.StatusColumn = 'Agreed')`
This expression should be wholly self-contained, using brackets if they are supported in the database, because under some circumstances, such as if a spatial reference is

being used, Cadcorp SIS automatically generates a WHERE clause with this expression appended using And.

Returns STRING

the full ODBC connection string used

Example

```
rv = GisDefineRecordset ("Depths", "", "Soundings", _
  "NorthValue, EastValue", "XValue, YValue", "Depth < 0"
```

creates a recordset Depths where the user is prompted for the Database. The columns NorthValue and EastValue are to be read and assigned the aliases XValue and YValue. The recordset will include only soundings which are negative values.

Notes The `tables`, `columns`, and `aliases` arguments must all contain the same number of comma-delimited entries. The table/column/alias entry at each position in the each comma-delimited list must be consistent with the other two lists. When calling `CreateDbBlobOverlay` or `CreateDbPointOverlay`, the `nf` arguments refer to positions within the table/column/alias comma-delimited lists, starting at 0.

Available MM, ME, MD, OM, OD, ASC

❍ Delete

Delete all the items in a named list.

Syntax

```
GisDelete ( list )
```

Arguments `list` STRING

a named list containing the items to be deleted

Notes If this method succeeds, the named list and all the items in the list will be deleted. To delete a list without deleting the items, use the `EmptyList` method.

Available MM, ME, MD, OV, OM, OD, ASC

❍ DeleteItem

Delete the current open item.

Syntax

```
GisDeleteItem ( )
```

Available MM, ME, MD, OV, OM, OD, ASC

❍ DeleteNolObject

Delete a named object from a named object library (NOL).

Syntax

```
GisDeleteNolObject ( nPos, aclass, name )
```

Arguments `nPos` SHORT INTEGER

the position in the list of NOLs of the NOL containing the named object to be deleted

`aclass` STRING

the class of named object to be deleted. See *Named Object Library Classes* in the on-line help for valid classes.

`aname` STRING

the named object to delete

Example

```
GisDeleteNolObject 1, "ABrush", "Parish"
```

Available ME, MD, OD, ASC

❍ DeleteRecordset

Delete a named recordset.

Syntax `GisDeleteRecordset ( rs )`

Arguments *rs* STRING
a named recordset to delete, previously created using `DefineRecordset`

Available MM, ME, MD, OM, OD, ASC

❍ DescribeProperty

Set the description of a property. The property window for the remainder of the session will use the description.

Syntax `GisDescribeProperty ( prop, desc )`

Arguments *prop* STRING
the property being described

desc STRING
the new property description

Example `GisDescribeProperty "TYP&", "Polygon Category"`

Available MM, ME, MD, OM, OD, ASC

❍ DeselectAll

Clear the current selection list.

Syntax `GisDeselectAll ( )`

Available MM, ME, MD, OV, OM, OD

❍ DigitiserSnap

Send a digitised position into the current command.

Syntax `GisDigitiserSnap ( x, y, z, nButton )`

Arguments *x, y, z* DOUBLE
the digitised position

nButton SHORT INTEGER
the number of the digitiser button pressed. This argument is currently ignored.

Example `GisDigitiserSnap 123.4, 55.9, 0, 0`

Available ME, MD

❍ DoCommand

Execute a command by use of its ACom equivalent. In Cadcorp SIS Control, the command will be executed immediately.

Syntax
```
ControlName.DoCommand ( comname )
GisDoCommand ( comname )
```

Arguments `comname` STRING
the command to execute

Example
```
SIS.DoCommand "AComShowProgramWindow"
GisDoCommand "AComShowProgramWindow"
```
displays the Program Window. This is useful during the development of an application.

Notes This method executes a `CallCommand` (if the command is a one-shot) or a `SwitchCommand` (if the command is a callback) when used via GisLink.

Available MM, ME, MD, OV, OM, OD

❍ DrapeBitmap

Drape a bitmap item, stored in a named object library, in the 3D window.

Syntax
```
GisDrapeBitmap ( name )
```

Arguments `name` STRING
the name of the bitmap item

Example
```
GisDrapeBitmap ("AerialPhoto")
```
drapes the AerialPhoto bitmap item over a 3D surface in the 3D window. Bitmap items can be created and saved in a named object library using the `CreateDrapeBitmap` method.

Available MD, OD, ASC

❍ DrawList

Draw items in a named list with overridden styles. This changes the display only until the next time it is redrawn.

Syntax
```
GisDrawList ( drawcode, list, pen, brush, shape,font )
```

Arguments `drawcode` SHORT INTEGER

`SIS_CURRENTWINDOW`	redraw the items in the current window only
`SIS_CURRENTSWD`	redraw the items in all windows which contain the current SWD
`SIS_ALLWINDOWS`	redraw the items in all windows

`list` STRING
the named list containing the items to be drawn

`pen` STRING
the pen with which to draw the items

`brush` STRING
the brush with which to draw the items

`shape` STRING
the shape with which to draw the items

`font` STRING
the font with which to draw the items

Example `GisDrawList SIS_CURRENTWINDOW, "Conservation", "Purple", "Pink", Cross", _ "Arial"`

Available MM, ME, MD, OV, OM, OD, ASC

❍ EmptyGroup

Empty the current open group item, deleting all items within the group.

Syntax `GisEmptyGroup ( )`

Available MM, ME, MD, OM, OD, ASC

❍ EmptyList

Empty all the items from a named list and delete the named list.

Syntax `GisEmptyList ( list )`

Arguments *list* STRING
the named list to empty and delete

Notes Every time a list is used, the results are appended to the current values in the list, unless it has been emptied. The `EmptyList` method does not delete the items within Cadcorp SIS. To delete the items in a list, use the `Delete` method.

Available MM, ME, MD, OV, OM, OD, ASC

❍ EnsureOpenWithin

Force datasets in the current window to open any items within the extents, at the given scale.

Syntax `GisEnsureOpenWithin ( x1, y1, z1, x2, y2, z2, s )`

Arguments *x1, y1, z1, x2, y2, z2* DOUBLE
the co-ordinates of the cuboid within which all items will be opened

s DOUBLE
the scale at which to open items

Available MM, ME, MD, OV, OM, OD, ASC

❍ EvaluateFlt

Evaluate a formula, which has a floating point result.

Syntax `rv = GisEvaluateFlt (objectType, nObject, formula )`

Arguments *objectType* SHORT INTEGER
➲page221, **Object types**

nObject LONG INTEGER
the index of the object type

formula STRING
the formula to evaluate

Returns DOUBLE
the evaluated floating point value of the given formula

Example
```
rv = GisEvaluateFlt (SIS_OT_CURITEM, 0, (GisGetFlt (SIS_OT_CURITEM, 0, _
  "Value#" ) /100 ) )
```

Available MM, ME, MD, OV, OM, OD, ASC

❍ EvaluateInt

Evaluate a formula, which has an integer result.

Syntax
```
rv = GisEvaluateInt (objectType, nObject, formula )
```

Arguments

objectType SHORT INTEGER
➲page221, **Object types**

nObject LONG INTEGER
the index of the object type

formula STRING
the formula to evaluate

Returns LONG INTEGER
result of the formula, as an integer

Example
```
rv = GisEvaluateInt (SIS_OT_CURITEM,0,(GisGetInt(SIS_OT_CURITEM, _
  0, "Number&")*100))
```

Available MM, ME, MD, OV, OM, OD, ASC

❍ EvaluateStr

Evaluate a formula, which has a string result.

Syntax
```
rv = GisEvaluateStr (objectType, nObject, formula )
```

Arguments

objectType SHORT INTEGER
➲page221, **Object types**

nObject LONG INTEGER
the index of the object type

formula STRING
the formula to evaluate

Returns STRING
the evaluated string value of the formula

Available MM, ME, MD, OV, OM, OD, ASC

❍ Exit

Exit the Cadcorp SIS session, using the given savecode.

Syntax
```
GisExit ( savecode )
```

Arguments `savecode` SHORT INTEGER

`SIS_NOSAVE`	do not save any modified datasets and/or SWDs
`SIS_SAVE`	save all modified datasets
`SIS_PROMPTSAVE`	prompt the user with each modified dataset and/or SWD

Example `GisExit SIS_NOSAVE`
exits the current session without saving any changes

Available MM, ME, MD

❍ ExplodeOverlayTheme

Explode an overlay theme into a new overlay, creating graphic items based on the theme, such as pie charts, bar charts, and so on.

Syntax `GisExplodeOverlayTheme ( pos, nTheme, s )`

Arguments `pos` SHORT INTEGER
the position of the overlay in the overlays list whose theme is to be exploded

`nTheme` SHORT INTEGER
the index of the theme, starting at 0. Use the Number of themes property, `_nTheme&`, to find out the number of theme objects in an overlay.

`s` DOUBLE
the scale at which to explode the overlay theme graphics. Use the Display scale property, `_displayScale#`, to explode the graphics at the current screen scale.

Example `GisExportOverlayTheme 0, 1, 1250`

Available MM, ME, MD, OM, OD, ASC

❍ Export

Export data using a Plug-in Exporter.

Syntax `GisExport ( clsExport, filename, params )`

Arguments `clsExport` STRING
the Plug-in Exporter class to use

`filename` STRING
the name of the exported file

params STRING
optional parameters used to configure the Plug-in Exporter. There must be no spaces in *params*.

Description	Value of clsExport	Parameter 1	Parameter 2
ArcView *.shp	`AArcInfoShapeExporter`	*index* overlay position to export	*type* 0: Point 1: Arc 2: Polygon 3: Multipoint

Example:

```
GisExport "AArcInfoShapeExporter","c:\test.shp","index=0,type=3"
```

AutoCAD *.dwg	`ADwgExporter`	*attributes* True: export attributes False: do not export attributes	*itemlayers* True: use Layer name False: use Overlay name

Example:

```
GisExport "ADwgExporter","c:\test.dwg","attributes=True,itemlayer=false"
```

AutoCAD *.dxf	`ADxfExporter`	*attributes* True: export attributes False: do not export attributes	*Itemlayers* True: use Layer name False: user Overlay name

Example:

```
GisExport "ADxfExporter","c:\test.dxf","attributes=True, _
   itemlayer=false"
```

EuroNAV *.gxf	`AGxfExporter`	none	none

Example:

```
GisExport "AGxfExporter","c:\test.gxf",""
```

MapInfo *.mif	`AMapInfoExporter`	*index* overlay position to export	*width* width of char column, default is 30

Example:

```
GisExport "AMapInfoExporter","c:\test.mif","index=1,width=30"
```

Oracle	`AOscExporter`	none	none

Example:

```
GisExport "AOscExporter", ""
```

Description	Value of clsExport	Parameter 1	Parameter 2
SED Export	`ASisExporter`		

Example:
```
GisExport "ASisExporter", "c:\test.sed",""
```

Notes Other third party exporters may be available. Consult your supplier for information on these.

Available MM, ME, MD, OM, OD, ASC

❍ ExportBds

Export all the current viewable data to a Base Dataset (BDS) file. This command exports only items displayed on the screen, not all the items in the dataset(s).

Syntax
```
GisExportBds ( filename, precision )
```

Arguments

filename STRING
the name of the exported file

precision SHORT INTEGER
the precision of the items in the exported file

16	16-bit integers
32	32-bit integers
64	64-bit double precision floating point numbers

Example
```
GisExportBds "c:\data\planning.bds", 64
```

Available MM, ME, MD, OM, OD

❍ ExportBmp

Export the current view to a Windows Bitmap (BMP) file.

Syntax
```
GisExportBmp ( filename, bMono, w, h )
```

Arguments

filename STRING
the name of the exported file

bMono SHORT INTEGER

True	create a black-and-white bitmap
False	create a bitmap of the same colour depth as the system graphics

w, h LONG INTEGER
the width and height of the bitmap in pixels

Example
```
GisExportBmp "c:\data\planning.bmp", False, 1024, 1024
```

Available MM, ME, MD, OM, OD

❍ ExportECW

Export the current view to an ECW file

Syntax `GisExportECW ( filename, w, h, compression )`

Arguments *filename* STRING
the name of the exported file

w, h LONG
the width and height of the ECW image in pixels

compression LONG
the target compression ratio

Example `ExportECW ("c:\Raster\map.ecw", 600, 400, 10)`
exports the current view as a 600 x 400 raster image in ECW format with a 10:1 compression ratio

Available MD, OD

❍ ExportFeatureTable

Export a named feature table to a comma-separated file.

Syntax `GisExportFeatureTable ( ftable, filename )`

Arguments *ftable* STRING
the named feature table to export

filename STRING
the name of the exported file

Example `GisExportFeatureTable "Land-Line", "c:\data\Land-Line.csv"`

Available ME, MD, OD, ASC

❍ ExportJpeg

Export the current view to a JPEG file.

Syntax `GisExportJpeg ( filename, w, h )`

Arguments *filename* STRING
the name of the exported file

w, h LONG INTEGER
the width and height of the bitmap, in pixels

Example `GisExportJpeg "c:\data\planning.jpg", 1024, 1024`

Available MM, ME, MD, OM, OD

❍ ExportPdf

Export the current view to an Adobe Portable Document Format file.

Syntax `ExportPdf (filename, paperFormat, params)`

Arguments *filename* STRING
the name of the exported file

paperFormat STRING

a string describing the paper size, format, resolution, and so on. See Notes, below.

params STRING

a comma-separated list of key-value pairs. These values are available for inspection in Acrobat Reader.

`Title=value`

`Author=value`

`Subject=value`

`Keywords=value`

Values containing non-alphanumeric characters, such space, asterisk, and so on, should be enclosed in quotation marks (ASCII character 34). Visual Basic programmers can enter the quote symbol twice to embed a single quote symbol.

`ContentLandscape` BOOLEAN

if True, rotates the content, rather than the paper. If False, rotates the paper rather than the content.

Example

```
ExportPdf ("C:\Docs\map.pdf", "A3:2cm@600dpi", "Title=""Town Centre"", _
  Author=MF,Subject=Maps, ContentLandscape=False")
```

exports the current view as a 600 dpi file, with a 2cm border, in A3 portrait format. The PDF document's information dictionary contains entries for Title (Town Centre), Author (MF), and Subject (Maps).

Notes

The *paperFormat* argument is a string that defines the paper size, border size, output resolution, and units.

Here are some examples:

Argument	Format
"A4"	A4 portrait, no border, 300 dpi
"A4*"	A4 landscape, no border, 300 dpi
"500x800"	500pts wide, 800 pts high, no border, 300 dpi
"20x30cm"	20cm wide, 30cm high, no border, 300 dpi
"A4:36"	A4 portrait, 36pts border, 300 dpi
"20x30cm:150"	20cm wide, 30cm high, 150pts border, 300 dpi
"20x30cm:15mm"	20cm wide, 30cm high, 15mm border, 300 dpi
"A4*@600"	A4 landscape, no border, 600 dpi
"A3:2cm@600dpi"	A3 portrait, 2cm border, 600 dpi

Build the *paperFormat* argument from the following components. The square brackets indicate optional parameters.

Block	Consists of
paperFormat	*PaperSize* [*BorderSep Border*] [*ResolSep Resolution*]
PaperSize	*PaperSizePredef* [*InLandscape*] [*PaperSizeUnits*]
PaperSizePredef	one of A4, A3, or a paper size that appears in the Export PDF dialog in Cadcorp SIS
InLandscape	*
PaperSizeUnits	*Size* x *Size* [*Unit*]
Size	a positive floating point number
Unit	one of cm, mm, or one of the units appearing in the Export PDF dialog in Cadcorp SIS. The default units are points.
BorderSep	:
Border	*Size* [*Unit*]
ResolSep	@
Resolution	*Number* [*ResolUnit*]
Number	a positive integer
ResolUnit	*dpi*

Available MM, ME, MD, OM, OD, ASC

❍ ExportPng

Export the current view to a Portable Network Graphics (PNG) file.

Syntax `GisExportPng ( filename, w, h )`

Arguments *filename* STRING
the name of the exported file

w, h LONG INTEGER
the width and height of the bitmap, in pixels

Example `GisExportPng "c:\data\planning.png", 1024, 1024`

Available MM, ME, MD, OM, OD

❍ ExportVrml

Export the current view to a VRML (3D export) file.

Syntax `GisExportVrml ( filename )`

Arguments *filename* STRING
the name of the exported file

Available MD, OD

❍ ExportWmf

Export the current view to a Windows Metafile (WMF) file.

Syntax `GisExportWmf ( filename )`

Arguments *filename* STRING
the name of the exported file

Available MM, ME, MD, OM, OD

❍ FacetGeometry

Replace curved geometry segments with shorter straight segments.

Syntax `GisFacetGeometry ( list, tolerance )`

Arguments *list* STRING
the named list containing the items to be facetted

tolerance DOUBLE
the maximum distance allowed between the original curve and the new facet. The smaller the number used, the greater the number of segments.

Example `GisFacetGeometry ("Rivers", 2)`
all curved lines and curved area boundaries of items in the Rivers named list will be facetted into straight segments. The maximum deviation from the original curve will be 2 metres.

Available ME, MD, OD, ASC

❍ FindDatasetOverlay

Find an overlay which contains the given dataset.

Syntax `rv = GisFindDatasetOverlay (nDataset, pos, bForward )`

Arguments *nDataset* LONG INTEGER
the serial number of the dataset to be matched. The serial number can be obtained from the `_nDataset&` property of an overlay, or from the `GetDataset`, `GetDatasetContainer`, or `FindExternalDataset` methods.

pos SHORT INTEGER
the position in the list of overlays from which to calculate the search start (see *bForwards*). If set to –1, the entire list is searched in the order specified by *bForwards*.

bForwards SHORT INTEGER

True search from *pos* + 1 (or the start if *pos* is –1) to the end of the list of overlays

False search from *pos* –1 (or the end if *pos* is –1) to the beginning of the list of overlays

Returns SHORT INTEGER
the position in the list of overlays of an overlay which contains the dataset, or –1 if the dataset is not found

Example `OPos = GisFindDatasetOverlay (4, -1, True)`

Available MM, ME, MD, OM, OD, ASC

❍ FindExternalDataset

Get the serial number of a dataset which is already open.

Syntax `rv = GisFindExternalDataset (dataset )`

Arguments *dataset* STRING
the name of the dataset to find

Returns LONG INTEGER
the serial number of the given dataset ➲page 222, **Serial numbers**

Available MM, ME, MD, OM, OD, ASC

❍ Get3DEye

Get the position of the eye in a 3D window.

Syntax `rv = GisGet3Eye ( )`

Returns STRING
a comma-delimited string containing the x, y, and z co-ordinates of the eye position. Use `SplitPos` to get the x, y, and z values themselves.

Available MD, OD

❍ Get3Dlook

Get the position looked towards in a 3D window.

Syntax `rv = GisGet3Dlook ( )`

Returns STRING
a comma-delimited string containing the x, y, and z co-ordinates of the look position. Use `SplitPos` to get the x, y, and z values themselves.

Available MD, OD

❍ GetAxesAngle

Get the angle of the current axes.

Syntax `rv = GisGetAngle ( )`

Returns SHORT INTEGER
the angle of the current axes in decimal degrees

Available MM, ME, MD, OM, OD, ASC

❍ GetAxesFromLatLonHgt

Get the x, y, and z co-ordinates of a position from its latitude, longitude, and height above sea-level.

Syntax `GisGetAxesFromLatLonHgt ( lat, lat, hgt, datum )`

Arguments *lat, lon, hgt* DOUBLE
the latitude, longitude (both in degrees) and height of the required position in metres

datum STRING
the named geoid datum to use. This can be any named datum previously created using `DefineNolDatum`, or loaded from a named object library, eg WGS 84.

Returns STRING
a comma-delimited string containing the x, y, and z co-ordinates of the position within the current axes

Notes Use `SplitPos` to get the x, y, and z values themselves.

Available MM, ME, MD, OV, OM, OD, ASC

❍ GetAxesPrj

Get a copy of the current axes projection, replacing any existing projection with the same name.

Syntax
```
GisGetAxesPrj ( Projection )
```

Arguments *projection* STRING
the named projection to create or replace. The projection will either be Cartesian or spherical. Use `GetAxesType` to find out which.

Example
```
GisGetAxesPrj "APrjCopy"
```

Available MM, ME, MD, OV, OM, OD, ASC

❍ GetAxesType

Find out whether the current axes are Cartesian or spherical.

Syntax
```
rv = GisGetAxesType ( )
```

Returns SHORT INTEGER
`SIS_AXES_CARTESIAN`
the current axes are Cartesian
`SIS_AXES_SPHERICAL`
the current axes are spherical

Notes The X, Y and Z axes of a Cartesian projection are orthogonal to each other, and their units are metres. The X and Y axes of a spherical projection are degrees of longitude and latitude. The Z axis of a spherical projection measures height above sea-level in metres.

Available MM, ME, MD, OV, OM, OD, ASC

❍ GetBlob

Get a Blob string of the current open item within a projection.

Syntax
```
rv = GisGetBlob ( projection, fmt, precision )
```

Arguments *projection* STRING
the named projection of the returned Blob string

fmt SHORT INTEGER

the format of the stored item Blob

SIS_BLOB_SIS	Cadcorp SIS format
SIS_BLOB_OGIS_WKT	OpenGIS Well-Known Text format

precision SHORT INTEGER

the precision of the Blob string. The Cadcorp SIS format recognises the values 16, 32, and 64 (or 0 for the default value of 64). The values 16, 32, and 64 specify the precision in bits of each co-ordinate. These values correspond to short integers, long integers, and double precision floating point respectively. Using a smaller precision will return shorter Blob strings.

Returns STRING

a string describing the current open item in the chosen format. Cadcorp SIS format strings are encoded and cannot be interpreted except by Cadcorp SIS.

Example

```
Blob = GisGetBlob ( "*APrjNatGrid", 0, 0 )
```

returns the blob string of the current item within the National Grid projection

Notes There is no limit on the length of Blob strings generated by Cadcorp SIS. There are however limits elsewhere: GisLink can only handle strings of a few kilobytes (this does not apply to the Cadcorp SIS Control); Cadcorp SIS datasets which read Blobs from ODBC data sources use a fixed size communication buffer (use the OpenGIS SQL92 Database Maximum Blob size, Editable Blobs Maximum Blob size, and View Blobs Maximum Blob size properties to change the size of the communication buffer).

Available ME, MD, OD, ASC

❍ GetBlobB

Get a Blob string of the current open item, within a projection.

Syntax

```
rv = GisGetBlobB ( projection, fmt, precision )
```

Arguments

projection STRING

the named projection of the returned Blob string

fmt SHORT INTEGER

the format of the stored item Blob:

SIS_BLOB_SIS	Cadcorp SIS format
SIS_BLOB_OGIS_WKB	OpenGIS Well-Known Binary format
SIS_BLOB_OGIS_WKT	OpenGIS Well-Known Text format

precision SHORT INTEGER

the precision of the Blob string. The Cadcorp SIS format recognises the values 16, 32, and 64 (or 0 for the default value of 64). The values 16, 32, and 64 specify the precision in bits of each co-ordinate. These values correspond to short integers, long integers and double precision floating point respectively. Using a smaller precision will return shorter Blob strings.

Returns the Blob data of the current item within the National Grid projection as a variant, containing an array of bytes that describe the current open item in the chosen format.

Example `Blob = GisGetBlobB ( "*APrjNatGrid", 0, 0 )`
returns the Blob data of the current item within the National Grid projection

Notes There is no limit on the length of Blob strings generated by Cadcorp SIS. There are however limits elsewhere: GisLink can handle strings of only a few kilobytes (this does not apply to the Cadcorp SIS Control); Cadcorp SIS datasets which read Blobs from data sources can use a fixed size communication buffer (use the OpenGIS SQL92 Database Maximum Blob size, Editable Blobs Maximum Blob size, and View Blobs Maximum Blob size properties to change the size of the communication buffer).

Available OD, ASC

❍ GetBlobExtent

Get the extent of a Blob string, within a projection.

Syntax
```
rv = GisGetBlobExtent (blob, prj, fmt )
```

Arguments

blob STRING
the stored item Blob string

prj STRING
the named projection of the stored item Blob

fmt SHORT INTEGER
the format of the stored item Blob:

`SIS_BLOB_SIS`	Cadcorp SIS format
`SIS_BLOB_OGIS_WKT`	OpenGIS Well-Known-Text format

Returns STRING
a comma-delimited string containing a pair of x, y, and z co-ordinates describing the extents. Use `SplitExtent` to get the pair of x, y and z values themselves.

Example `blobExtent = GisGetBlobExtent ( blob, "*APrjNatGrid", 0 )`
returns the extent of the value of blob within the National Grid projection

Available MM, ME, MD, OM, OD, ASC

❍ GetCommandTick

Get the tick state of an application-defined command previously added using `AddCommand`. In other words, does the menu for this command have a pushed-in icon next to it?

Syntax
```
rv = GisGetCommandTick ( comname )
rv = ControlName.GetCommandTick ( comname )
```

Arguments *comname* STRING
the command whose tick state is to be verified. This is the value of *menu* in the method `AddCommand`.

Returns SHORT INTEGER

True	the tick is on
False	the tick is off

Example `bTickState = GGisGetCommandTick("&Utility|&Autosave")`

Available MM, ME, MD, OM, OD

❍ GetCoordExtent

Get the extent corresponding to a co-ordinate format string.

Syntax `rv = GisGetCoordExtent (coordClass, coord )`

Arguments *`coordClass`* STRING
the co-ordinate format class to use in interpreting the *`coord`* argument

`coord` STRING
the co-ordinate string to be interpreted, in the format specified by the *`coordClass`* argument

Returns STRING
a comma-delimited string containing a pair of x, y, and z co-ordinates describing the extents of the co-ordinate format position

Example `rv = GisGetCoordExtent ("AGridNatGrid", "100,100,100")`

Notes Use `SplitExtent` to get the pair of x, y, and z values themselves. If the co-ordinate format position describes a point, the extents will describe the same point.

The supported co-ordinate formats can be found using the `ASysVarCoordClasses` system variable.

Available MM, ME, MD, OM, OD, ASC

❍ GetCoordString

Get the string representation of a position.

Syntax `rv = GisGetCoordString (coordClass, x, y, z )`

Arguments *`coordClass`* STRING
the co-ordinate format class to use in interpreting the x, y, z co-ordinates

`x, y, z` DOUBLE
the position to be converted into the format specified by the *`coordClass`* argument

Returns STRING
a string in the co-ordinate format specified by the *`coordClass`* argument

Example `rCoord = GisGetCoordString ("AGridNatGrid", 100, 100, 100)`

Notes The supported co-ordinate formats can be found using the `ASysVarCoordClasses` system variable.

Available MM, ME, MD, OV, OM, OD, ASC

❍ GetDataset

Get the serial number of the current open item's dataset. ➲page 222, **Serial numbers**

Syntax `rv = GisGetDataset ( )`

Returns LONG INTEGER

the index of the current open item's dataset

Available MM, ME, MD, OM, OD, ASC

❍ GetDatasetContainer

Get the serial number of the dataset which contains the specified dataset (eg to identify the dataset on which a tile is located).

Syntax `rv = GisGetDatasetContainer ( ndataset, nContainer )`

Arguments *`nDataset`* LONG INTEGER

the serial number of the dataset whose container is required ➲page 222, **Serial numbers**

`nContainer` SHORT INTEGER

the index of the container dataset in the dataset's list of containers (eg if a tile is included in two index datasets, *`nContainer`* can have the value 0 or 1)

Returns LONG INTEGER

the serial number of the dataset container, or 0 if the dataset has no containers or the *`nContainer`* value is greater than the available number of containers

Available MM, ME, MD, OM, OD, ASC

❍ GetDatasetExtent

Get the extents of all of the items in a dataset.

Syntax `rv = GisGetDatasetExtent ( nDataset )`

Arguments *`nDataset`* LONG INTEGER

the serial number of the dataset whose extents are required ➲page 222, **Serial numbers**

Returns STRING

a comma-delimited string containing a pair of x, y, and z co-ordinates describing the extents. Use `SplitExtent` to get the pair of x, y, and z values themselves.

Available MM, ME, MD, OM, OD, ASC

❍ GetDatasetPrj

Get a copy of a dataset projection, replacing any existing projection with the same name.

Syntax `GisGetDatasetPrj ( nDataset, projection )`

Arguments *`nDataset`* LONG INTEGER

the serial number of the dataset whose projection is required ➲page 222, **Serial numbers**

projection STRING
the named projection to create or replace

Example `GisGetDatasetPrj (7, "NewProjection")`
gets the current dataset projection from the dataset, whose serial number is 7, and stores it by the new name NewProjection

Available MM, ME, MD, OM, OD, ASC

❍ GetDisplayExtent

Get the padded visible extents of the current window.

Syntax `rv = GisGetDisplayExtent ( )`

Returns STRING
a comma-delimited string containing a pair of x, y, and z co-ordinates describing the padded visible extents of the current window. Use `SplitExtent` t o get the pair of x, y and z values themselves.

Notes The view extents are padded horizontally or vertically to take account of any difference between the aspect ratio of the view and that of the window. The result will always depend on the shape of the window, and changes whenever the aspect ratio of the window changes, eg when the window is resized by the user. It should therefore not be stored for later use. It can, however, be used with `EnsureOpenWithin` to guarantee that all datasets are opened within the visible extents.

Available MM, ME, MD, OV, OM, OD

❍ GetErrorString

Get the string associated with a Cadcorp SIS error code.

Syntax `rv = SIScontrolName.GetErrorString ( errorCode )`

Arguments *errorCode* SHORT INTEGER
the error code whose string is required. –1 gets the error string for the error code stored in the system variable `_ExecError&`.

Returns STRING
the error string for *errorCode*

Example `ErrorMessage = SIS.GetErrorString (55)`
returns NO COMPOSED WINDOW, the error string for the error code 55

Available OV, OM, OD

❍ GetExtent

Get the extents of the current open item.

Syntax `rv = GisGetExtent ( )`

Returns STRING
a comma-delimited string containing a pair of x, y, and z co-ordinates describing the extents. Use `SplitExtent` to get the pair of x, y, and z values themselves.

Available MM, ME, MD, OM, OD, ASC

❍ GetFeatureFilterBranches

Get the feature codes branching from a parent feature code in a named feature filter.

Syntax `rv = GisFeatureFilterBranches (filter, fcode )`

Arguments *filter* STRING

the named feature filter to query

fcode SHORT INTEGER

the feature code whose children are to be queried. Use 0 to query the top-level feature codes.

Returns STRING

a space-separated string in the following format: *nCodes code1 code2 ... codeN*

Examples `rv = GisFeatureFilterBranches ("Feature Filter.Land-Line", 10102)`

returns a string containing the child feature codes '3 1 4 1006', of the feature code 10102

`CCode = GisGetFeatureFilterBranches ("Feature Filter.Land-Line", 10106)`

returns all the feature codes associated with road

Notes Predefined filters are prefixed by their type:

Prefix	Description	Example
`Class Filter.`	class based filters	`'Class Filter.Area'`
`Compound Filter.`	two filters added together	`'Compound Filter.(Default)'`
`Feature Filter.`	filters based on feature codes	`'Feature Filter.Land-Line'`
`Property Filter.`	filters based on properties	`'Property Filter.Closed'`

Available MM, ME, MD, OM, OD, ASC

❍ GetFeatureTableBranches

Get the feature codes branching from a parent feature code in the currently loaded feature table. Use `LoadFeatureTable` to load a feature table for editing.

Syntax `rv = GisGetFeatureTableBranches (fcode, bCascade )`

Arguments *fcode* SHORT INTEGER

the feature code whose children are to be queried. Use 0 to query the top-level feature codes.

bCascade SHORT INTEGER

True get all of the feature codes in the hierarchy below the given feature code

False get the feature codes directly below the given feature code

Returns STRING

a space-separated string in the following format: *nCodes code1 code2 ... codeN*

Example `CCode = GisGetFeatureTableBranches (1032, True)`

returns all the child feature codes of the code 1032

Available ME, MD, OD, ASC

❍ GetFlt

Get the value of a floating point property on the given object type.

Syntax `rv = GisGetFlt (objectType, nObject, propertyname )`

Arguments *objectType* SHORT INTEGER

➲page 221, **Object types**

nObject LONG INTEGER

the index of the object type

propertyname STRING

the name of the property: all floating point properties end in #

Returns DOUBLE

the floating point value of the property

Example `rv = GisGetFlt (SIS_OT_CURITEM, 0, "Value#")`

Available MM, ME, MD, OV, OM, OD, ASC

❍ GetGeomAngleFromLength

Get the tangent angle a specified length along the geometry of the current open item.

Syntax `rv = GisGetGeomAngleFromLength (nGeom, arclen )`

Arguments *nGeom* LONG INTEGER

the index of the geometry component, starting at 0

arclen DOUBLE

the length along the geometry component

Returns DOUBLE

the angle, in radians, of the geometry component at the length along the geometry component, measured anti-clockwise from the current x-axis

Example `rv = GisGetGeomAngleFromLength (0, 120)`

Notes Use `GetNumGeom` to get the number of geometry components in an item.

Items such as blocks, groups, multi-line, and so on, can be made up of multiple geometry. Therefore the value of *nGeom* can be greater than 0. Items such as lines, areas, shapes, and so on, are made by a single piece of geometry. Therefore the value would be 0.

This method should be used only if the geometry component is two-dimensional. If the geometry component is three-dimensional, consider using `GetGeomTgtFromLength`.

Available MM, ME, MD, OM, OD, ASC

❍ GetGeomDim

Get the dimension of the geometry from the current open item.

Syntax `rv = GisGetGeomDim (nGeom )`

Arguments *nGeom* LONG INTEGER

the index for the geometry component, starting at 0

Returns LONG INTEGER

the dimension of the geometry component. For example, an area item is 2-dimensional, a line item is 1-dimensional, a point item is 0-dimensional.

Notes Use `GetNumGeom` to get the number of geometry components in an item.

Items such as blocks, groups, multi-line, and so on, can be made up of multiple geometry. Therefore the value of *`nGeom`* can be greater than 0. Items such as lines, areas, shapes, and so on, are made by a single piece of geometry. Therefore the value would be 0.

Available MM, ME, MD, OM, OD, ASC

❍ GetGeomLength

Get the length of the geometry from the current open item.

Syntax `rv = GisGetGeomLength (nGeom )`

Arguments *`nGeom`* LONG INTEGER

the index of the geometry component, starting at 0

Returns DOUBLE

the length of the geometry component

Notes Use `GetNumGeom` to get the number of geometry components in an item.

Items such as blocks, groups, multi-line, and so on, can be made up of multiple geometry. Therefore the value of *`nGeom`* can be greater than 0. Items such as lines, areas, shapes, and so on, are made by a single piece of geometry. Therefore the value would be 0.

Available MM, ME, MD, OM, OD, ASC

❍ GetGeomLengthUpto

Get the length along the geometry of the current open item up to a position.

Syntax `rv = GisGetGeomLengthUpto (nGeom, arclenStart, x, y, z )`

Arguments *`nGeom`* LONG INTEGER

the index of the geometry component, starting at 0. Use `GetNumGeom` to get the number of geometry components in an item.

`arclenStart` DOUBLE

the length along the geometry component from which to start the measurement. Use –1.0 to measure from the start of the geometry.

`x, y, z` DOUBLE

the position along the geometry component to measure up to

Returns DOUBLE

the measured length, or –1.0

Notes The *`arclenStart`* argument is useful for geometry which passes through a position more than once, eg a figure-of-eight. To handle this situation, call this method repeatedly, using –1.0 for the *`arclenStart`* argument for the first call, and the returned value for each subsequent call, until –1.0 is returned.

Available MM, ME, MD, OM, OD, ASC

❍ GetGeomNumPt

Get the number of vertices in the geometry of the current open item.

Syntax `rv = GisGetGeomNumPt (nGeom )`

Arguments *nGeom* LONG INTEGER
the index of the geometry component, starting at 0

Returns LONG INTEGER
the number of vertices in the geometry component

Notes Use `GetNumGeom` to get the number of geometry components in an item.

Items such as blocks, groups, multi-line, and so on, can be made up of multiple geometry. Therefore the value of *nGeom* can be greater than 0. Items such as lines, areas, shapes, and so on, are made by a single piece of geometry. Therefore the value would be 0.

Available MM, ME, MD, OM, OD, ASC

❍ GetGeomNumSeg

Get the number of segments in the geometry of the current open item.

Syntax `rv = GisGetGeomNumSeg (nGeom )`

Arguments *nGeom* LONG INTEGER
the index of the geometry component, starting at 0

Returns LONG INTEGER
the number of segments in the geometry component

Notes Use `GetNumGeom` to get the number of geometry components in an item.

Items such as blocks, groups, multi-line, and so on, can be made up of multiple geometry. Therefore this value can be greater than 0. Items such as lines, areas, shapes, and so on, are made by a single piece of geometry. Therefore this value would be 0.

Available MM, ME, MD, OM, OD, ASC

❍ GetGeomPosFromLength

Get the position a specified length along the geometry of the current open item.

Syntax `rv = GisGetGeomPosFromLength (nGeom, arclen )`

Arguments *nGeom* LONG INTEGER
the index of the geometry component, starting at 0

arclen DOUBLE
the length along the geometry component

Returns STRING
a comma-delimited string containing the x, y, and z co-ordinates of a position along the geometry component. Use `SplitPos` to get the x, y, and z values themselves.

Example `rv = GisGetGeomPosFromLength (0, 500)`

Notes Use `GetNumGeom` to get the number of geometry components in an item.

Items such as blocks, groups, multi-line, and so on, can be made up of multiple geometry. Therefore the value of *nGeom* can be greater than 0. Items such as lines, areas,

shapes, and so on, are made by a single piece of geometry. Therefore the value would be 0.

Available MM, ME, MD, OM, OD, ASC

❍ GetGeomPt

Get the position of a vertex in the geometry of the current open item.

Syntax `rv = GisGetGeomPt (nGeom, nPt )`

Arguments *nGeom* LONG INTEGER
the index of the geometry component, starting at 0

nPt LONG INTEGER
the index of the vertex, starting at 0

Returns STRING
a comma-delimited string containing the x, y, and z co-ordinates of the vertex. Use `SplitPos` to get the x, y, and z values themselves.

Notes Use `GetNumGeom` to get the number of geometry components in an item.

Items such as blocks, groups, multi-line, and so on, can be made up of multiple geometry. Therefore this value can be greater than 0. Items such as lines, areas, shapes, and so on, are made by a single piece of geometry. Therefore this value would be 0.

Available MM, ME, MD, OM, OD, ASC

❍ GetGeomSegAxis

Get the axis of a bulged segment within the geometry of the current open item.

Syntax `rv = GisGetGeomSegAxis (nGeom, nSeg )`

Arguments *nGeom* LONG INTEGER
the index of the geometry component, starting at 0

nSeg LONG INTEGER
the index of the segment within the geometry component

Returns STRING
a comma-delimited string containing the x, y, and z components of the axis vector. Use `SplitPos` to get the x, y, and z values themselves.

Notes Use `GetNumGeom` to get the number of geometry components in an item.

Items such as blocks, groups, multi-lines, and so on, can be made up of multiple geometry. Therefore the value of *nGeom* can be greater than 0. Items such as lines, areas, shapes, and so on, are made by a single piece of geometry. Therefore the value would be 0.

Available MM, ME, MD, OM, OD, ASC

❍ GetGeomSegBulge

Get the bulge of a segment within the geometry in the current open item.

Syntax `rv = GisGetGeomSegBulge (nGeom, nSeg )`

Arguments *nGeom* LONG INTEGER
the index of the geometry component, starting at 0

nSeg LONG INTEGER
the index of the segment within the geometry component

Returns DOUBLE
the bulge value of the given segment. The bulge factor is the tangent of one quarter of the swept angle. A bulge factor of 0.0 implies a straight segment.

Notes Use `GetNumGeom` to get the number of geometry components in an item.

Items such as blocks, groups, multi-lines, and so on, can be made up of multiple geometry. Therefore the value of *nGeom* can be greater than 0. Items such as lines, areas, shapes, and so on, are made by a single piece of geometry. Therefore the value would be 0.

Available MM, ME, MD, OM, OD, ASC

❍ GetGeomSegShape

Get the shape of a segment within the geometry of the current open item.

Syntax
```
rv = GisGetGeomSegShape ( nGeom, nSeg )
```

Arguments *nGeom* LONG INTEGER
the index of the geometry component, starting at 0

nSeg LONG INTEGER
the index of the segment within the current geometry component

Returns SHORT INTEGER

`SIS_LINE_STRAIGHT`	the segment is straight
`SIS_LINE_BULGE`	the segment is a bulge
`SIS_LINE_BEZIER`	the segment is a bezier

Notes Use `GetNumGeom` to get the number of geometry components in an item.

Items such as blocks, groups, multi-line, and so on, can be made up of multiple geometry. Therefore the value of *nGeom* can be greater than 0. Items such as lines, areas, shapes, and so on, are made by a single piece of geometry. Therefore the value would be 0.

Available MM, ME, MD, OM, OD, ASC

❍ GetGeomSelfIntersection

Get the position of self-intersection of the geometry of the current open item.

Syntax
```
rv = GisGetGeomSelfIntersection ( nGeom, arclenStart )
```

Arguments *nGeom* LONG INTEGER
the index of the geometry component, starting at 0

arclenStart DOUBLE
the length along the geometry component from which to start the search. Use –1.0 to search from the start of the geometry.

Returns DOUBLE

the distance along the geometry component of any self-intersection, or –1.0 if no self-intersection was found. This method returns 0.0 if any error occurs. Any error can be queried from the ASysVarExecError system variable.

Notes Use `GetNumGeom` to get the number of geometry components in an item.

Items such as blocks, groups, multi-line, and so on, can be made up of multiple geometry. Therefore the value of *`nGeom`* can be greater than 0. Items such as lines, areas, shapes, and so on, are made by a single piece of geometry. Therefore the value would be 0.

The *`arclenStart`* argument is useful for geometry which passes through a position more than once, eg a figure-of-eight. To handle this situation, call this method repeatedly, using –1.0 for the *`arclenStart`* argument for the first call, and then the returned value for each subsequent call, until –1.0 is returned.

Available MM, ME, MD, OM, OD, ASC

❍ GetGeomTgtFromLength

Get the tangent vector a specified length along the geometry of the current open item.

Syntax

```
rv = GisGetGeomTgtFromLength (nGeom, arclen )
```

Arguments *`nGeom`* LONG INTEGER

the index of the geometry component, starting at 0. Use `GetNumGeom` to get the number of geometry components in an item.

`arclen` DOUBLE

the length along the geometry of the current open item

Returns STRING

a comma-delimited string containing the x, y, and z components of the tangent vector. Use `SplitPos` to get the x, y, and z values themselves.

Available MM, ME, MD, OM, OD, ASC

❍ GetGridItemValue

Get the value of the cell in the current open grid item at a position. Grid cell values can be set using `SetGridItemValue`.

Syntax

```
rv = GisGetGridValue (x, y, z )
```

Arguments *`x, y, z`* DOUBLE

the position at which to query the grid item value

Returns DOUBLE

the grid item value at the given position

Available MD, OD, ASC

❍ GetHook

Get the hook point of the current open item.

Syntax

```
rv = GisGetHook ( )
```

Returns STRING

a comma-delimited string containing the x, y, and z co-ordinates of the hook point. Use `SplitPos` to get the x, y, and z values themselves.

Available MM, ME, MD, OM, OD, ASC

❍ GetImplicitNolObject

Get the implicit equivalent of an object in a named object library. The implicit string can be used in `DefineNolObject` to create a new named object.

Syntax

```
rv = GisGetImplicitNolObject (aclass, aname )
```

Arguments *aclass* STRING

the class of named object to be queried:

`ABrush`	brush
`AColourset`	colourset
`APen`	pen

name STRING

the named object whose equivalent implicit string is to be queried

Returns STRING

an implicit string equivalent of a named object, eg P_SOLID_255:0:0_0_0 for the standard pen Red

Example

```
Imp = GisGetImplicitNolObject ("ABrush", "Parish")
```

Available MM, ME, MD, OM, OD, ASC

❍ GetInt

Get the value of an integer property on the given object type.

Syntax

```
rv = GisGetInt (objectType, nObject, propertyName )
```

Arguments *objectType* SHORT INTEGER

➲page221, **Object types**

nObject LONG INTEGER

the index of the object type

propertyname STRING

the name of the property: all integer properties end in &

Returns LONG INTEGER

the integer value of the property

Example

```
rv = GisGetInt (SIS_OT_CURITEM, 0, "_id&")
```

Available MM, ME, MD, OV, OM, OD, ASC

❍ GetLatLonHgtFromAxes

Get the latitude, longitude, and height above sea-level of an x, y, z position.

Syntax

```
rv = GisGetLatLonHgtFromAxes (x, y, z, datum )
```

Arguments `x, y, z` DOUBLE
the position within the current axes

`datum` STRING
the named geoid datum to use. This can be any named datum previously created using `DefineNolDatum`, or loaded from a named object library, eg WGS 84.

Returns STRING
a comma-delimited string containing the latitude, longitude (both in degrees) and height of the given position in the given geoid datum. Use `SplitPos` to get the latitude, longitude and height values themselves.

Example `rv = GisGetLatLonHgtFromAxes (100, 100, 0, "WGS 84")`

Available MM, ME, MD, OV, OM, OD, ASC

❍ GetListExtent

Get the extents of all of the items in a named list.

Syntax `GisGetListExtent ( list )`

Arguments `list` STRING
the named list containing the items whose extents are required

Returns STRING
a comma-delimited string containing a pair of x, y, and z co-ordinates describing the extents. Use `SplitExtent` to get the pair of x, y, and z values themselves.

Available MM, ME, MD, OM, OD, ASC

❍ GetListItemFlt

Get the value of a floating point property on an item in a named list.

Syntax `GetListItemFlt (list, n, propertyName )`

Arguments `list` STRING
the named list to query

`n` LONG
the index of the item in the named list

`PropertyName` STRING
the name of the property. All floating point properties end in #.

Returns DOUBLE
the floating point value of the given property, for the given item in the given named list

Example `Area = GetListItemFlt ("Zones", 6, "_area#")`
retrieves the area in square metres of item 6 in the Zones named list. Named list indices run from zero to one less than the size of `list`, so this example accesses the seventh item in the list.

Available MM, ME, MD, OM, OD, ASC

❍ GetListItemInt

Get the value of a long integer property on an item in a named list.

Syntax `GetListItemInt (list, n, propertyName )`

Arguments *list* STRING
the named list to query

n LONG INTEGER
the index of the item in the named list

PropertyName STRING
the name of the property. All integer properties end in &.

Returns LONG INTEGER
the integer value of the given property, for the given item in the given named list.

Example `itemID = GetListItemInt ("Zones", 6, "_id&")`
retrieves the item ID of item 6 in the Zones named list. Named list indices run from zero to one less than the size of *list*, so this example accesses the seventh item in the list.

Available MM, ME, MD, OM, OD, ASC

❍ GetListItemStr

Get the value of a string property on an item in a named list.

Syntax `GetListItemStr (list, n, propertyName )`

Arguments *list* STRING
the named list to query

n LONG
the index of the item in the named list

PropertyName STRING
the name of the property. All integer properties end in &.

Returns STRING
the string value of the given property, for the given item in the given named list

Example `Brush = GetListItemStr ("Zones", 6, "_brush")`
retrieves the brush (fill style) of item 6 in the Zones named list. Named list indices run from zero to one less than the size of *list*, so this example accesses the seventh item in the list.

Available MM, ME, MD, OM, OD, ASC

❍ GetListSize

Get the number of items in a named list.

Syntax `rv = GisGetListSize (list )`

Arguments *list* STRING
the named list to query

Returns LONG INTEGER
the number of items in the given named list

Notes Named list indices run from zero to one less than the return value.

Available MM, ME, MD, OV, OM, OD, ASC

❍ GetNumAscClients

Get the number of Cadcorp SIS Active Server Component clients. This method can be used to monitor how busy the server is.

Syntax `rv = ControlName.GetNumAscClients ( )`

Returns SHORT INTEGER

the number of current Cadcorp SIS Active Server Component clients

Available ASC

❍ GetNumGeom

Get the number of geometry pieces in the current open item.

Syntax `rv = GisGetNumGeom ( )`

Returns LONG INTEGER

the number of geometry pieces in the current open item. Items such as blocks, groups, multi-line, and so on, can be made up of multiple geometry. Therefore this value can be greater than 0. Items such as lines, areas, shapes, and so on, are made by a single piece of geometry. Therefore this value would be 0.

Available MM, ME, MD, OM, OD, ASC

❍ GetNumNol

Get the number of named object libraries (NOLs) in use.

Syntax `rv = GisGetNumNol ( )`

Returns SHORT INTEGER

the number of currently loaded NOLs

Notes NOL indices run from zero to one less than the return value.

Available MM, ME, MD, OV, OM, OD, ASC

❍ GetNumSel

Get the number of items selected in the current SWD.

Syntax `rv = GisGetNumSel ( )`

Returns SHORT INTEGER

the number of selected items

Notes Selected item indices run from zero to one less than the return value.

Available MM, ME, MD, OV, OM, OD

❍ GetNumSwd

Get the number of different SWD files open in the current session.

Syntax `rv = GisGetNumSwd ( )`

Returns SHORT INTEGER

the number of open SWD files

Notes SWD file indices run from zero to one less than the return value.

Available MM, ME, MD

❍ GetNumWnd

Get the number of windows open.

Syntax `Rv = GisGetNumWnd ( )`

Returns SHORT INTEGER

the number of open SWD files

Notes Window indices run from zero to one less than the return value.

Available MM, ME, MD

❍ GetOverlayFilter

Get a copy of an overlay drawing filter, replacing any existing filter with the same name.

Syntax `GisGetOverlayFilter ( pos, filter )`

Arguments *pos* SHORT INTEGER

the position of the overlay in the overlays list whose drawing filter is to be copied

filter STRING

the named filter to create or replace

Example `GisGetOverlayFilter 0, "Properties"`

Notes The overlays start at position 0.

Available MM, ME, MD, OM, OD, ASC

❍ GetOverlayLocus

Get a copy of an overlay drawing locus, replacing any existing locus with the same name.

Syntax `GisGetOverlayLocus ( pos, locus )`

Arguments *pos* SHORT INTEGER

the position of the overlay in the overlays list whose drawing locus is to be copied

locus STRING

the named locus to create or replace

Available ME, MD, OD, ASC

❍ GetOverlaySchema

Get a copy of an overlay schema, replacing any existing schema with the same name.

Syntax `GisGetOverlaySchema ( pos, schema )`

Arguments *pos* SHORT INTEGER

the position of the overlay in the overlays list, whose schema is to be copied

schema STRING

the named schema to create or replace

Example `GisGetOverlaySchema 4, "Schema2"`

Available MM, ME, MD, OM, OD, ASC

❍ GetOverlayTheme

Get a copy of an overlay theme, replacing any existing theme with the same name.

Syntax `GisGetOverlayTheme ( pos, theme, nTheme )`

Arguments *pos* SHORT INTEGER
the position of the overlay in the overlays list whose theme is to be copied

theme STRING
the named theme to create or replace

nTheme SHORT INTEGER
the index of the theme, starting at 0. Use the `_nTheme&` property to find out the number of themes in an overlay.

Example `GisGetOverlayTheme 0, "NewContours", 0`

Available MM, ME, MD, OM, OD, ASC

❍ GetOverlayThemeLegend

Get an overlay theme legend as a Blob string within a projection. Label themes do not have legends.

Syntax `rv = GisGetOverlayThemeLegend (pos, nTheme, projection, fmt, precision )`

Arguments *pos* SHORT INTEGER
the position in the overlays list of the overlay whose theme legend is required

nTheme SHORT INTEGER
the index of the theme, starting at 0. Use the `_nTheme&` property to find out the number of themes in an overlay.

projection STRING
the named projection of the returned Blob string

fmt SHORT INTEGER

`SIS_BLOB_SIS`	Cadcorp SIS format
`SIS_BLOB_OGIS_WKT`	OpenGIS Well-Known-Text format

precision SHORT INTEGER
the precision of the Blob string. The Cadcorp SIS format recognises the values 16, 32, and 64 (or 0 for the default value of 64). The values 16, 32, and 64 specify the precision in bits of each co-ordinate. These values correspond to short integers, long integers, and double precision floating point respectively. Using a smaller precision will return shorter Blob strings.

Returns STRING
a string describing the overlay theme legend item in the chosen format. The string will be blank if no overlay theme legend exists. Cadcorp SIS format strings are encoded and cannot be interpreted except by Cadcorp SIS.

Example `LegendBlob = GisGetOverlayThemeLegend (0, 0, "A*PrjNatGrid", 0, 0, 64)`

returns the blob string for the first theme on the first overlay in the current window, within the National Grid projection.

Notes There is no limit on the length of Blob strings generated by Cadcorp SIS. There are however limits elsewhere: GisLink can handle strings of only a few kilobytes (this does not apply to the Cadcorp SIS Control); Cadcorp SIS datasets which read Blobs from ODBC data sources use a fixed size communication buffer (use the `_MaxBlobSize&` property of Blob style datasets to change the size of the communication buffer).

Available MM, ME, MD, OV, OM, OD, ASC

❍ GetPhotoWorldPos

Get the world position from a paper position within the current open photo item.

Syntax `rv = GisGetPhotoWorldPos (paperX, paperY )`

Arguments *paperX, paperY* DOUBLE
the paper position to convert to a world position

Returns STRING
a comma-delimited string containing the x, y, and z co-ordinates of the world position. Use `SplitPos` to get the x, y and z values themselves.

Available MM, ME, MD, OM, OD, ASC

❍ GetPos

Get a position from the user.

Syntax `rv = GisGetPos (x, y, z )`

Arguments *x, y, z* DOUBLE
the snapped position

Returns SHORT INTEGER

True the user snapped a position

False the user pressed the Escape key, or another command was selected

If the return value is true, any snapped item will be place in a special named list `*snapped`.

Available MM, ME, MD

❍ GetPosEx

Get a position from the user and return the action taken.

Syntax `rv = GisGetPosEx (x, y, z )`

Arguments *x, y, z* DOUBLE
the snapped position

Returns LONG INTEGER

`SIS_ARG_BACKSPACE`	the user pressed the Backspace key
`SIS_ARG_ENTER`	the user pressed the Enter key
`SIS_ARG_ESCAPE`	the user pressed the Escape key, or another command was selected
`SIS_ARG_POSITION`	the user snapped a position

If the return value is `SIS_ARG_POSITION`, any snapped item will be placed in a special named list `*snapped`.

Available MM, ME, MD

❍ GetPropertyDescription

Get the description of a property.

Syntax `rv = GisGetPropertyDescription ( prop )`

Arguments *prop* STRING
the property whose description is required

Returns STRING
the description of the property

Example `Desc = GisGetPropertyDescription "URN$"`

Available MM, ME, MD, OV, OM, OD, ASC

❍ GetSpatialReference

Get the spatial reference for the current open item within a spanned cube in a projection.

Syntax `rv = GisGetSpatialReference (projection, span )`

Arguments *projection* STRING
the named projection used in the calculation of the spatial reference

span DOUBLE
the span of the cube used in the calculation of the spatial reference

Returns STRING
a 16 character string which encodes a spatial reference

Example `SpatRef = GisGetSpatialReference ("*APrjNatGrid", 2000000)`
returns the spatial reference of the current item based on the National Grid projection, within a 2 000 000m cube

Notes To help speed up the display of point data stored in an external database, this method could be used to find the spatial reference of the points, to write them back to a column in the points table.

The spatial reference string encodes a position and a radius which together describe an extents circle. The span used when calculating a spatial reference must be big enough to cover all the possible co-ordinates. A smaller span will give spatial references with a finer resolution. The spatial reference does not affect the accuracy or resolution of the positions of the item it is associated with, only the accuracy of whether or not the

item is loaded in a particular view. The worst that can happen with a coarse resolution is that extra items are loaded.

Available ME, MD, OM, ASC

❍ GetSpatialReferenceFromExtent

Get the spatial reference for an extent within a spanned cube in a projection.

Syntax

```
GisGetSpatialReferenceFromExtent ( x1, y1, z1, x2, y2, z2,
projection, span )
```

Arguments

x1, y1, z1, x2, y2, z2 DOUBLE
the cuboid extents to be queried

projection STRING
the named projection used in the calculation of the spatial reference

Span DOUBLE
the span of the cube used in the calculation of the spatial reference

Returns STRING
a 16 character string which encodes a spatial reference

Notes The spatial reference string encodes a position and a radius which together describe an extents circle. The span used when calculating a spatial reference must be big enough to cover all of the possible co-ordinates. A smaller span will give spatial references with a finer resolution. The spatial reference does not affect the accuracy or resolution of the positions of the item it is associated with, only the accuracy of whether or not the item is loaded in a particular view. The worst that can happen with a coarse resolution is that extra items are loaded.

Available ME, MD, OD, ASC

❍ GetStr

Get the value of a string property.

Syntax

```
rv = GisGetStr (objectType, nObject, propertyName )
```

Arguments

objectType SHORT INTEGER
➲page 221, **Object types**

nObject LONG INTEGER
the index of the object type

propertyName STRING
the name of the property: all string properties end in $. The special properties `_properties$` and `_properties_edit$` are used to get lists of available properties for querying and editing respectively.

Returns STRING
the string value of the property

Example

```
rv = GisGetStr (SIS_OT_CURITEM, 0, "Details$")
```

Available MM, ME, MD, OV, OM, OD, ASC

❍ GetStrW

Get the Unicode value of a string property.

Syntax `rv = GisGetStrW (objectType, nObject&, propertyName )`

Arguments

objectType SHORT INTEGER

➲page 221, **Object types**

nObject LONG INTEGER

the index of the object type

propertyName STRING

the name of the property: all string properties end in $. The special properties `_properties$` and `_properties_edit$` are used to get lists of available properties for querying and editing respectively.

Returns STRING

the Unicode value of the string property

Example `rv = GisGetStrW (SIS_OT_CURITEM, 0, "_brush$")`

Available OV, OM, OD, ASC

❍ GetViewExtent

Get the visible extents of the current window.

Syntax `rv = GisGetViewExtent ( )`

Returns STRING

a comma-delimited string containing a pair of x, y, and z co-ordinates describing the visible extents of the current window. Use `SplitExtent` to get the pair of x, y, and z values themselves.

Available MM, ME, MD, OV, OM, OD, ASC

❍ GetViewPos

Get the real world position in the current view from a screen position and size in pixels.

Syntax `ControlName.GetViewPos ( x, y, z, xPos, yPos, xSize, ySize )`

Arguments

x, y, z DOUBLE

returned world position in regular units

xPos, yPos SHORT INTEGER

pixel position in the image to query

xSize, ySize SHORT INTEGER

size of image in pixels

Notes Cadcorp SIS Active Server Component does not keep track of the resolution of images that clients have requested with the `Render` method, so the image size must be re-specified.

Available ASC

❍ GetViewPosEx

Get the position in the current view from a position and size in pixels, as a comma-separated string.

Syntax `ControlName.GetViewPosEx ( xPos, yPos, xSize, ySize )`

Arguments *xPos, yPos* SHORT INTEGER
pixel position in the image to query

xSize, ySize SHORT INTEGER
size of image in pixels

Returns STRING
a comma-separated string containing the x, y, and z co-ordinates of the position. Use `SplitPos` to get the x, y, and z values themselves.

Notes Cadcorp SIS Active Server Component does not keep track of the resolution of images that clients have requested with the `Render` method, so the image size must be re-specified.

Available ASC

❍ GetViewPrj

Get a copy of the view projection, replacing any existing projection with the same name. The copy will be placed in the current library.

Syntax `GisGetViewPrj ( projection )`

Arguments *projection* STRING
the named projection to create or replace

Example `GisGetViewPrj "APrjWinkel1"`

Available MM, ME, MD, OM, OD, ASC

❍ ImportDataset

Import a dataset into the current SWD. This method creates an internal overlay which contains copies of all the items in the given dataset.

Syntax `GisImportDataset ( dataset, pos )`

Arguments *dataset* STRING
the filename of the dataset to import

pos SHORT INTEGER
the position in the overlays list at which to insert the overlay. If this argument specifies a position in the existing overlays, the new overlay will not replace the existing overlay at the given position, but will shuffle any other overlays down the list.

Example
```
GisImportDataset "c:\data\SP3217.ntf", 2
GisImportDataset "c:\data\property.bds", 1
```

Notes The internal overlay does not refer to the dataset file, so any changes to the dataset file will not be reflected in the internal overlay. The `InsertDataset` method should be used to refer to a dataset file.

Available MM, ME, MD, OM, OD, ASC

❍ ImportFeatureTable

Import a named feature table from a comma-separated file, replacing any existing feature table with the same name.

Syntax `GisImportFeatureTable ( ftable, filename )`

Arguments *ftable* STRING
the named feature table to create or replace

filename STRING
the name of the file from which to create the feature table

Example `GisImportFeatureTable "NEWLand-Line", "c:\data\land-line.csv"`

Available ME, MD, OD, ASC

❍ InsertDataset

Insert a dataset into the current SWD.

Syntax `GisInsertDataset ( dataset, pos )`

Arguments *dataset* STRING
the filename of the dataset to insert

pos SHORT INTEGER
the position in the overlays list at which to insert the dataset. If this argument specifies a position in the existing overlays, the new overlay will not replace the existing overlay at the given position, but will shuffle any other overlays down the list.

Example `GisInsertDataset "c:\data\counties.shp", 5`

Available MM, ME, MD, OV, OM, OD, ASC

❍ InsertFeatureCode

Insert a new feature code into the currently loaded feature table. Use `LoadFeatureTable` to load a feature table for editing.

Syntax `GisInsertFeatureCode ( fcode )`

Arguments *fcode* SHORT INTEGER
the feature code to be inserted

Example `GisInsertFeatureCode 1001`

Available ME, MD, OD, ASC

❍ InsertGeomPt

Insert a new vertex into the geometry of the current open item.

Syntax `GisInsertGeomPt ( nGeom, arclen, x, y, z )`

Arguments *nGeom* LONG INTEGER
the index of the geometry component, starting at 0. Use `GetNumGeom` to get the number of geometry components in an item.

arclen DOUBLE
the length along the geometry component to insert the new vertex

x,y,z DOUBLE
the position of the new vertex

Example `GisInsertGeomPt 0, 120, 431899, 928607, 0`

Available ME, MD, OD, ASC

❍ InsertOverlayTheme

Insert a copy of a named theme into an overlay in the current window.

Syntax `GisInsertOverlayTheme ( pos, theme, nTheme )`

Arguments *pos* SHORT INTEGER
the position of the overlay in the overlays list to which the theme is to be added

theme STRING
the named theme to add. This can be any named theme previously created or loaded from a named object library.

nTheme SHORT INTEGER
the position in the list of themes on the overlay at which to insert the theme. If this argument specifies a position in the existing themes, the new theme will not replace the existing theme at the given position, but will shuffle any other themes down the list.

Example `GisInsertOverlayTheme 0, "Contours", 2`

Notes Any named theme given will be copied, so any subsequent changes to the named theme will not be reflected in the overlay theme.

Available MM, ME, MD, OV, OM, OD, ASC

❍ InsertSchemaColumn

Insert a new column into the currently loaded schema. Use `LoadSchema` to load a schema for editing.

Syntax `GisInsertSchemaColumn ( formula, nColumn )`

Arguments *formula* STRING
the formula of the new column

nColumn SHORT INTEGER
the position in the columns list at which to insert the new column. If this argument specifies a position in the existing columns, the new column will not replace the existing column at the given position, but will shuffle any other columns down the list.

Notes This does not affect overlays that already use the schema being edited. The schema has to be reapplied using `StoreSchema` to save the changes, then `SetOverlaySchema`.

Available MM, ME, MD, OM, OD, ASC

❍ IsAscLicensed

Test whether the Cadcorp SIS Active Server Component is licensed.

Syntax `rv = ControlName.IsAscLicensed ( )`

Returns SHORT INTEGER

This method returns 0 if there are too many clients connected to the server. If this happens, the failure should be reported to the end-user, optionally suggesting that they try again later.

Available ASC

❍ IsoRoute

Find link and node items, which can be reached from a position, within a given cost. When the cost is related to time, this query is often called an isochrone.

Syntax `GisIsoRoute ( list, x, y, z, r, isoVal, formula, filter, locusNoGo )`

Arguments *`list`* STRING

the list of link and node items, which can be reached. Any link item whose mid-point can be reached will be put in the list.

`x, y, z` DOUBLE

the position to start from

`r` DOUBLE

the maximum distance from the start point to a link item. The topological algorithm will spread out from the closest link found. The distance from the point to the closest link is not included in the cost calculation. Ideally, the start point should be on a link item.

`isoVal` DOUBLE

the maximum cost to incur during route finding

`formula` STRING

the formula, or simple property, to use in the route finding calculation as the 'cost' of a link item. For example, using the simple property `_length#` will find all link and node items within a fixed distance from the point.

`filter` STRING

optionally specify a named filter, which all link items must pass to be considered as part of a route

`locusNoGo` STRING

optionally specify a named locus through which no route may pass. The named locus used will normally have its testing mode set to exclude any link items which cross it, using a call similar to the following:

```
SetLocusTest
("locus",SIS_TEST_NOCHANGE,SIS_TEST_CROSSING,SIS_TEST_NOCHANGE)
```

Example

```
GisIsoRoute "Routes", 2000, 1500, 10, 30, "_length#/((30*5280/3.2808)/60)",
"links", "NoGo"
```

This would normally be followed by:

```
GisCreateBoolean "Routes", SIS_BOOLEAN_OR
```

to create a multiline item of all the selected items.

Available ME, MD, OD, ASC

❍ JoinLines

Join line items within a tolerance.

Syntax `GisJoinLines ( list, tolerance )`

Arguments *`list`* STRING
the named list containing the line items to be joined

`tolerance` DOUBLE
the tolerance in current units within which to consider joins

Example `GisJoinLines "Lines", 1`
joins all the lines in the list lines that are within the tolerance of each other

Notes This method will join line items only within the given tolerance. The named list will be re-filled with the results of the join, and will have only one element if all of the original line items were successfully joined together.

Available ME, MD, OD, ASC

❍ LineTo

Draw a line from the current drawing position.

Syntax `GisLineTo ( x, y, z )`

Arguments *`x, y, z`* DOUBLE
the new line position

Notes The line is appended to the current line sequence, started by the last `MoveTo`, and extended using `BulgeTo`, `BezierTo` or this method. The current drawing position is at the end of the line, ie *x*, *y*, *z*, after calling this method.

Available MM, ME, MD, OV, OM, OD, ASC

❍ LoadFeatureTable

Load a named feature table for editing.

Syntax `GisLoadFeatureTable ( ftable )`

Arguments *`ftable`* STRING
the named feature table to load for editing

Example `GisLoadFeatureTable "NewFTable"`

Notes Use `""` to create a new, empty feature table. If a feature table is already loaded, loading another will lose any changes made to the first feature table. Use `StoreFeatureTable` to save changes to the currently loaded feature table.

Available ME, MD, OD, ASC

❍ LoadSchema

Load a named schema for editing.

Syntax `GisLoadSchema ( schema )`

Arguments `schema` STRING
the named schema to load for editing. Use `""` to create a new empty schema. If a schema is already loaded, loading another will lose any changes made to first schema. Use `StoreSchema` to save changes to the currently loaded schema.

Available MM, ME, MD, OM, OD, ASC

❍ LoadSwd

Replace the current SWD with the contents of an SWD file.

Syntax `ControlName.LoadSwd ( filename )`

Argument `filename` STRING
the filename of the SWD file to load

Notes The contents of the SWD file (ie overlay setup, saved view) are copied and the SWD file is not locked. Other Cadcorp SIS Controls that use the SWD which is being replaced will not automatically use the new SWD. The SWD must be saved first using the `SaveSwd` method.

Available OV, OM, OD, ASC

❍ LoadTheme

Load a named theme for editing.

Syntax `GisLoadTheme ( theme )`

Arguments `theme` STRING
the named theme to load for editing

Notes If a theme is already loaded, loading another will lose any changes made to first theme. Use `StoreTheme` to save changes to the currently loaded theme.

Available MM, ME, MD, OV, OM, OD, ASC

❍ LocusIntersect

Create a named locus by intersecting two existing loci, replacing any existing locus with the same name.

Syntax `GisLocusIntersect ( locusOut, locus1, locus2 )`

Arguments `locusOut` STRING
the named locus to create or replace

`locus1, locus2` STRING
the loci to intersect

Available ME, MD, OD, ASC

❍ MeasureAzimuth

Measure the azimuth between two positions.

Syntax `rv = GisMeasureAzimuth (x1, y1, z1, x2, y2, z2, datum )`

Arguments `x1, y1, z1` DOUBLE
the start point of the measurement

x2, y2, z2 DOUBLE
the end point of the measurement

datum STRING
the Geoid Datum to do the measuring within, eg WGS 84

Returns DOUBLE
the azimuth found

Available ME, MD, OD, ASC

❍ MeasureGreatCircle

Measure the Great Circle distance between two positions.

Syntax
```
rv = GisMeasureGreatCircle (x1, y1, z1, x2, y2, z2, datum )
```

Arguments *x1, y1, z1* DOUBLE
the start point of the measurement

x2, y2, z2 DOUBLE
the end point of the measurement

datum STRING
the geoid datum to measure within, eg WGS 84

Returns DOUBLE
the Great Circle distance found

Notes The measurement uses the following algorithm:

1. Transform (*x1*, *y1*, *z1*) and (*x2*, *y2*, *z2*) into the geoid datum.
2. Drop the two points onto the surface of the geoid (ie ignore heights).
3. Get the average latitude of the two points.
4. Get the radius of the geoid at the average latitude.
5. Get the angle separating the two points in radians.
6. Multiply the radius by the angle.

The radius of the geoid at the average latitude is approximated as follows:
```
radAve = radEquator + (radPole - radEquator) * sin(latAve)
```

Available ME, MD, OD, ASC

❍ MeasureRoute

Measure the best route between two positions.

Syntax
```
rv = GisMeasureRoute (x1, y1, z1, x2, y2, z2, formula, filter,
locusNoGo, bCreateLine )
```

Arguments *x1, y1, z1* DOUBLE
the start point of the route

x2, y2, z2 DOUBLE
the end point of the route

formula STRING
the formula, or item property, to use in the route finding calculation as the ‘cost’ of a link item. For example, using the simple property `_length#` property will find the shortest route, and using the formula `_length#/Speed#`, provided each link has a

user-defined `Speed#` property, will find the quickest route. Any formula can be used, although if a string formula is used it must be a string representation of a numeric value.

`filter` STRING

optionally specify a named filter which all link items must pass to be considered as part of the route

`locusNoGo` STRING

optionally specify a named locus which the route cannot pass through. The named locus used will normally have its testing mode set to exclude any link items which cross it, using a call similar to the following:

```
SetLocusTest("locus", SIS_TEST_NOCHANGE, SIS_TEST_CROSSING, _
  SIS_TEST_NOCHANGE)
```

`bCreateLine` SHORT INTEGER

True	create a new line item which is a copy of the found route
False	do not create a line item

Returns DOUBLE

the length of the route found

Example

```
Distance = GisMeasureRoute (100, 150, 0, 500, 300, 0, "_length#", filter, _
  "NoGo", True)
```

It would be possible to use `GetPosEx` to generate a snap position.

Available ME, MD, OD, ASC

❍ Message

Show a message in the message panel of the main frame window.

Syntax

```
GisMessage ( message )
```

Arguments *`message`* STRING

the message to display

Example

```
GisMessage "No Planning Application Selected"
```

displays the message “No Planning Application Selected” in the message panel of the main frame

Available MM, ME, MD

❍ MetreFromStr

Get a dimension in metres from a string, regardless of the units used in the string.

Syntax

```
rv = GisMetreFromStr ( s )
```

Arguments *`s`* STRING

the string representation of the dimension

Returns DOUBLE

Example

```
rv = GisMetreFromStr ("1km")
```

returns a floating point number 1000 from the string `"1km"`

Available MM, ME, MD, OM, OD

❍ MoveAxes

Set the position of the Cartesian axes.

Syntax `GisMoveAxes ( x, y, z )`

Arguments *x, y, z* DOUBLE
the new position of the axes origin. This position is relative to the current axes origin.

Available ME, MD, OD, ASC

❍ MoveList

Move, rotate, and scale editable items in a named list.

Syntax `GisMoveList ( list, x, y, z, a, s )`

Arguments *list* STRING
the named list containing the items to be moved, rotated, and scaled

x, y, z DOUBLE
the distances to move in the x, y, and z directions. Note that these values are relative, ie the items will be moved by these values from their current positions.

a DOUBLE
the rotation, in radians

s DOUBLE
scaling to apply

Example `GisMoveList "SeedPoints", 10, 10, 0, 0.79, 1`

Notes The distances (*x*, *y*, *z*) will be scaled and rotated by *s* and *a* respectively, before being applied to the items in the named list.

Available MM, ME, MD, OM, OD, ASC

❍ MoveTo

Set the current drawing position.

Syntax `GisMoveTo ( x, y, z )`

Arguments *x, y, z* DOUBLE
the new drawing position

Notes Line sequences are created using `BezierTo`, `BulgeTo`, and `LineTo`, each of which starts from the current drawing position. This method will flush any current line sequence, creating a line item and start a new line sequence.

Available MM, ME, MD, OV, OM, OD, ASC

❍ MultiRoute

Measure routes between items in a named list.

Syntax `rv = GisMultiRoute ( list, nStart, maxNum, maxVal, rSnap, formula, filter, locusNoGo )`

Arguments *list* STRING
the items to find routes between. The origin of each item in the named list is used as a potential target location.

nStart LONG INTEGER
the index within list of the place to start from. The route finder will find many routes, which all start from the *nStart* item.

maxNum LONG INTEGER
the maximum number of targets to find

maxVal DOUBLE
the maximum route to search within

rSnap DOUBLE
the maximum distance from the start point to a link item. The topological algorithm will spread out from the closest link found. The distance from the point to the closest link is not included in the cost calculation. Ideally, the start point should be on a link item.

formula STRING
the formula, or simple property, to use in the route finding calculation as the 'cost' of a link item. For example, using the simple length property will find the shortest route, and, using the formula `_length#/Speed#`, will find the quickest route, if each link has a user-defined `Speed#` property. Any formula may be used, although, if a string formula is used, it must be a string representation of a numeric value.

filter STRING
optionally specify a named filter, which all link items must pass to be considered as part of the route

locusNoGo STRING
optionally specify a named locus, which the route cannot pass through

Returns STRING
The return value is a string listing the distance from the starting item to other items in the named list. Each list element has the format (*index*, *val*), which describes the index of a target item within *list*, and the route value from *nStart* to that item.

Available ME, MD, OD, ASC

❍ NolCatalog

List all of the objects of a given class in all the named object libraries (NOLs).

Syntax
```
rv = GisNolCatalog ( aclass, bCurOnly )
```

Arguments *aclass* STRING
the class of named object to list:

`ABlock`	blocks
`ABrush`	brushes
`AColourset`	coloursets
`ADatum`	projection datums
`AFtable`	feature tables
`AGraticuleStyle`	graticule styles
`ALibItem`	stored items

`ALocus`	loci
`APen`	pens
`APrintTemplate`	print templates
`APrj`	projections
`ASchema`	schemas
`AShape`	shapes
`ATheme`	themes
`AToolBarDefn`	toolbar definitions
`AView`	views

`bCurOnly` SHORT INTEGER

True	return objects from the current NOL only
False	return objects from all NOLs

Returns STRING

a tab-separated list of named objects

Example `ObjList = GisNolCatalog ("ABrush", False)`

Notes The returned string is tab-separated because object names can contain spaces. The *`aclass`* strings `APstyle` and `ABstyle` are recognised for backwards compatibility (now `APen` and `ABrush` respectively).

Available MM, ME, MD, OV, OM, OD, ASC

❍ NolClose

Close a named object library (NOL) file, optionally saving any changes.

Syntax `GisNolClose ( nPos, bSave )`

Arguments *`nPos`* SHORT INTEGER

the position in the list of NOLs of the NOL to be closed

`bSave` SHORT INTEGER

True	save any changes to the NOL
False	discard any changes to the NOL

Example `GisNolClose 1, True`

Notes Non-file NOLs such as (temporary) cannot be removed from the list of currently loaded NOLs. Calling this method with an editable non-file NOL will empty it of all objects instead. Calling this method with a read-only non-file NOL will have no effect.

A NOL file cannot be closed if it is the default NOL. However, a non-file NOL will be emptied even if it is the default NOL. The default NOL can be set and queried using the `_DefaultNol$` system variable.

Available MM, ME, MD, OV, OM, OD, ASC

❍ NolCompact

Discard all old named object library (NOL) objects and defragment memory used by the NOL.

Syntax `GisNolCompact ( nPos )`

Arguments *nPos* SHORT INTEGER
the position in the list of NOLs of the NOL to be compacted

Available MM, ME, MD, OM, OD, ASC

❍ NolCreate

Create an empty named object library (NOL) file. This call will fail if the NOL file already exists.

Syntax `GisNolCreate ( filename )`

Arguments *filename* STRING
the name of the new NOL file. The new NOL is created in the current attached directory, unless a full path name is given. The new NOL can be added using `NolInsert`.

Example `GisNolCreate "c:\libraries\ewnol.nol"`

Available ME, MD, OD, ASC

❍ NolInsert

Insert a named object library (NOL) file.

Syntax `GisNolInsert ( filename, nPos, bReadOnly )`

Arguments *filename* STRING
the NOL file to insert

nPos SHORT INTEGER
the position in the list of NOLs at which to insert the new NOL

bReadOnly SHORT INTEGER

True	make the NOL read-only
False	make the NOL writable, ie new named objects can be created in the NOL

Example `GisNolInsert "c:\libraries\ewnol.nol", 0, True`

Available MM, ME, MD, OV, OM, OD, ASC

❍ NolOwn

Set the ownership of a named object library (NOL).

Syntax `GisNolOwn ( nPos, bOwn )`

Arguments *nPos* SHORT INTEGER
the position in the list of NOLs of the NOL whose ownership is to be changed

bOwn SHORT INTEGER

True attempt to get the ownership of the NOL

False disown the NOL

Example `GisNolOwn 2, True`

Notes NOLs cannot be disowned if they have been modified.

Available MM, ME, MD, OM, OD, ASC

❍ NolSave

Save a named object library (NOL) file.

Syntax `GisNolSave ( nPos )`

Arguments *nPos* SHORT INTEGER

the position in the list of NOLs of the NOL to be saved

Available MM, ME, MD, OM, OD, ASC

❍ OpenClosestItem

Open the item closest to a 3D position, within a specified search radius.

Syntax `GisOpenClosestItem ( x, y, z, r, stat, filter )`

Arguments *x, y, z* DOUBLE

the position to search from

r DOUBLE

the search radius

stat STRING

the status of items to be included in the search:

I all items

V visible, hittable, and editable items

H hittable and editable items

E editable items only

filter STRING

optionally specifies a named filter, which items must pass to be included in the search

Example `GisOpenClosestItem 100, 90, 0, 500, "E", "Areas"`

Available MM, ME, MD, OV, OM, OD, ASC

❍ OpenDatasetItem

Open the item in the named dataset with the given ID number. If the dataset is not open, Cadcorp SIS will open it and attempt to own it.

Syntax `GisOpenDatasetItem ( dataset, id )`

Arguments *dataset* STRING

the full filename of the dataset whose item is to be opened

id LONG INTEGER
the ID number of the item to be opened

Example `GisOpenDatasetItem "c:\data\planning\planning.bds", 103`
opens the item whose ID is 103, on the dataset c:\data\planning\planning.bds

Available MM, ME, MD, OM, OD, ASC

❍ OpenExistingDatasetItem

Open an item from an existing dataset with the given ID number.

Syntax `GisOpenExistingDatasetItem ( nDataset, id )`

Arguments *nDataset* LONG INTEGER
the serial number of the dataset whose item is to be opened ➲page222, **Serial numbers**

id LONG INTEGER
the ID number of the item to be opened

Example `GisOpenExistingDatasetItem 0, 103`

Available MM, ME, MD, OM, OD, ASC

❍ OpenFormulaItem

Open an item within a dataset which matches a formula.

Syntax `GisOpenFormulaItem ( nDataset, formula )`

Arguments *nDataset* LONG INTEGER
the serial number of the dataset whose item is to be opened ➲page222, **Serial numbers**

formula STRING
the formula to check dataset items against

Example `GisOpenFormulaItem 1, "URN$ = " & Chr(34) & "1998/190" & Chr(34)`

Notes The first item which matches the formula will be opened, so this method is best suited to finding an item using a unique value.

Available MM, ME, MD, OV, OM, OD, ASC

❍ OpenItem

Open the item in the current dataset with the given ID number.

Syntax `GisOpenItem ( id )`

Arguments *id* LONG INTEGER
the ID number of the item to be opened

Available MM, ME, MD, OV, OM, OD, ASC

❍ OpenList

Open an item from a named list.

Syntax `GisOpenList ( list, n )`

Arguments `list` STRING
the named list containing the item to be opened

`n` LONG INTEGER
the index of the item in the named list

Example `GisOpenList "Points", 0`

Notes Named list indices run from zero to one less than the list's size.

Available MM, ME, MD, OV, OM, OD, ASC

❍ OpenOverlayItem

Open the item on an overlay with the given ID number.

Syntax `GisOpenOverlayItem ( pos, id )`

Arguments `pos` SHORT INTEGER
the position in the overlays list of the overlay containing the item to be opened. The position of overlays starts at 0.

`id` LONG INTEGER
the ID number of the item to be opened

Available MM, ME, MD, OM, OD, ASC

❍ OpenSel

Open an item in the current selection list.

Syntax `GisOpenSel ( nsel )`

Arguments `nsel` SHORT INTEGER
the index of the item in the selection list

Notes Selected item indices run from zero to one less than the number of selected items.

Available MM, ME, MD, OV, OM, OD, ASC

❍ OwnDataset

Set the ownership of a dataset.

Syntax `GisOwnDataset ( dataset, bOwn )`

Arguments `dataset` STRING
the filename of the dataset whose ownership is to be changed

`bOwn` SHORT INTEGER

True	attempt to get the ownership of the dataset
False	disown the dataset

Example `GisOwnDataset "c:\data\property.bds", True`
This will give the user ownership of the dataset c:\data\property.bds. To get ownership, you will need to have editable access to the dataset.

Available MM, ME, MD, OM, OD, ASC

❍ Paste

Paste the contents of the Windows clipboard into the current overlay.

Syntax `GisPaste ( )`

Notes If the status of the current overlay is not editable, an editable internal overlay will be created.

Available MM, ME, MD, OV, OM, OD

❍ PasteFrom

Paste a file into the current SWD.

Syntax `GisPasteFrom ( filename, bLinked )`

Arguments *filename* STRING
the filename to paste

bLinked SHORT INTEGER

True create a link to the filename

False paste the contents of the filename

Example `GisPasteFrom "c:\data\picture.jpg", True`

Notes The item created will fill the current view extents, taking the different aspect ratios into account.

Available MM, ME, MD, OV, OM, OD

❍ PhotoGrid

Set the default grid on the current open photo item.

Syntax `GisPhotoGrid ( )`

Available MM, ME, MD, OM, OD, ASC

❍ PlaceGroup

Place the current open group item, at a given position, leaving it open.

Syntax `GisPlaceGroup ( x, y, z )`

Arguments *x, y, z* DOUBLE
the position at which to place the new group item

Notes Graphics created after `CreateGroup` will be drawn using the cursor as (0.0,0.0,0.0). If `CloseItem`, `Release` or `UpdateItem` is called, the graphics will be locked to the cursor for the user to place with two screen snaps (position and alignment). Calling this method before `CloseItem`, `Release`, or `UpdateItem` will explicitly position the graphics at the given position. The new group item is left open and will therefore continue to have any new graphics added to it.

Available MM, ME, MD, OM, OD, ASC

❍ PlacePrintTemplate

Place a print template in the current SWD, filling it with the previously composed window created by the `Compose` method.

Syntax `GisPlacePrintTemplate ( ptemplate, a, s )`

Arguments *`ptemplate`* STRING
the name of a print template object stored in a previously loaded named object library

`a, s` DOUBLE
the angle, in radians, and scale of the print template

Example `GisPlacePrintTemplate "A4 Portrait", 0.7583, 1250`

Available MM, ME, MD, OV, OM, OD, ASC

❍ Prompt

Set the prompt to be displayed in prompt panel of the main frame window when `GetPos` or `GetPosEx` is used.

Syntax `GisPrompt ( prompt )`

Arguments *`prompt`* STRING
the prompt to display

Example `GisPrompt "Please Select Planning Application"`
displays “Please Select Planning Application” in the prompt panel on the main window

Available MM, ME, MD

❍ RecallNolItem

Create an item from a stored named object library (NOL) item.

Syntax `GisRecallNolItem ( item )`

Arguments *`item`* STRING
the NOL item to recall

Available ME, MD, OD, ASC

❍ RecallNolView

Recall a named view from a named object library (NOL).

Syntax `GisRecallNolView ( view )`

Arguments *`view`* STRING
the NOL view to recall

Available MM, ME, MD, OM, OD, ASC

❍ Redraw

Redraw a window or windows.

Syntax `GisRedraw ( redrawcode )`

Arguments `redrawcode` SHORT INTEGER

`SIS_CURRENTWINDOW`	redraw the current window only
`SIS_CURRENTSWD`	redraw all windows, which contain the current SWD
`SIS_ALLWINDOWS`	redraw all windows
`SIS_QUICK_REDRAW`	optional flag which causes map windows to use their cached bitmap, instead of doing a full regeneration. For example, using `SIS_CURRENTSWD + SIS_QUICK_REDRAW` after moving items in a named list will repaint the map image, to heal the underlying graphics without re-reading the displayed data from file.

Available MM, ME, MD, OV, OM, OD

❍ RedrawExtent

Redraw windows or part of a window.

Syntax `GisRedrawExtent ( redrawcode, x1, y1, z1, x2, y2, z2 )`

Arguments `redrawcode` SHORT INTEGER

`SIS_CURRENTWINDOW`	redraw the current window only
`SIS_CURRENTSWD`	redraw all windows which contain the current SWD
`SIS_ALLWINDOWS`	redraw all windows
`SIS_QUICK_REDRAW`	optional flag which makes map windows use their cached bitmap, instead of doing a full 'regeneration'. For example, use `SIS_CURRENTSWD + SIS_QUICK_REDRAW` after moving items in the named list `*Sprites`.

`x1, y1, z1, x2, y2, z2` DOUBLE
the cuboid extents to be redrawn

Available MM, ME, MD, OM, OD

❍ RefreshDataset

Make sure that a dataset is up to date.

Syntax `GisRefreshDataset ( nDataset )`

Arguments `nDataset` LONG INTEGER
the serial number of the dataset to refresh ➲page 222, **Serial numbers**

Example `GisRefreshDataset 1`
refreshes the dataset whose serial number is 1. This will get the most recently saved copy from disk.

Available MM, ME, MD, OM, OD, ASC

❍ RefreshDbTable

Refresh a named table from a database.

Syntax `GisRefreshDbTable ( table )`

Arguments *table* STRING
the named table to refresh

Example `GisRefreshDbTable "Planning"`

Available MM, ME, MD, OM, OD, ASC

❍ RegisterGroupType

Register a sub-class of a group, which the users cannot directly modify. The registered group sub-class is used by `CreateGroup`.

Syntax `GisRegisterGroupType ( clsName )`

Arguments *clsName* STRING
the class of the group to register

Available MM, ME, MD, OM, OD, ASC

❍ RegisterTrigger

Register a trigger.

Syntax `GisRegisterTrigger ( triggerEvent, caption )`

Arguments *triggerEvent* STRING
the event that triggers the button press

caption STRING
the caption of the button to be pressed

Example `GisRegisterTrigger "AComLineEx::KeyEnter", "DrawLineEnter"`
pushes the button on the startup form with the caption `DrawLineEnter` when an AComLineEx::KeyEnter event occurs

Notes Every command in the system has a trigger which can be monitored by Visual Basic. One-shot commands have Succeeded or Failed triggers. Callback commands have-Snap, Keyback, KeyEnter, KeyTab and End triggers. These triggers can be used to set off an event. Once the trigger is used, you can unregister it by giving it the caption `""`.

Available MM, ME, MD

❍ Release

Return control to Cadcorp SIS from a Visual Basic program. When the user invokes a Visual Basic registered command, they are locked out of Cadcorp SIS menus until this routine is called. This does not terminate the conversation.

Syntax `GisRelease ( )`

Notes When the user selects a GisLink command, previously added using `AddCommand`, the Cadcorp SIS application will not respond to mouse clicks or keyboard presses until this method, or the `ReleaseNA` method, is called.

Available MM, ME, MD

❍ ReleaseNA

Perform a Release but do not make Cadcorp SIS the active application.

Syntax `GisReleaseNA ( )`

Notes When the user selects a GisLink command, previously added using `AddCommand`, the Cadcorp SIS application will not respond to mouse clicks or keyboard presses until this method, or the `Release` method, is called. For example, use `ReleaseNA` after the `DrawList` method to prevent the display from being refreshed.

Available MM, ME, MD

❍ RemoveAtt

Remove an attribute from the current open item.

Syntax `GisRemoveAtt ( mnem )`

Arguments *mnem* STRING
the name of the user-defined attribute to be removed

Available MM, ME, MD, OM, OD, ASC

❍ RemoveCommand

Remove an application-defined command previously added using `AddCommand`.

Syntax
```
SisControl.RemoveCommand ( menu )
GisRemoveCommand ( menu )
```

Arguments *menu* STRING
the value of menu in the method `AddCommand`

Example `Sis.RemoveCommand "Query#Display Property Details"`
removes the Display Property Details command from the Query menu
`GisRemoveCommand "Query|Display Property Details"`
removes the Display Property Details command from the Query menu

Available MM, ME, MD, OM, OD

❍ RemoveFeatureCode

Remove an existing feature code from the currently loaded feature table.

Syntax `GisRemoveFeatureCode ( fcode )`

Arguments *fcode* SHORT INTEGER
the feature code to remove

Notes This method does not delete any children of the given feature code.
Use `LoadFeatureTable` to load a feature table for editing.

Available ME, MD, OD, ASC

❍ RemoveOverlay

Remove an overlay from the current SWD, deleting it if it is an internal overlay. If the dataset has been modified, all edits since the last save will be lost.

Syntax `GisRemoveOverlay ( pos )`

Arguments *pos* SHORT INTEGER
the position in the overlays list of the overlay to be removed

Notes The overlay will be deleted if it is an internal overlay.

Available MM, ME, MD, OV, OM, OD, ASC

❍ RemoveOverlayTheme

Remove a theme from an overlay in the current SWD.

Syntax `GisRemoveOverlayTheme ( pos, nTheme )`

Arguments *pos* SHORT INTEGER
the position of the overlay in the overlays list from which the theme is to be removed

nTheme SHORT INTEGER
the index of the theme, starting at 0. Use the `_nTheme&` property to find out the number of themes in an overlay.

Example `GisRemoveOverlayTheme 4, 2`
removes the third theme, on the fifth overlay, in the current window

Available MM, ME, MD, OV, OM, OD, ASC

❍ RemoveProperty

Remove a property from an object.

Syntax `GisRemoveProperty ( objectType, nObject, propertyName )`

Arguments *objectType* SHORT INTEGER
➲page221, **Object types**

nObject LONG INTEGER
the index of the object type

propertyName STRING
the property to remove. Only user-defined attributes can be removed in this way.

Example `GisRemoveProperty SIS_OT_CURITEM, 0, "Value#"`

Notes This method is similar to the `RemoveAtt` method, but can operate on all object types.

Available ME, MD, OD, ASC

❍ RemoveSchemaColumn

Remove an existing column from the currently loaded schema.

Syntax `GisRemoveSchemaColumn ( nColumn )`

Arguments *nColumn* SHORT INTEGER
the schema column to remove

Notes This does not effect overlays that already use the schema being edited. The schema has to be reapplied using `StoreSchema`, to save changes, then `SetOverlaySchema`.

Use `LoadSchema` to load a schema for editing.

Available MM, ME, MD, OM, OD, ASC

❍ Render

Create an image of a given size to be displayed by the client browser.

Syntax `ControlName.Render ( xSize, ySize. strFormat )`

Arguments *xSize, ySize* SHORT INTEGER

the size in pixels to make the image

strFormat STRING

the format you want the image to be rendered in. The valid formats are:

image/jpeg	JPEG is understood by almost all browsers
image/png	PNG is understood by latest versions of most browsers
image/x-png	another way of using PNG
image/x-MS-bmp	Windows BMP is understood by most Microsoft browsers

Cadcorp SIS ASC will render the current view into a raster image and then feed the image into the HTML data stream. If the user's browser supports the format, they will see the image.

Example `SIS.Render 512, 512, "image/jpeg"`

creates a JPEG image 512 by 512 pixels to be displayed in the client browser

Notes Choose a format that is supported by the target users' browsers. If they have a more recent browser, consider using PNG. If they have older browsers, choose JPEG.

Available ASC

❍ ReorderOverlay

Change the order of overlays.

Syntax `GisReorderOverlay ( oldPos, newPos )`

Arguments *oldPos* SHORT INTEGER

the position in the list of overlays of the overlay to be reordered

newPos SHORT INTEGER

the position in the overlays list at which to re-insert the overlay. If this argument specifies a position in the existing overlays, the reordered overlay will not replace the existing overlay at the given position, but will shuffle any other overlays down the list.

Example `GisReorderOverlay 4, 0`

Available MM, ME, MD, OM, OD, ASC

❍ RubberSheetRaster

Apply the current rubber sheet transformation to the currently open bitmap item.

Syntax `GisRubberSheet ( )`

Available ME, MD, OD, ASC

❍ SaveBitmap

Save the current open bitmap item to a file.

Syntax `GisSaveBitmap ( filename, typeBitmap )`

Arguments *`filename`* STRING
the named of the saved file

`typeBitmap` SHORT INTEGER
the bitmap format to use:

`SIS_SAVEBMP_BMP`	a Windows bitmap of the same colour depth as the system graphics
`SIS_SAVEBMP_DITHERBMP`	an 8-bit, 256 colour Windows bitmap
`SIS_SAVEBMP_JPG`	24-bit, 16.7 million colour JPEG
`SIS_SAVEBMP_PNG`	8-bit, 256 colour or 24-bit, 16.7 million colour Portable Network Graphics file
`SIS_SAVEBMP_TIFF`	Tagged Image File Format
`SIS_SAVEBMP_TIFF_GP4`	Group 4 (fax) compressed black-and-white TIFF
`SIS_SAVEBMP_TIFF_PACKBITS`	Packbits compressed TIFF

Example `GisSaveBitmap "d:\dataphoto.bmp", SIS_SAVEBMP_BMP`

Available MM, ME, MD, OM, OD, ASC

❍ SaveDataset

Save a dataset if it has been modified.

Syntax `GisSaveDataset ( dataset )`

Arguments *`dataset`* STRING
the name of the dataset to save

Example `GisSaveDataset "c:\data\property.bds"`

Available MM, ME, MD, OM, OD, ASC

❍ SaveSwd

Save the current SWD to a file.

Syntax `ControlName.SaveSwd ( filename )`

Arguments *`filename`* STRING
the filename of the SWD file to save

Example `SIS.SaveSwd "c:\projects\BASEMAP.SWD"`

Notes This is equivalent to the `SwdSaveAs` method, available when using GisLink with Cadcorp SIS applications.

Available OV, OM, OD

❍ Scan

Scan for items in the current window, storing any found in a named list.

Syntax `rv = GisScan ( list, stat, filter, locus )`

Arguments

list STRING
the named list in which to store any items found

stat STRING
the status of items to be included in the search

I all items

V visible, hittable, and editable items

H hittable and editable items

E editable items only

filter STRING
optionally specify a named filter which items must pass to be included in the scan

locus STRING
optionally specify a named locus which items must fall within to be included in the scan

Returns LONG INTEGER
the number of items found, ie the number of items placed in the named list

Example `NItems = GisScan ("ItemsFound", "H", "Conservation", "Talbot")`

Available MM, ME, MD, OV, OM, OD, ASC

❍ ScanDataset

Scan a dataset for items, storing any found in a named list.

Syntax `rv = GisScanDataset ( list, nDataset, filter, locus )`

list STRING
the named list in which to store any items found

nDataset LONG INTEGER
the serial number of the dataset to be scanned. The number can be obtained from the `_nDataset&` property of an overlay, or from the `GetDataset`, `GetDatasetContainer`, or `FindExternalDataset` methods.

filter STRING
optionally specify a named filter which items must pass to be included in the scan

locus STRING
optionally specify a named locus which items must fall within to be included in the scan

Returns LONG INTEGER
the number of items found, ie the number of items placed in the named list

Example `NFound = GisScanDataset ("ItemsFound", 4, "Conservation", "Talbot")`

Available MM, ME, MD, OV, OM, OD, ASC

❍ ScanGeometry

Find items which satisfy a condition with the current open item.

Syntax

```
rv = GisScanGeometry ( list, geomTest, geomMode, filter, locus )
```

Arguments

list STRING
the named list in which to store any items found

geomTest LONG INTEGER
the geometry test to use ➲page219, **Geometry tests**

geomMode SHORT INTEGER
the geometry test mode to use:

SIS_GM_ORIGIN	items whose origin (a single point) must pass the testing method with the selected item
SIS_GM_EXTENTS	items whose rectangular extents must pass the testing method with the selected item
SIS_GM_GEOMETRY	items whose geometry must pass the testing method with the selected item

filter STRING
optionally specify a named filter which items must pass to be included in the scan

locus STRING
optionally specify a named locus which items must pass to be included in the scan

Returns LONG INTEGER
the number of items found, ie the number of items placed in the named list

Example

```
NItems = GisScanGeometry ("Points", SIS_GT_CONTAIN, SIS_GM_GEOMETRY, _
"PointsOnly", "Talbot')
```

Notes
Only hittable or editable area items are scanned.

Available
MM, ME, MD, OM, OD, ASC

❍ ScanList

Scan a named list for items matching a named filter and/or named locus.

Syntax

```
rv = GisScanList ( listOut, listIn, filter, locus )
```

Arguments

listOut STRING
the named list in which to store any items found

listIn STRING
the named list to be scanned

filter STRING
optionally specify a named filter which items must pass to be included in the scan

locus STRING
optionally specify a named locus which items must fall within to be included in the scan

Returns LONG INTEGER
the number of items found, ie the number of items placed in the named list

Example
```
NItems = GisScanList ("Points", 0, "PointsOnly", "Talbot")
NFound = GisScanList ("ListOut", "ListIn", "Class Filter.Area", " ")
```

Notes The `listOut` argument may be the same as the `listIn` argument to re-use the existing named list.

Available MM, ME, MD, OV, OM, OD, ASC

❍ ScanOverlay

Scan an overlay for items, storing any found in a named list.

Syntax
```
rv = GisScanOverlay ( list, pos, filter, locus )
```

Arguments

`list` STRING
the named list in which to store any items found

`pos` SHORT INTEGER
the position in the overlays list of the overlay to be scanned

`filter` STRING
optionally specify a named filter which items must pass to be included in the scan. The filter specified, if any, will be used in addition to any overlay filter.

`locus` STRING
optionally specify a named locus which items must fall within to be included in the scan. The locus specified, if any, will be used in addition to any overlay locus.

Returns LONG INTEGER
the number of items found, ie the number of items placed in the named list

Example
```
NFound = GisScanOverlay ("OverlayList", 0, "BandB", "")
```
all the items on the overlay in position 0 in the current window that pass the filter BandB will be placed in the list OverlayList

Available MM, ME, MD, OV, OM, OD, ASC

❍ ScanPointContainers

Find area items which contain a point.

Syntax
```
rv = GisScanPointContainers ( list, x, y, z, filter, locus )
```

Arguments

`list` STRING
the named list in which to store any area items found

`x, y, z` DOUBLE
the position at which to scan

`filter` STRING
optionally specify a named filter which area items must pass to be included in the scan

`locus` STRING
optionally specify a named locus which area items must fall within to be included in the scan

Returns LONG INTEGER
The number of area items found, ie the number of area items placed in the named list.

Example
```
NFound = GisScanPointContainers ("Areas", 1000, 1500, 0, "Conservation", _
  "Talbot")
```

Notes This method will find any item which is a closed area, eg area, bitmap, QZone, polygon, and surface if the surface item is a TIN.
Only hittable or editable area items are scanned.

Available ME, MD, OD, ASC

❍ ScrollView

Scroll the current window by a number of pixels.

Syntax `GisScrollView ( dx, dy )`

Arguments *dx* SHORT INTEGER
the number of pixels to scroll horizontally

dy SHORT INTEGER
the number of pixels to scroll vertically

Example `GisScrollView 64, 16`

Available MM, ME, MD, OV, OM, OD, ASC

❍ SelectAll

Select all hittable and editable items.

Syntax `GisSelectAll ( )`

Available MM, ME, MD, OV, OM, OD

❍ SelectItem

Toggle the selection status of the current open item, ie add it to the selection list if it is not selected, and remove it from the selection list if it is selected.

Syntax `GisSelectItem ( )`

Notes The item will be selected only if it appears on an overlay with status hittable or editable.

Available MM, ME, MD, OV, OM, OD

❍ SelectList

Toggle the selection status of items in a named list, ie add an item to the selection list if it is not selected, and remove it from the selection list if it is selected.

Syntax `GisSelectList ( list )`

Arguments *list* STRING
the named list containing the items to have their selection status toggled

Notes The items will be selected only if they appear on an overlay with status hittable or editable.

Available MM, ME, MD, OV, OM, OD

❍ SendPrint

Print the current window.

Syntax `GisSendPrint ( driver, device, outputName, forceColour, fStretch )`

Arguments *driver, device, output* STRING
the configuration of the printer

forceColour SHORT INTEGER

`SIS_PRINTCAPS_QUERY`	query the printer driver to get colour capabilities of printer
`SIS_PRINTCAPS_MONO`	force output to monochrome (pens are black or white, brushes and bitmaps are gray)
`SIS_PRINTCAPS_COLOUR`	assume printer can handle 24-bit colours

fStretch DOUBLE
the scaling factor to apply to the print to make it fit onto the printer paper

Example
```
GisSendPrint Printer.DriverName, Printer.DeviceName, Printer.Port, _
  SIS_PRINTCAPS_QUERY, 1
```
sets the parameters for driver, device, output, to match those of the default printer

Notes This method of printing is intended for use only in the Cadcorp SIS Control. The values of the *driver*, *device*, and *output* arguments should be found from the printing facilities available in the container application developer environment (*driver* will typically be `winspool`). If each of the *driver*, *device*, and *output* arguments are blank strings, the properties in `SIS_OT_PRINTER` will be used. If the `_device$`, `_driver$`, and `_output$` properties of `SIS_OT_PRINTER` are empty, this method will display the same sequence of dialogs as the application's Print command. This technique will not work with GisLink customisations, which should start the Print command directly. When the dialogs are shown, the *forceColour* and *fStretch* arguments are ignored.

Available MM, ME, MD, OV, OM, OD, ASC

❍ Set3DView

Set the eye and look position in a 3D view.

Syntax `GisSet3DView ( bSetView0, xEye, yEye, zEye, xLook, yLook, zLook )`

Arguments *bSetView0* SHORT INTEGER

True	set the 'view 0' eye and look positions (see Notes)
False	set the current eye and look positions

xEye, yEye, zEye DOUBLE
the eye position

xLook, yLook, zLook DOUBLE
the look position

Notes 'View 0' is used in two places: it is the view that the Reset 3D View command will return to; and the 'view to' look position is always used as the look position when the 3D view is in Model mode (`SIS_3DMODE_MODEL`).

Available MD, OD

❍ SetAxesAngle

Rotate the axes to an angle.

Syntax `GisSetAxesAngle ( a )`

Arguments *a* DOUBLE
the axes rotation angle, in degrees. This value is absolute.

Example `GisSetAxesAngle 90`

Available ME, MD, OD, ASC

❍ SetAxesGrid

Show/hide a grid of points or lines, with optional snapping.

Syntax `GisSetAxesGrid ( x, y, bShowGrid, bShowPoints, bAllowSnap )`

Available *x, y* DOUBLE
the grid spacing

bShowGrid SHORT INTEGER

True show the grid

False hide the grid

bShowPoints SHORT INTEGER

True display the grid as points

False display the grid as lines

bAllowSnaps SHORT INTEGER

True allow snapping on the grid points

False disallow snapping on the grid points

The G snapcode can be used to force a snap to a grid point. Grid snapping can be enabled without displaying the grid.

Example `GisSetAxesGrid 10000, 10000, True, False, True`
creates a 10k grid, displaying a line grid with snaps

Available ME, MD, OD, ASC

❍ SetAxesNormal

Reset the axes to the origin and orientation of the underlying projection.

Syntax `GisSetAxesNormal ( )`

Available ME, MD, OD, ASC

❍ SetAxesPrj

Set the current axes projection.

Syntax `GisSetAxesPrj ( projection )`

Arguments *projection* STRING

the named projection to use. This can be any named projection previously created using `DefineNolPrjLatLon`, `DefineNolPrjTm`, or loaded from a named object library.

Example `GisSetAxesPrj "UserPrj"`

Notes The axes projection sets up the x, y, z co-ordinates used in the API routines and the co-ordinate system in the user interface. In the API, the units are always metres and degrees. However, in the user interface the units can be changed (eg miles and radians). When the current axes are changed, the view of the current map window may also be changed to be compatible. If the current window is not a map window, or there is no current window, the default axes projection is set instead.

Available MM, ME, MD, OM, OD, ASC

❍ SetCombinedFilterClause

Add a clause to a named combined class/property filter.

Syntax `GisSetCombineFilterClause ( filter, aclass, flag, clause )`

Arguments *filter* STRING

the named combined class/property filter to edit, previously created using `CreateCombinedFilter`

aclass STRING

the item class which this clause will affect

flag SHORT INTEGER

`SIS_CLASSEXCLUDE`	exclude from filter
`SIS_CLASSINCLUDE`	include in filter

Each of these flags may have the modifier `SIS_OPENBRANCH` added to them, to open a class tree branch in readiness for modifying the filter behaviour of a sub-class of the item class specified in the *aclass* argument.

clause STRING

the property clause, eg `_closed&=-1`

Example
```
GisSetCombineFilterClause "Polygons>50", "Area", SIS_CLASS_EXCLUDE, _
  "_area#<50"
```
excludes areas whose `_area#` value is less than 50 from the filter Polygons>50

Notes The item class name in the *aclass* argument, which reflects the Cadcorp SIS C++ class name, is not necessarily the same as that which appears in the Cadcorp SIS user interface, which is translatable. In particular the polygon and chain item classes should be specified as SeedArea and SeedChain respectively. The class name to use in this method is stored in the `_class$` item property. The translatable class name is stored in the `_classLocal$` property.

Available MM, ME, MD, OM, OD, ASC

❍ SetCommandBitmap

Set the image displayed on Cadcorp SIS menus for an application-defined command.

Syntax `GisSetCommandBitmap ( comname, bitmap )`

Arguments *comname* STRING
the name of the command to which the bitmap applies

bitmap STRING
the name and location of the bitmap. The bitmap must measure 16 pixels (width) by 15 pixels (height).

Example `GisSetCommandBitmap ("&Construct|&Hexagon","c:\VBProjects\Images\Hex.bmp")`
sets the bitmap image on the menu item of the Construct>Hexagon custom command, previously created by `GisAddCommand`

Notes This method is available only to GisLink applications. If a bitmap is not assigned to the menu item, Cadcorp SIS will display the default 'tool' image.

Available MM, ME, MD

❍ SetCoordUnits

Change the preferred angle, linear, area or volume units used in the user interface.

Syntax `GisSetCoordUnits ( ndim, units, places )`

Arguments *ndim* SHORT INTEGER
the unit type to change

`SIS_UNIT_ANGLE`
`SIS_UNIT_LINEAR`
`SIS_UNIT_AREA`
`SIS_UNIT_VOLUME`

units SHORT INTEGER
the units to use

Angle	Linear	Area	Volume
SIS_UNITA_DEGREES	SIS_UNIT1_M	SIS_UNIT2_M	SIS_UNIT3_M
SIS_UNITA_RADIANS	SIS_UNIT1_MM	SIS_UNIT2_MM	SIS_UNIT3_MM
SIS_UNITA_DMS	SIS_UNIT1_CM	SIS_UNIT2_CM	SIS_UNIT3_CM
	SIS_UNIT1_KM	SIS_UNIT2_KM	SIS_UNIT3_LITRE
	SIS_UNIT1_FEET	SIS_UNIT2_FEET	SIS_UNIT3_FEET
	SIS_UNIT1_INCHES	SIS_UNIT2_INCHES	SIS_UNIT3_INCHES
	SIS_UNIT1_IMPERIAL	SIS_UNIT2_YARDS	SIS_UNIT3_YARDS
	SIS_UNIT1_YARD	SIS_UNIT2_ACRE	SIS_UNIT3_GALLON_IMP
	SIS_UNIT1_FATHOM	SIS_UNIT2_HECTARE	SIS_UNIT3_GALLON_US
	SIS_UNIT1_MILE	SIS_UNIT2_TUBO	
	SIS_UNIT1_NAUTMILE	SIS_UNIT2_MILE	
		SIS_UNIT2_NAUTMILE	

places SHORT INTEGER
the number of decimal places to display, in the range 0 to 15

Example `GisSetCoordUnits SIS_UNIT_AREA, SIS_UNIT2_HECTARE, 2`
sets the units for area measurement to Hectares with two decimal places

Available MM, ME, MD, OV, OM, OD

❍ SetDatasetPrj

Set a dataset projection.

Syntax `GisSetDatasetPrj ( nDataset, projection )`

Arguments *nDataset* LONG INTEGER
the serial number of the dataset whose projection is to be set ➲page222, **Serial numbers**

projection STRING
the named projection to use

Example `GisSetDatasePrj 1, "*APrjWinkel1"`
sets the projection of the dataset whose serial number is 1 to *APrjWinkel1

Notes Use this method carefully. Changing a dataset projection will result in all items in a dataset being repositioned in the world. Typically, use it only when a dataset is first created.

Available MM, ME, MD, OM, OD, ASC

❍ SetDefaultPrj

Set the default viewing and co-ordinate system projections.

Syntax `GisSetDefault ( projection )`

Arguments *projection* STRING
the named projection to use

Example `GisSetDefault "*APrjMercator"`

Notes This method always changes the default projections, which will affect future windows. If the current window is a map window, its viewing projection and co-ordinate system will also be changed.

Available MM, ME, MD, OV, OM, OD

❍ SetFlt

Set the value of a floating point property.

Syntax `GisSetFlt ( objectType, nObject, propertyName, value )`

Arguments *objectType* SHORT INTEGER
➲page221, **Object types**

nObject LONG INTEGER
the index of the object type

propertyName STRING
the name of the property: all floating point properties end in #

value DOUBLE
the new floating point value of the property

Notes When using `SIS_OT_CURITEM` for *`objectType`* , if no property exists of the name *`propertyName`*, and *`propertyName`* does not start with the underscore (_) character, a user-defined attribute will be created, with the name *`propertyName`*$, and the value *`value`*.

Example `GisSetFlt SIS_OT_OVERLAY, 0, "_scale#", 2500`

Available MM, ME, MD, OV, OM, OD, ASC

❍ SetForegroundWindow

Make a Microsoft Visual Basic form the foreground window.

Syntax `GisSetForegroundWindow ( hWndForm )`

Arguments *`hWndForm`* SHORT INTEGER
the Microsoft Visual Basic form to make the foreground window

Example `GisSetForegroundWindow 1`
makes the window whose window handle is 1 the foreground window

Notes This method is intended for use only in 16-bit versions of Microsoft Visual Basic (3.0 and 4.0). In Microsoft Visual Basic 4.0 32-bit and Microsoft Visual Basic 5.0 and 6.0, use the Win32 API call `SetForegroundWindow` instead.

Available MM, ME, MD

❍ SetGazetteerView

Find and zoom to a location using a plug-in gazetteer.

Syntax `GisSetGazetteerView ( clsGazetteer, params )`

Arguments *`clsGazetteer`* STRING
the name of the plug-in gazetteer

`params` STRING
the parameters for the search

Available MM, ME, MD, OV, OM, OD, ASC

❍ SetGeomPt

Set the position of a vertex in the current open item.

Syntax `GisSetGeomPt ( nGeom, nPt, x, y, z )`

Arguments *`nGeom`* LONG INTEGER
the index of the geometry component, starting at 0. Use `GetNumGeom` to get the number of geometry components in an item.

`nPt` LONG INTEGER
the index of the vertex, starting at 0. Use `GetGeomNumPt` to get the number of vertices in a geometry component.

`x,y,z` DOUBLE
the new position of the given vertex

Example `GisSetGeomPt 0, 4338991, 936006, 0`

Available MM, ME, MD, OM, OD, ASC

❍ SetGeomSegBulge

Set the bulge value of a segment in the current open item.

Syntax `GisSetGeomSegBulge ( nGeom, nSeg, bulge )`

Arguments *nGeom* LONG INTEGER
the index of the geometry component, starting at 0. Use `GetNumGeom` to get the number of geometry components in an item.

nSeg LONG INTEGER
the index of the segment within the current open item. The indices run from 1 to `GetGeomNumSeg` – 1, eg a line item with two points has one segment, index 1.

bulge DOUBLE
the new bulge value of the given segment. The bulge factor is the tangent of one quarter of the swept angle. A bulge factor of 0.0 implies a straight segment.

Example `GisSetGeomSegBulge 0, 3, 0.25`

Available MM, ME, MD, OM, OD, ASC

❍ SetGridItemValue

Set the value in a cell of the current open grid item. Grid cell values can be queried using `GetGridItemValue`.

Syntax `GisSetGridItemValue ( x, y, z, v )`

Arguments *x, y, z* DOUBLE
the position at which to set the grid item value

v DOUBLE
the new value for the grid cell

Example `GisSetGridItemValue 100, 100, 0, 50`

Available MD, OD, ASC

❍ SetInt

Set the value of an integer property.

Syntax `GisSetInt ( objectType, nObject, propertyName, value )`

Arguments *objectType* SHORT INTEGER
➲page221, **Object types**

nObject LONG INTEGER
the index of the object type

propertyName STRING
the name of the property: all integer properties end in &

value LONG INTEGER
the new integer value of the property

Example `GisSetInt SIS_OT_OVERLAY, 0, "_precision&", 32`

Notes When using `SIS_OT_CURITEM` for *objectType* , if no property exists of the name *propertyName*, and *propertyName* does not start with the underscore (_) character, a

user-defined attribute will be created, with the name *propertyName&*, and the value *value*. For example:

```
GisSetInt "SIS_OT_CURITEM, 0, "RefNo&",23
```

Available MM, ME, MD, OV, OM, OD, ASC

❍ SetListFlt

Set the value of a floating point property on all the items in a named list.

Syntax

```
GisSetListFlt ( list, propertyName, value )
```

Arguments

list STRING
the named list containing the items whose property is to be set

propertyName STRING
the name of the property: all floating point properties end in #

value DOUBLE
the new floating point value of the property

Example

```
GisSetListFlt "Points", "Val#", 100.98
```

Notes If no property exists of the name *propertyName*, and *propertyName* does not start with the _ (underscore) character, a user-defined attribute will be created with the name *propertyName#*, and the value *value*.

Available MM, ME, MD, OM, OD, ASC

❍ SetListInt

Set the value of an integer property on all the items in a named list.

Syntax

```
GisSetListInt ( list, propertyName, value )
```

Arguments

list STRING
the named list containing the items whose property is to be set

propertyName STRING
the name of the property: all integer properties end in &

value LONG INTEGER
the new integer value of the property

Example

```
GisSetListInt "Points", "TYP&", 10
```

Notes If no property exists of the name *propertyName*, and *propertyName* does not start with the _ (underscore) character, a user-defined attribute will be created with the name *propertyName&*, and the value *value*.

Available MM, ME, MD, OM, OD, ASC

❍ SetListStr

Set the value of a string property on all the items in a named list.

Syntax

```
GisSetListStr ( list, propertyName, value )
```

Arguments

list STRING
the named list containing the items whose property is to be set

propertyName STRING
the name of the property: all string properties end in $

value STRING
the new string value of the property

Example `GisSetListStr "Points", "Category$", "RTA"`

Notes If no property exists of the name *propertyName*, and *propertyName* does not start with the _ (underscore) character, a user-defined attribute will be created with the name *propertyName*$, and the value *value*.

Available MM, ME, MD, OM, OD, ASC

❍ SetOverlayFilter

Apply a copy of a named filter to an overlay in the current SWD.

Syntax `GisSetOverlayFilter ( pos, filter )`

Arguments *pos* SHORT INTEGER
the position of the overlay in the overlays list whose drawing filter is to be set

filter STRING
the named filter to copy. This can be any previously created named filter, or a filter loaded from a named object library or `""` to unset the overlay drawing filter.

Example `GisSetOverlayFilter 1 "Polygons>50"`
sets the include filter on the overlay 1 to Polygons>50

Notes The overlays start at position 0.
Any named filter given will be copied, so any subsequent changes to the named filter will not be reflected in the overlay drawing filter.

Available MM, ME, MD, OV, OM, OD, ASC

❍ SetOverlayLocus

Apply a copy of a named locus to an overlay in the current SWD.

Syntax `GisSetOverlayLocus ( pos, locus )`

Arguments *pos* SHORT INTEGER
the position of the overlay in the overlays list whose drawing locus is to be set

locus STRING
the named locus to copy. This can be any named locus previously created, loaded from a named object library, or `""` to unset the overlay drawing locus.

Example `GisSetOverlayLocus 0, "Schema10"`

Notes Any named locus given will be copied, so any subsequent changes to the named locus will not be reflected in the overlay drawing locus.

Available MM, ME, MD, OV, OM, OD, ASC

❍ SetOverlaySchema

Apply a copy of a named schema to an overlay in the current SWD.

Syntax `GisSetOverlaySchema ( pos, schema )`

Arguments `pos` SHORT INTEGER
the position of the overlay in the overlays list whose schema is to be set

`schema` STRING
the named schema to copy. This can be any named schema previously created, any named scheme loaded from a named object library, or `""` to unset the overlay schema.

Notes Any named schema given will be copied, therefore any subsequent changes to the named schema will not be reflected in the overlay schema.

Available MM, ME, MD, OM, OD, ASC

❍ SetPhotoWorldCentre

Set the centre of the view within the current open photo item.

Syntax `GisSetPhotoWorldCentre ( x, y, z )`

Arguments `x, y, z` DOUBLE
the vector displacements of the current photo view centre to the new photo view centre, in photo world co-ordinates

Example To calculate the vector displacement, do the following:

1. From the photo object properties, get the Origin X and Origin Y properties (`_ox#` and `_oy#`) in paper co-ordinates:
   `OpenSel 0PhotoOriginX = GetFlt(SIS_OT_CURITEM, 0, "_ox#")PhotoOriginY = GetFlt(SIS_OT_CURITEM, 0, "_oy#")`
2. Convert the paper co-ordinates into world co-ordinates:
   `PhotoWorldOrigin = GetPhotoWorldPos(PhotoOriginX, PhotoOriginY)`
3. Use the `SplitPos` function to get the world co-ordinates:
   `SplitPos x, y, z, PhotoWorldOrigin`
4. Move the contents of the photo 50 metres north-east:
   `SetPhotoWorldCentre x + 50, y + 50, z`

Notes This will re-centre the map in the photo item to the new real world co-ordinates you have chosen. In the above example, the map is moved by 50 metres in both the x and y directions (north east).

Available MM, ME, MD, OM, OD, ASC

❍ SetRubberTransform

Set the current rubber sheet transformation from the currently open rubber sheet item. This method must be used before transforming raster or vector data.

Syntax `GisSetRubberTransformation ( method )`

Arguments `method` SHORT INTEGER
the displacement method to use (see Notes)

Notes Three displacement methods are available:
`SIS_RUBBER_BEST_FIT`
Only three of the displacement items are used, so if you have created and selected more than three you will not be able to predict the results.

`SIS_RUBBER_LINEAR_PATCH`

All the start points of the displacements are triangulated. Only points which line inside one of the triangles or on its border, are transformed – other points are left unchanged.

This method is most suited to fitting digitised vector data to known positions.

`SIS_RUBBER_INVERSE_SQUARE`

All displacement items are used to transform every point, but closer displacements have more effect on a particular point than the more distance ones. If a point is exactly on the start of a displacement, it will be transformed to the end displacement.

Points a long way away from all displacements will be moved to the average displacement. Therefore, if the average displacement is zero, the transformation will only have a local effect.

This method is most suited to fitting a scanned bitmap onto existing data accurately.

Available ME, MD, OD, ASC

❍ SetStr

Set the value of a string property.

Syntax `GisSetStr ( objectType, nObject, propertyName, value )`

Arguments

objectType SHORT INTEGER

➲page 221, **Object types**

nObject LONG INTEGER

the index of the object type

propertyName STRING

the name of the property: all string properties end in $

value STRING

the new string value of the property

Example `GisSetStr SIS_OT_CURITEM, 0, "TYP$", "Conservation Area"`

Notes When using `SIS_OT_CURITEM` for *objectType*, if no property exists of the name *propertyName*, and *propertyName* does not start with the _ (underscore) character, a user-defined attribute will be created, with the name *propertyName*$, and the value *value*.

Available MM, ME, MD, OV, OM, OD, ASC

❍ SetStrW

Set the Unicode value of a string property.

Syntax `GisSetStrW ( objectType, nObject, propertyName, value )`

Arguments

objectType SHORT INTEGER

➲page 221, **Object types**

nObject LONG INTEGER

the index of the object type

propertyName STRING
the name of the property: all string properties end in $. The special properties `_properties$` and `_properties_edit$` are used to get lists of available properties for querying and editing respectively.

value STRING
the new Unicode value of the given string property

Example `GisSetStrW SIS_OT_CURITEM, 0, "_brush$", "black"`

Notes When using `SIS_OT_CURITEM` for *objectType*, if no property exists of the name *propertyName*, and *propertyName* does not start with the _ (underscore) character, a user-defined attribute will be created, with the name *propertyName*$, and the value *value*.

Available OV, OM, OD, ASC

❍ SetUnits

Set the preferred units used in the user interface.

Syntax `GisSetUnits ( units, places )`

Arguments *units* STRING
a string, describing the units to use. Currently valid strings are: m, mm, cm, km, foot, inch, imperial, yd, fathom, mile, and nautical mile.

places SHORT INTEGER
the number of decimal places to display

Example `GisSetUnits "m", 2`
sets the units to metres to two decimal places

Notes This method is included for backwards compatibility. Use `SetCoordUnits` instead.

Available MM, ME, MD, OM, OD

❍ SetupLink

Start the conversation with Cadcorp SIS. This should be called only once per Visual Basic program, typically in the `Form_Load()` of the main form.

Syntax `rv = GisSetupLink ( hWndForm )`

Arguments *hWndForm* SHORT INTEGER
the Microsoft Visual Basic form containing the ListCaps and command buttons

Returns SHORT INTEGER

non-zero	link succeeded
0	link failed

Example

```
Private Sub Form_Load()
  If GisSetupLink(hwnd) = 0 Then
    MsgBox "Error connecting to Cadcorp SIS application"
    End
  End If
End Sub
```

Notes In Microsoft Visual Basic 4.0 32-bit and Microsoft Visual Basic 5.0 and 6.0, the `hWndForm` argument is a long integer.

Available MM, ME, MD

❍ SetViewExtent

Set the visible extents of the current window.

Syntax `GisSetViewExtent ( x1, y1, z1, x2, y2, z2 )`

Arguments *x1, y1, z1, x2, y2, z2* DOUBLE
the new rectangular extents of the view

Available MM, ME, MD, OV, OM, OD, ASC

❍ SetViewPrj

Set the projection of the current window's view.

Syntax `GisSetViewPrj ( projection )`

Arguments *projection* STRING
the named projection to use

Notes If the current axes are incompatible with the new view projection, the current axes will also be changed. If the current window is not a map window, or there is no current window, the default view projection is set.

Available MM, ME, MD, OV, OM, OD, ASC

❍ ShowMenu

Set the visibility of the main menu. By default the main menu is visible.

Syntax `GisShowMenu ( bShow )`

Arguments *bShow* LONG INTEGER

True show the main menu

False hide the main menu

Example `GisShowMenu False`
hides the main menu

Available MM, ME, MD

❍ SimplifyGeom

Simplify the geometry of the current open item. The meaning of Simple geometry is that defined by the OpenGIS consortium.

Syntax `GisSimplifyGeom ( )`

Available ME, MD, OD, ASC

❍ Snap2D

Simulate a 2D snap, making the snapped item current, and returning the snapped position.

Syntax `Snap2D (x, y, r, bEditOnly, codes, filter, locus)`

Arguments

x, y DOUBLE

the position from which to snap. Item geometries with any Z values will be considered.

r DOUBLE

the approximate radius to search within

bEditOnly LONG INTEGER

should Cadcorp SIS consider only editable items?

code STRING

the types of geometry you wish to snap to. You can supply a single letter, or a list of letters. Cadcorp SIS finds the closest matching geometry. Some snapcodes take priority over others. For example, if you specify `LV` and a vertex is only slightly further away than a line, Cadcorp SIS will snap to the vertex.

A find smallest area containing position

B snap to box-text justification point

C snap to centre of curvature of closest line

E snap to nearest end of closest line

H snap to hook position, or origin, of item containing closest line

L snap to closest line

M snap to middle of closest line segment

P snap to closest point

R snap to corner of closest raster pixel that constrasts with its neighbour

T snap to closest text

V snap to closest line vertex

X snap to closest linear intersection

These snapcodes apply to geometry, not item classes. For example, the L snapcode finds the closest linear geometry, which may be part of a line item, or the boundary of an area item. Similarly, the A snapcode will find any area-filling item (eg polygon).

filter STRING

optionally specifies a named filter which items must pass to be considered for snapping

locus STRING

optionally specifies a named locus which items must pass to be considered for snapping

Returns STRING

a comma-delimited string containing the x, y, and z co-ordinates of the snapped position within the current axes. Use `SplitPos` to get the x, y, and z values themselves.

Available MM, ME, MD, OM, OD, ASC

❍ SnipGeometry

Snip away portions of the items inside or outside the current item.

Syntax `GisSnipGeometry ( list, bInside )`

Arguments *list* STRING
the named list containing the items to be snipped

bInside SHORT INTEGER

True the portions of the items inside the current item will be deleted

False the portions of the items outside the current item will be deleted

Note This method does not operate on topological items.

Example `GisSnipGeometry ("Roads", False)`
all items in the Roads named list will be snipped to the boundary of the current item, and all parts outside of the item will be deleted

Available ME, MD, OD, ASC

❍ SplitExtent

Split a comma-delimited extents string into numbers.

Syntax `rv = GisSplitExtent ( x1, y1, z1, x2, y2, z2, ext )`

Arguments *x1, y1, z1, x2, y2, z2* DOUBLE
the extents read from the extents string

ext STRING
the comma-delimited extents string to split, normally returned as a result of `GisGetCoordExtent`

Returns SHORT INTEGER

True extents successfully split

False failed to split extents

Available MM, ME, MD, OV, OM, OD, ASC

❍ SplitPos

Split a comma-delimited position string into numbers.

Syntax `rv = GisSplitPos ( x, y, z, pos )`

Arguments *x, y, z* DOUBLE
the position read from the position string

pos STRING
the comma-delimited position string to split

Returns SHORT INTEGER

True	extents successfully split
False	failed to split extents

Available MM, ME, MD, OV, OM, OD, ASC

❍ StoreAsArea

Store the previous `MoveTo`/`LineTo` operations as an area item.

Syntax `GisStoreAsArea ( )`

Notes If the current open item is a group, the graphics will be locked to the cursor for the user to place with two screen snaps (position and alignment).

Available MM, ME, MD, OM, OD, ASC

❍ StoreAsLine

Store the previous `MoveTo`/`LineTo` operations as a line item. This method is an alternative to using `MoveTo` to store the previous `MoveTo`/`LineTo` operations.

Syntax `GisStoreAsLine ( )`

Notes If the current open item is a group, the graphics will be locked to the cursor for the user to place with two screen snaps (position and argument).

Available MM, ME, MD, OM, OD, ASC

❍ StoreFeatureTable

Store the currently open feature table, replacing any existing feature table with the same name.

Syntax `GisStoreFeatureTable ( ftable )`

Arguments *`ftable`* STRING
the name of the feature table to create or replace

Notes Use `LoadFeatureTable` to load a feature table for editing.

Available ME, MD, OM, ASC

❍ StoreSchema

Store the currently open schema in the current NOL or workspace, replacing any existing schema with the same name.

Syntax `GisStoreSchema ( schema )`

Arguments *`schema`* STRING
the name of the schema to create or replace

Notes Use `LoadSchema` to load a schema for editing.

Available MM, ME, MD, OM, OD, ASC

❍ StoreTheme

Store the currently open theme, replacing any existing theme with the same name. Use `LoadTheme` to load a theme for editing.

Syntax `GisStoreTheme ( theme )`

Arguments *theme* STRING
the name of the theme to create or replace

Notes In the Cadcorp SIS Control (Viewer level), themes can be stored only to the (temporary) NOL.

Available MM, ME, MD, OV, OM, OD, ASC

❍ StrFromMetre

Format a metre dimension as a string, in a chosen format.

Syntax `rv = GisStrFromMetre ( metre, ndim, showunits )`

Arguments *metre* DOUBLE
the metre dimension to be formatted

ndim SHORT INTEGER

`SIS_LENGTHDIM`	format the dimension as a length
`SIS_AREADIM`	format the dimension as an area
`SIS_VOLUMEDIM`	format the dimension as a volume

showunits SHORT INTEGER

True	show the units in the formatted string
False	do not show the units in the formatted string

Returns STRING
a string formatted in the current units

Example `rv = GisStrFromMetre (1000, SIS_LENGTHDIM,TRUE)`

Available MM, ME, MD, OM, OD

❍ SwdClose

Close all of the windows of the current SWD, using the chosen savecode.

Syntax `GisSwdClose ( savecode )`

Arguments *savecode* SHORT INTEGER

`SIS_NOSAVE`	do not save the SWD
`SIS_SAVE`	save the SWD if it has been modified
`SIS_PROMPTSAVE`	prompt the user to save if the SWD has been modified

Available MM, ME, MD

❍ SwdNew

Create a new, empty SWD.

Syntax `GisSwdNew ( )`

Available MM, ME, MD

❍ SwdNewFromSwt

Create a new SWD from a saved window template.

Syntax `GisSwdNewFromSwt ( template )`

Arguments *template* STRING
the filepath and filename of the saved window template file to use for the new SWD

Available MM, ME, MD

❍ SwdNewWindow3D

Create a new 3D window based on the current SWD.

Syntax `GisSwdNewWindow3D ( xEye, yEye, zEye, xLook, yLook, zLook )`

Arguments *xEye, yEye, zEye* DOUBLE
the eye position, in metres

xLook, yLook, zLook DOUBLE
the look position, in metres

Example `GisSwdNewWindow3D 300, 300, 300, 50, 50, 50`
creates a new 3d window with the eye point at 300, 300, 300 and a look point of 50, 50, 50

Available MD

❍ SwdNewWindow

Create a new window based on the current SWD, with the given view.

Syntax `GisSwdNewWindow ( x1, y1, z1, x2, y2, z2 )`

Arguments *x1, y1, z1, x2, y2, z2* DOUBLE
the rectangular extents of the view in the new window, in metres

Example `GisSwdNewWindow 0, 0, 0, 200, 200, 0`
creates a new SWD based on the extents 0, 0, 0, 200, 200, 0 based on the current SWD

Available MM, ME, MD

❍ SwdNewWindowTable

Create a new table window based on the current SWD.

Syntax `GisSwdNewWindowTable ( )`

Available MM, ME, MD

❍ SwdOpen

Open an existing SWD file, optionally making it read-only.

Syntax `GisOpenSwd ( filename, readonly )`

Arguments *filename* STRING
the full pathname of the SWD file to open

readonly SHORT INTEGER

True make the SWD read-only

False make the SWD editable

Example `GisSwdOpen "c:\projects\keymap.swd", True`
opens the SWD keymap.swd as read-only

Available MM, ME, MD

❍ SwdSave

Save the current SWD.

Syntax `GisSwdSave ( )`

Available MM, ME, MD

❍ SwdSaveAs

Save the current SWD with a different name. The original file is not renamed but kept on disk.

Syntax `GisSaveAs ( filename )`

Arguments *filename* STRING
the new name for the SWD

Example `GisSwdSaveAs "c:\projects\medway\keyplan99.swd"`
saves the current SWD with the name c:\projects\medway\keyplan99.swd

Available MM, ME, MD

❍ SwdSaveAsSwt

Save the current SWD as a saved window template file.

Syntax `GisSwdSaveAsSwt ( filename )`

Arguments *filename* STRING
the new name for the saved window template file

Available MM, ME, MD

❍ SwitchCommand

Queue a command which requires mouse input (a callback command) for the current window and return immediately. The command will be started in the current window when `GisRelease` has been called.

Syntax `GisSwitchCommand ( comname )`

Arguments `comname` STRING
the ACom command to queue

Example `GisSwitchCommand "AComLineEx"`
queues the AComLineEx command (**Construct>2D Geometry>Line**)

Available MM, ME, MD

❍ SwtClose

Close all the windows of the current saved window template.

Syntax `GisSwtClose ( savecode )`

Arguments `savecode` SHORT INTEGER

`SIS_NOSAVE`	do not save the SWT
`SIS_SAVE`	save the SWT if it has been modified
`SIS_PROMPTSAVE`	prompt the user to save if the SWT has been modified

Available MM, ME, MD

❍ SwtNew

Create a new, empty saved window template.

Syntax `GisSwtNew ( )`

Available MM, ME, MD

❍ SwtNewFromSwt

Create a new, empty SWD from a saved window template.

Syntax `GisSwtNewFromSwt ( filename )`

Arguments `filename` STRING
the saved window template file to use for the new SWD

Available MM, ME, MD

❍ SwtOpen

Open an existing saved window template file.

Syntax `GisSwtOpen ( filename )`

Arguments `filename` STRING
the saved window template file to open

Available MM, ME, MD

❍ SwtSave

Save the current saved window template.

Syntax `GisSwtSave ( )`

Available MM, ME, MD

❍ SwtSaveAs

Rename and save the current saved window template file.

Syntax `GisSwtSaveAs ( filename )`

Arguments *filename* STRING
the new name for the saved window template file

Available MM, ME, MD

❍ SwtSaveAsSwt

Save the current saved window definition as a saved window template file.

Syntax `GisSwtSaveAsSwt ( filename )`

Arguments *filename* STRING
the new name for the saved window template file

Available MM, ME, MD

❍ TableNewWindow

Create or activate a view onto a named table, displaying columns within the table. The view is displayed in a table window.

Syntax `GisTableNewWindow ( table )`

Arguments *table* STRING
the named table to view

Example `GisTableNewWindow "HouseData"`
displays a table window containing the information from the columns of the table HouseData

Available MM, ME, MD, OM, OD, ASC

❍ Takeover

Take over control from the application.

Syntax `rv = GisTakeOver ( )`

Returns SHORT INTEGER
True, if Takeover succeeded, or False if Takeover failed

Notes This method should be called before calling other GisLink methods when not responding to a previously added GisLink command, eg following a button press on a modeless dialog or in response to an outside event like a digitiser snap. This method will fail if the Cadcorp SIS application is in a dialog, or if another Microsoft Visual Basic customisation already has control.

Available MM, ME, MD

❍ TickCommand

Set or clear a tick on an application-defined command previously added using `AddCommand`.

Syntax

```
ControlName.TickCommand ( comname, tick )
GisTickCommand ( comname, tick )
```

Arguments

`comname` STRING

the command to tick. This is the value of menu in the method `AddCommand`.

`tick` SHORT INTEGER

True	switch the tick on
False	switch the tick off

Example

```
Sis.TickCommand "Query#Property Details", TRUE
GisTickCommand "Query|Property Details", TRUE
```

both examples set the tick state of the **Query>Property Details** command to true

Available MM, ME, MD, OM, OD

❍ TopoClean

Clean up topological Link items.

Syntax

```
GisTopoClean ( list, tolerance, options )
```

Arguments

`list` STRING

the named list containing the link items to be cleaned

`tolerance` DOUBLE

the tolerance to use. Link items whose length is less than this value will be removed. If `SIS_CLEAN_TOPO_FIX_UNDER_OVER` is specified, link items with a dangling end will be joined to another link item within this distance.

`options` SHORT INTEGER

`SIS_CLEAN_TOPO_NONE`	delete link items shorter than this value
`SIS_CLEAN_TOPO_REMOVE_DANGLING`	delete link items which are not joined to other link items, regardless of their length
`SIS_CLEAN_TOPO_FIX_UNDER_OVER`	close small gaps, and delete small dangling link items
`SIS_CLEAN_TOPO_REMOVE_SEEDS`	delete seed items whose link items are all deleted by this operation

You can add `SIS_CLEAN_TOPO_REMOVE_DANGLING`, `SIS_CLEAN_TOPO_FIX_UNDER_OVER`, and `SIS_CLEAN_TOPO_REMOVE_SEEDS` together, to perform several types of cleaning at once.

On completion, the named list will contain the topological items which have been changed.

Example

```
GisTopoClean "CleanList", 1, SIS_CLEAN_TOPO_NONE
```

Available ME, MD, OD, ASC

❍ TopoCombineNamedSeeds

Create a named seed object by using a Boolean operation on existing named seed objects.

Syntax `GisTopoCombineNamedSeeds ( seedOutput, seed1, seed2, boolop )`

Arguments *`seedOutput`* STRING
the new named seed object, which will be a polygon item

`seed1, seed2` STRING
the named seed objects to combine, which must be polygon items in the same dataset

`boolop` SHORT INTEGER
➲page221, **Boolean tests**

Example `GisTopoCombineNamedSeeds "County", "District1", "District2", SIS_BOOLEAN_OR`
displays the named seed County. To display the newly created named seed County as a polygon, use the `TopoCreatePolygon` method.

Available ME, MD, OD, ASC

❍ TopoConvertToArea

Convert the current open polygon item into an area item.

Syntax `GisTopoConvertToArea ( bDeleteUnusedLinks )`

Arguments *`bDeleteUnusedLinks`* SHORT INTEGER

True delete any redundant link items after converting to an area item

False keep all link items

Available ME, MD, OD, ASC

❍ TopoConvertToChain

Convert the current open line item into a topological chain item. This method will insert node items at the start and end of the chain item, and at any intersections with existing link items.

Syntax `GisTopoConvertToChain ( category )`

Arguments *`category`* STRING
the `_cluster$` property of the resulting chain item will be assigned the value of *`category`*

Example `GisTopoConvertToChain "BusRoute"`

Available ME, MD, OD, ASC

❍ TopoConvertToLine

Convert the current open chain item into a line item.

Syntax `GisTopoConvertToLine ( bDeleteUnusedLinks )`

Arguments `bDeleteUnusedLinks` SHORT INTEGER

True delete unused link items after converting to a line item

False keep all link items

Available ME, MD, OD, ASC

❍ TopoConvertToPolygon

Convert the current open area item into a topological polygon item.

Syntax `GisTopoConvertToPolygon ( category )`

Arguments `category` STRING

the `_cluster$` property of the resulting polygon item will be assigned the value of `category`

Example `GisTopoConvertToPolygon "District"`

Available ME, MD, OD, ASC

❍ TopoCreateArea

Create an area item from the current open polygon item, which can be read-only.

Syntax `GisTopoCreateArea ( )`

Available ME, MD, OD, ASC

❍ TopoCreateBoolean

Create a named seed object using the selected Boolean operation on existing polygon items.

Syntax `GisTopoCreateBoolean ( seed, list, boolop )`

Arguments `seed` STRING

the new named seed object, which will be a polygon item

`list` STRING

a named list containing the polygon items. The items to be combined must all be in the same dataset.

`boolop` SHORT INTEGER

➲page221, **Boolean tests**

Example `GisTopoCreateBoolean "County", "PolysFound", SIS_BOOLEAN_DIFF`

Available ME, MD, OD, ASC

❍ TopoCreateChain

Create a chain item from a named seed object.

Syntax `GisTopoCreateChain ( seed, category )`

Arguments `seed` STRING

the named seed object from which the chain item will be created

category STRING
the `_cluster$` property of the resulting chain item will be assigned the value of *category*

Example `GisTopoConvertToChain "BusRoute"`

Available ME, MD, OD, ASC

❍ TopoCreateEmptyNamedSeed

Create a new, empty transient named seed object.

Syntax `GisTopoCreateEmptyNamedSeed ( seed, nDataset )`

Arguments *seed* STRING
the name of the new named seed object

nDataset LONG INTEGER
the serial number of the topological dataset. The serial number can be obtained from the `_nDataset&` property of an overlay, or from the `GetDataset`, `FindExternalDataset`, or `TopoGetNamedSeedDataset` methods.

Example
```
GisTopoCreateEmptyNamedSeed "Park", GisGetInt (SIS_OT_OVERLAY, 0, _
  "_nDataset&")
```

Available ME, MD, OD, ASC

❍ TopoCreateLine

Create a line item from the current open chain item, which may be read-only.

Syntax `GisTopoCreateLine ( )`

Available ME, MD, OD, ASC

❍ TopoCreateLink

Create a topological link item, copying the geometry from the current line item.

Syntax `GisTopoCreateLink ( )`

Notes This method will insert node items only at the start and end of the current line item. Therefore any intersections with existing link items will not be detected.

Available ME, MD, OD, ASC

❍ TopoCreateNamedSeed

Create a transient named seed object from the current seed item.

Syntax `GisTopoCreateNamedSeed ( seed )`

Arguments *seed* STRING
the name of the new named seed object

Example `GisTopoCreateNamedSeed "Conservation Area"`

Available ME, MD, OD, ASC

❍ TopoCreateNode

Create a node item, merging it in to any existing topology.

Syntax `GisTopoCreateNode ( x, y, z )`

Arguments *x, y, z* DOUBLE
the position at which to create the new node item

Available ME, MD, OD, ASC

❍ TopoCreatePolygon

Create a polygon item from a named seed object.

Syntax `GisTopoCreatePolygon ( seed, category )`

Arguments *seed* STRING
the named seed object from which the polygon item will be created

category STRING
the `_cluster$` property of the resulting polygon item will be assigned the value of *category*

Example `GisTopoCreatePolygon "PolygonSeed", "District"`

Available ME, MD, OD, ASC

❍ TopoDeleteLink

Delete the current open link item.

Syntax `GisTopoDeleteLink ( )`

Available ME, MD, OD, ASC

❍ TopoDeleteNamedSeed

Delete a transient named seed object.

Syntax `GisTopoDeleteNamedSeed ( seed )`

Arguments *seed* STRING
the named seed object to delete

Example `GisTopoDeleteNamedSeed "Stevenage"`

Available ME, MD, OD, ASC

❍ TopoDeleteNode

Delete or simplify the current open node item.

Syntax `GisTopoDeleteNode ( )`

Notes Simplification means that two connected link items can be joined together, and merged into a single link item which no longer uses the node. If all connected link items can be joined in this way, the node is deleted.

Available ME, MD, OD, ASC

❍ TopoDeleteSeed

Delete the current open topological chain or polygon item.

Syntax `GisTopoDeleteSeed ( bDeleteUnusedLinks )`

Arguments *bDeleteUnusedLinks* SHORT INTEGER

True delete any unused link items after the chain or polygon item has been deleted

False keep all link items

Available ME, MD, OD, ASC

❍ TopoEdgeFill

Create a named seed object by following along the current open link item to make a closed loop.

Syntax `GisTopoEdgeFill ( seed, bForwards, filter )`

seed STRING

the new named seed object, which will be a polygon item

bForwards SHORT INTEGER

True follow the current open link item from the start node to the end node

False follow the current open link item from the end node to the start node

filter STRING

optionally specifies a named filter which all link items must pass to be considered as part of the polygon

Returns SHORT INTEGER

True on success, False on failure

Example `GisTopoEdgeFill ("Stevenage", True, "CountyBoundary")`

Notes The link following algorithm always turns left at a node. Therefore this method will always return an anti-clockwise polygon. The exception to this is if the current link item is an edge link. In this case, one value of *bForwards* will return a clockwise perimeter polygon and the other will return an anti-clockwise polygon.

Available ME, MD, OD, ASC

❍ TopoFindRoute

Create a named seed object by route-finding between two node items within a dataset.

Syntax `rv = GisTopoFindRoute ( seed, idNode1, idNode2, nDataset, formula, filter, locusNoGo )`

Arguments *seed* STRING

the name of the new named seed object

idNode1 LONG INTEGER

the start node of the route

idNode2 LONG INTEGER

the end node of the route

nDataset LONG INTEGER
the serial number of the topological dataset. The serial number can be obtained from the `_nDataset&` property of an overlay, or from the `GetDataset`, `FindExternalDataset`, or `TopoGetNamedSeedDataset` methods.

formula STRING
the formula, or simple property, to use in the route finding calculation as the 'cost' of a link item. For example, using the simple property `_length#` property will find the shortest route, and using the formula `_length#/Speed#`, provided each link has a user-defined `Speed#` property, will find the quickest route. Any formula can be used, although if a string formula is used it must be a string representation of a numeric value.

filter STRING
optionally specify a named filter which all link items must pass to be considered as part of the route

locusNoGo STRING
optionally specify a named locus through which the route cannot pass. The named locus used will normally have its testing mode set to exclude any link items which cross it, using a call similar to the following:

```
GisCreateLocusFromItem "NoGo", SIS_GT_INTERSECT, SIS_GM_GEOMETRY
```

Returns SHORT INTEGER
True on success, False on failure

Example

```
GisTopoFindRoute "Journey1", startNode, finishNode, datasetID, "_length#", _
  "RoadFilter", "NoGo"
```

Available ME, MD, OD, ASC

❍ TopoFloodFill

Create a named seed object by flood-filling link items within a dataset.

Syntax

```
rv = GisTopoFloodFill ( seed, x, y, z, nDataset, filter )
```

Arguments

seed STRING
the name of the new named seed object

x, y, z DOUBLE
the position from which to flood-fill

nDataset LONG INTEGER
the serial number of the topological dataset. The serial number can be obtained from the `_nDataset&` property of an overlay, or from the `GetDataset`, `FindExternalDataset`, or `TopoGetNamedSeedDataset` methods.

filter STRING
optionally specifies a named filter which all link items must pass to be considered as part of the flood-fill

Returns SHORT INTEGER
True on success, False on failure

Example

```
GisTopoFloodFill "Garden", Xpos, Ypos, Zpos, datasetID, "FenceFilter"
```

Available ME, MD, OD, ASC

❍ TopoGetLinkNode

Get the ID of a node item from the current open link item.

Syntax `rv = GisTopoGetLinkNode ( bStart )`

Arguments *bStart* LONG INTEGER

True return the node at the start of the current link item

False return the node at the end of the current link item

Returns LONG INTEGER

the ID of the node at the start or end of the current link item

Example `NodeNum = GisTopoGetLinkNode (True)`

Available ME, MD, OD, ASC

❍ TopoGetLinkNumSeed

Get the number of seed items attached to the current open link item.

Syntax `rv = GisTopoGetNumSeed ( )`

Returns SHORT INTEGER

the number of seed items attached to the current link item

Available ME, MD, OD, ASC

❍ TopoGetLinkSeed

Get the signed ID of a polygon or chain item from the current open link item.

Syntax `rv = GisTopoGetLinkSeed ( n )`

Arguments *n* SHORT INTEGER

the index of the polygon or chain item in the list of polygon and chain items attached to the current link item, starting at 0. Use `TopoGetLinkNumSeed` to find out the number of seed items attached to a link item.

Returns LONG INTEGER

the ID of a polygon or chain. The ID number will be negative if the link item is reversed in the polygon or chain item, ie the polygon or chain item runs from the end node to the start node.

Available ME, MD, OD, ASC

❍ TopoGetNamedSeedDataset

Get dataset with which a named seed item is compatible.

Syntax `rv = GisTopoGetNamedSeedDataset ( seed )`

Arguments *seed* STRING

the named seed object whose dataset is required

Returns LONG INTEGER

the serial number of the dataset with which the named seed object is compatible. The serial number cannot be relied upon to be identical in each session, or if the dataset is

removed and re-added. The serial number returned should therefore not be stored for long-term use.

Available ME, MD, OD, ASC

❍ TopoGetNamedSeedLoopLink

Get the ID of a link item from a named seed object.

Syntax `rv = GisTopoGetNamedSeedLoopLink ( seed, nLoop, n )`

Arguments *seed* STRING

the named seed object to be queried

nLoop SHORT INTEGER

the index in the list of loops in the named seed object of the loop to be queried, starting at 0. Use `TopoGetNamedSeedNumLoop` to find out the number of loops in a named seed object.

n SHORT INTEGER

the index in the loop of the link item to be queried, starting at 0. Use `TopoGetNamedSeedLoopSize` to find out the number of link items referred to by a loop in a named seed object.

Returns LONG INTEGER

the ID of a link item from a named seed object

Example `rv = GisTopoGetNamedSeedLoopLink ("Stevenage", 1, 2)`

Available ME, MD, OD, ASC

❍ TopoGetNamedSeedLoopSize

Get the number of link items referred to by a loop in a named seed object.

Syntax `rv = GisTopoGetNamedSeedLoopSize ( seed, nLoop )`

Arguments *seed* STRING

the named seed object to be queried

nLoop SHORT INTEGER

the index in the list of loops in the named seed object of the loop to be queried, starting at 0. Use `TopoGetNamedSeedNumLoop` to find out the number of loops in a named seed object.

Returns SHORT INTEGER

the number of link items in the named seed loop

Example `NLinks = GisTopoGetNamedSeedLoopSize ("County", 1)`

Available ME, MD, OD, ASC

❍ TopoGetNamedSeedNumLoop

Get the number of loops in a named seed object.

Syntax `rv = GisTopoGetNamedSeedNumLoop ( seed )`

Arguments *seed* STRING

the named seed object to be queried

Returns SHORT INTEGER

the number of loops in the named seed object

Available ME, MD, OD, ASC

❍ TopoGetNodeLink

Get the signed ID of a link item from the current open node item.

Syntax `rv = GisTopoGetNodeLink ( n )`

Arguments *n* SHORT INTEGER

the index in the list of link items attached to the current node item of the link item to be queried, starting at 0

Returns LONG INTEGER

the ID of a link item. The ID number will be negative if the link item ends at the current node item.

Available ME, MD, OD, ASC

❍ TopoGetNodeNumLink

Get the number of link items attached to the current open node item.

Syntax `rv = GisTopoGetNodeNumLink ( )`

Returns SHORT INTEGER

the number of link items attached to the current node item

Available ME, MD, OD, ASC

❍ TopoGrowNamedSeed

Add a link ID into a named seed object.

Syntax `GisTopoGrowNamedSeed ( seed, bStart, idLink )`

Arguments *seed* STRING

the named seed object to which the link item will be added

bStart SHORT INTEGER

True insert the link item at the start of the named seed object

False append the link item to the end of the named seed object

idLink LONG INTEGER

the ID of the link item to be added

Example `GisTopoGrowNamedSeed "County", True, 23`

Available ME, MD, OD, ASC

❍ TopoIsChain

Test if a named seed object is a topological chain.

Syntax `rv = GisTopoIsChain ( seed )`

Arguments `seed` STRING
the named seed object to test

Returns SHORT INTEGER

True the named seed object is a topological chain

False the named seed object is not a topological chain

Available ME, MD, OD, ASC

❍ TopoIsPolygon

Test if a named seed object is a topological polygon.

Syntax `rv = GisTopoIsPolygon ( seed )`

Arguments `seed` STRING
the named seed object to test

Returns SHORT INTEGER

True the named seed object is a topological polygon

False the named seed object is not a topological polygon

Available ME, MD, OD, ASC

❍ TopoMoveNode

Move the current open node item, dragging any connected link items.

Syntax `GisTopoMoveNode ( x, y, z )`

Arguments `x, y, z` DOUBLE
the new position of the node item

Notes The position specified is an absolute position.

Available ME, MD, OD, ASC

❍ TopoReverseSeed

Reverse the direction of the current open chain or polygon item.

Syntax `GisTopoReverseSeed ( )`

Available ME, MD, OD, ASC

❍ TopoShrinkNamedSeed

Remove a link item from a named seed object.

Syntax `rv = GisTopoShrinkNameSeed ( seed, bStart )`

Arguments `seed` STRING
the named seed object from which to remove the link item

bStart SHORT INTEGER

True remove the first link item in the named seed object

False remove the last link item in the named seed object

Returns LONG INTEGER

the ID of the link item removed

Example `LinkId = GisTopoShrinkNameSeed ("County", True)`

Available ME, MD, OD, ASC

❍ TraceGeom

Create a line item by tracing geometry from the current open item.

Syntax

```
GisTraceGeom ( nGeom, arclenStart, arclenEnd, offset,
bAllowSelfIntersections )
```

Arguments

nGeom LONG INTEGER

the index of the geometry component, starting at 0. Use `GetNumGeom` to get the number of geometry components in an item.

arclenStart DOUBLE

the length along the geometry component from which to start the trace

arclenEnd DOUBLE

the length along the geometry component at which to end the trace

offset DOUBLE

the offset, in current units, from the geometry component at which to create the line item

bAllowSelfIntersections SHORT INTEGER

True allow the traced, offset line item to contain self-intersections

False remove any self-intersections from the traced, offset line item

Example `GisTraceGeom 0, 20, 100, False`

Available ME, MD, OD, ASC

❍ UpdateItem

Update the current open item, leaving it current.

Syntax `GisUpdateItem ( )`

Notes If the current open item is a group, the graphics will be locked to the cursor for the user to place with two screen snaps (position and alignment). In this case the group will not be the current open item after this method.

`UpdateItem` is required after `SetInt`, `SetFlt`, or `SetStr` are used to modify the properties of the current item.

Available MM, ME, MD, OM, OD, ASC

❍ UpdateWorkspaceWindow

Update the current SWD in the workspace window, eg after changing the status of an overlay.

Syntax `GisUpdateWorkspaceWindow ( )`

Available MM, ME, MD

❍ WndArrangeIcons

Arrange any iconised windows in the main frame window.

Syntax `GisWndArrangeIcons ( )`

Notes This is equivalent to the **Window>Arrange Icons** command.

Available MM, ME, MD

❍ WndCascade

Cascade any non-iconised windows in the main frame window. Windows will overlap, with the title bars showing.

Syntax `GisWndCascade ( )`

Notes This is equivalent to the **Window>Cascade** command.

Available MM, ME, MD

❍ WndTile

Vertically tile any non-iconised windows in the main frame window. Windows will be arranged side by side, and with best fit.

Syntax `GisWndTile ( )`

Notes This is equivalent to the **Window>Tile Vertically** command.

Available MM, ME, MD

❍ WndTileHorizontal

Horizontally tile any non-iconised windows in the main frame window. Windows will be stacked on top of each other, and with a best fit.

Syntax `GisWndTileHorizontal ( )`

Notes This is equivalent to the **Window>Tile Horizontally** command.

Available MM, ME, MD

❍ WorkspaceClose

Save and close the current workspace file.

Syntax `GisWorkspaceClose ( savecode )`

Arguments `savecode` SHORT INTEGER

`SIS_NOSAVE`	do not save any modified datasets and/or SWDs
`SIS_SAVE`	save all modified datasets and/or SWDs
`SIS_PROMPTSAVE`	prompt the user with each modified dataset and/or SWD

Example `GisWorkspaceClose SIS_SAVE`
saves the current workspace, saving all changes made

Available MM, ME, MD

❍ WorkspaceNew

Create a new workspace file, after saving the existing workspace file.

Syntax `GisWorkspaceNew ( filename, savecode )`

Arguments `filename` STRING
the workspace file to be created

`savecode` SHORT INTEGER

`SIS_NOSAVE`	do not save any modified datasets and/or SWDs
`SIS_SAVE`	save all modified datasets and/or SWDs
`SIS_PROMPTSAVE`	prompt the user with each modified dataset and/or SWD

Example `GisWorkSpaceNew "c:\projects\Medway.sis", SIS_NOSAVE`
creates a new workspace Medway.sis after closing and not saving any changes made to the current workspace

Available MM, ME, MD

❍ WorkspaceOpen

Open an existing workspace file, after saving the existing workspace.

Syntax `GisWorkspaceOpen ( filename, savecode )`

Arguments `filename` STRING
the workspace file to be opened

`savecode` SHORT INTEGER

`SIS_NOSAVE`	do not save any modified datasets and/or SWDs
`SIS_SAVE`	save all modified datasets and/or SWDs
`SIS_PROMPTSAVE`	prompt the user with each modified dataset and/or SWD

Example `GisWorkspaceOpen "c:\projects\Medway.sis", SIS_SAVE`
opens the workspace Medway.sis saving the currently open workspace

Available MM, ME, MD

❍ WorkspaceSave

Save the current workspace.

Syntax `GisWorkspaceSave ( )`

Notes This method will not save any modified datasets or SWDs.

Available MM, ME, MD

❍ ZoomExtent

Zoom the view to the extents of all of the items in all of the visible, hittable, and editable overlays.

Syntax `GisZoomExtent ( )`

Available MM, ME, MD, OV, OM, OD, ASC

❍ ZoomView

Zoom the current window by a scale factor.

Syntax `GisZoomView ( f )`

Arguments *f* DOUBLE
the zooming factor, eg 2.0 to zoom out, or 0.5 to zoom in

Example `GisZoomView 2.0`

Available MM, ME, MD, OV, OM, OD, ASC

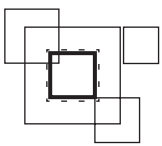
■ Notes

These notes apply to a number of methods.

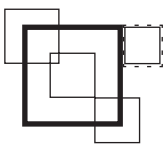
◆ Geometry tests

The following geometry tests are available. The selected item is shown by a solid line, the results use a dashed line.

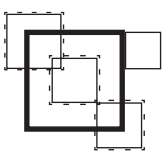
`SIS_GT_EQUAL` items must share the same outline and space as the selected item

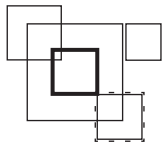
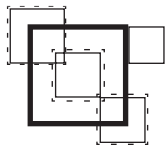
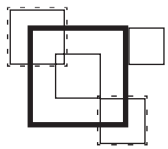
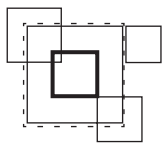
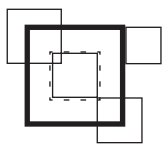
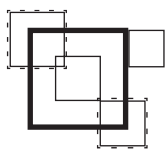


`SIS_GT_DISJOINT` items must be completely separate, with space between them and the selected item



`SIS_GT_INTERSECT` items which cross are adjacent or within the selected item

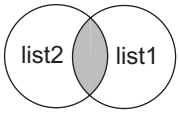


SIS_GT_TOUCH	items which touch (if only by one vertex) but do not cross the item	
SIS_GT_CROSS	items which are wholly or partly overlapped by the selected item	
SIS_GT_CROSSBY	items which partly overlap the selected item	
SIS_GT_WITHIN	items must be completely surrounding the selected item	
SIS_GT_CONTAIN	items must be completely within the selected item	
SIS_GT_OVERLAP	items which partly overlap the selected item	

◆ Boolean tests

The following geometry tests are available:

`SIS_BOOLEAN_AND` get the overlap between the areas in *`list`*. See the **Intersect Selected Areas** command.



`SIS_BOOLEAN_OR` merge the areas in *`list`* into a single area item. See the **Union Areas** command.

`SIS_BOOLEAN_XOR` form areas made up of alternate overlapping sub-areas of areas in *`list`*.
See the **Exclusive Or** command.

`SIS_BOOLEAN_DIFF` take bites out of the dominant area in *`list`*, using other areas in *`list`*.
See the **Subtract Area** command.

◆ Object types

The *`objectType`* argument used in a number of methods can take the following values:

`SIS_OT_CURITEM`	the current open item
`SIS_OT_DEFITEM`	the default item
`SIS_OT_DATASET`	datasets
`SIS_OT_OVERLAY`	overlays
`SIS_OT_WINDOW`	the window
`SIS_OT_NOL`	named object libraries
`SIS_OT_FTABLE`	the current feature table
`SIS_OT_SCHEMA`	the current schema
`SIS_OT_SCHEMACOLUMN`	a column in the current schema
`SIS_OT_THEME`	the current theme
`SIS_OT_THEMECOMPONENT`	a component in the current theme
`SIS_OT_PRINTER`	printer
`SIS_OT_SYSTEM`	system variables
`SIS_OT_OPTION`	system-wide Boolean options

◆ Serial numbers

Some of the methods use an *nDataset* argument to refer to the serial number of a dataset. The serial number can be obtained from the _nDataset& property of an overlay, or from the GetDataset, GetDatasetContainer, or FindExternalDataset methods.

The serial number cannot be relied upon to be identical in each session or if the dataset is removed and re-added. The number returned should therefore not be stored for long-term use.

◆ Connecting to databases

The *connect* argument in methods such as CreateDbOverlay and CreateOpenGisSqlOverlay enables Cadcorp SIS to connect to the database containing the data to be mapped. The methods you can use to connect to a database are:

- DAO (Microsoft Data Access Objects)
- ODBC (Open Database Connectivity)
- Oracle (direct driver)
- ADO (Microsoft ActiveX Data Objects)

Using DAO, *connect* should be DAO, followed by a semi-colon, followed by the pathname of the database.

Using ODBC, *connect* should be ODBC, followed by a semi-colon, followed by a combination of the following components:

```
DSN=DataSourceName
DATABASE=DatabasePathName
UID=UserName
PWD=Password
LOGINTIMEOUT=seconds
```

Components should be separated by semi-colons.

If the DSN exists, a connection will be established. The *Username* and *Password* components are necessary only if the database to which the DSN refers requires them. If they are not required, you can omit these components altogether, or provide an empty string.

If the DSN does not exist, you can create one on the fly by providing full connection details. For example:

```
ODBC;DSN=Anything;DATABASE=C:\test.mdb;UID=MyName;PWD=topsecret
```

If insufficient information is provided, the standard Windows Connection dialog will be displayed for the user to enter the required information.

Using Oracle, *connect* should be Oracle, and the following components:

```
Server=HostString
User=UserName
Password=Password
```

Components should be separated by semi-colons.

Using ADO, *connect* should be ADO, followed by a semi-colon, followed by the ADO Connection string.

For details on creating ADO connection strings, refer to your programming language's documentation.

Chapter 8

Examples

■ Introduction 223
■ General examples 223
■ Graphics 230
■ Cadcorp SIS system commands 230
■ Text 235
■ Windows 239
■ Overlays 241
■ Object properties 253
■ Named lists 257
■ Filters 261
■ Groups 267
■ Named object libraries 270
■ Spatial searches 275
■ Schemas 287
■ Themes 289
■ Printing 295

■ Introduction

This chapter provides Visual Basic code samples that illustrate some common operations. Use these as the basis for your own projects.

All the examples show the code required to perform the task being illustrated. They are not complete Visual Basic projects in their own right.

For details on how to set up your Visual Basic project, ➲Chapter 2: **"Customising with GisLink", Getting started**, page 5.

■ General examples

◆ Using triggers

This code shows you how to use a trigger during an AComLine command to close the polyline on hitting the Enter key. You must have an SWD open to use this routine.

This example requires the following form level declarations:

```
Private LineEnter As Boolean
Private LineEnd As Boolean

Line End Trigger
Private Sub DrawLineEnter_Click()
```

```
      ' Set flag for line enter trigger
   LineEnter = True
End Sub

Line Enter Trigger
Private Sub DrawLineEnd_Click()
   ' Set flag for line enter trigger
   LineEnd = True
End Sub

Private Sub BtnGeneralTrigger_Click()
   Me.WindowState = vbMinimized
   ' Every command within SIS generates a Trigger.
   ' One-shot commands have Succeeded and Failed triggers.
   ' Callback commands have Snap, KeyBack, KeyEnter, KeyTab and End triggers.

   ' Register triggers so we are notified when user does something
   ' - buttons with captions the same
   GisRegisterTrigger "AComLine::KeyEnter", "DrawLineEnter"
   GisRegisterTrigger "AComLine::End", "DrawLineEnd"

   ' Set flags to false, so we can tell when a trigger is used
   ' Triggers are flagged by the click event of trigger buttons
   LineEnd = False
   LineEnter = False

   GisSwitchCommand "AComLine"
   ' Start the command we have patched

   GisRelease
   ' Let user enter position into GIS window

   ' Wait for one of the triggers to be used
   Do
      DoEvents
      If LineEnter Then
         Exit Do
      ElseIf LineEnd Then

       ' Escape was pressed
         GisRelease
         Exit Sub
      End If
   Loop

' Clear Triggers
GisRegisterTrigger "AComLine::End", ""
GisRegisterTrigger "AComLine::KeyEnter", ""
GisSwitchCommand "AComSelectSlide"
GisRelease

End Sub
```

◆ Set up the Cadcorp SIS user environment

This example shows how to set up the Cadcorp SIS user environment under program control. It creates a workspace called `workspace` in the same directory as the program.

You do not need an SWD open to use this routine. If you have, you may be prompted to save your work.

```
Private Sub BtnGeneralEnviro_Click()

   ' Creates a workspace called "Workspace.sis" in your C: drive,
   ' prompting if any previous data is to be saved
   GisWorkspaceNew "\Workspace.sis", SIS_PROMPTSAVE

   ' On error (workspace already exists), try to open existing workspace
   If GisGetInt(SIS_OT_SYSTEM, 0, "_ExecError&") <> 0 Then
      GisWorkspaceOpen "\Workspace.sis", SIS_PROMPTSAVE
   End If

   ' Set default & view projections
   GisSetDefaultPrj "Latitude/Longitude.WGS 84"
   GisSetViewPrj "Cylindrical.Transverse Mercator"

   GisRelease

End Sub
```

◆ Measure a route

This example shows you how to measure the route along a topological network between two user given snaps. You must have an SWD open and a topological network displayed to use this routine. (Use the sample OSCAR file SS78NE.NTF.)

```
Private Sub BtnGeneralRoute2_Click()
   Dim arg As Long
   Dim x1 As Double, y1 As Double, z1 As Double
   Dim x2 As Double, y2 As Double, z2 As Double
   Dim RouteLength As String
   Dim ScaleFactor As Double

   ' Measure a route
   ' First snap for start point
   Do
      arg = GisGetPosEx(x1, y1, z1)
   Loop Until arg = SIS_ARG_ESCAPE Or arg = SIS_ARG_POSITION

   If arg = SIS_ARG_ESCAPE Then
      GisRelease
      Exit Sub
   End If

   ' Second snap for end point
   Do
   arg = GisGetPosEx(x2, y2, z2)
   Loop Until arg = SIS_ARG_ESCAPE Or arg = SIS_ARG_POSITION

   If arg = SIS_ARG_ESCAPE Then
      GisRelease
      Exit Sub
   End If
```

```
    ' Measure the route
    GisMeasureRoute x1, y1, z1, x2, y2, z2, "_length#", "", "", True
    ' True means "Copy route as a graphical line"

    GisOpenSel 0
    Routelength = GisGetFlt(SIS_OT_CURITEM, 0, "_length#")
    Routelength = "Length= " & Routelength

    ' Find the length of the route line and place it down as a text block
    Scalefactor = GisGetFlt(SIS_OT_WINDOW, 0, "_displayScale#")
    GisCreateBoxText x1, y1, z1, Scalefactor / 250, Routelength
    GisCloseItem
    GisRedraw SIS_CURRENTWINDOW

    GisRelease

End Sub
```

◆ Move items around the map base

This example shows you how Cadcorp SIS can quickly redraw items called *sprites* which lie on top of the normal graphics. This is especially useful in Command and Control applications. You should have a 1:50 000 colour raster map displayed to use this routine. (Use the sample file SS68.BMP.)

```
Private Sub BtnGeneralMoveItems_Click()
  Dim viewExtent As String
  Dim PauseTime As Integer
  Dim x1 As Double, y1 As Double, z1 As Double
  Dim x2 As Double, y2 As Double, z2 As Double
  Dim dx As Double, dy As Double, XX As Double
  Dim YY As Double
  Dim list As String, VehicleNo As String
  Dim X As Integer, Y As Integer, I As Integer

   ' Redraw a named list called sprites
   ' Used for command and control applications
   ' Where entities are moved over a Raster basemap

   ' Override the default overlay thresholds
   GisSetFlt SIS_OT_OVERLAY, 0, "_scalemax#", 1000000
   GisSetFlt SIS_OT_OVERLAY, 0, "_scalemin#", 1

   ' Set status of dataset to Visible
   GisSetInt SIS_OT_OVERLAY, 0, "_status&", SIS_VISIBLE

   ' Insert internal overlay called sprites
   GisCreateInternalOverlay "Sprites", 1

   ' Override the default overlay thresholds
   GisSetFlt SIS_OT_OVERLAY, 1, "_scalemax#", 1000000
   GisSetFlt SIS_OT_OVERLAY, 1, "_scalemin#", 1

   ' Set status of dataset to Editable
   GisSetInt SIS_OT_OVERLAY, 1, "_status&", SIS_EDITABLE
```

```
' set the dataset scale
DatasetNo& = GisGetInt(SIS_OT_OVERLAY, 1, "_nDataset&")
GisSetFlt SIS_OT_DATASET, DatasetNo&, "_scale#", 50000

' Zoom to the extent of the datasets
GisZoomExtent
viewExtent = GisGetViewExtent

' Pause the program for 5 seconds to allow the raster backdrop to be drawn
PauseTime = 5
' Set duration
Start = Timer
' Set start time
Do While Timer < Start + PauseTime
Loop

' Split out the co-ordinates
GisSplitExtent x1, y1, z1, x2, y2, z2, viewExtent
dx = x2 - x1
dy = y2 - y1

' Create 10 groups
For X = 1 To 10
   list = "List" & Str(X)
   XX = x1 + (Int(Rnd * dx))
   YY = y1 + (Int(Rnd * dy))
   ' Create group
   GisCreateGroup ""
   ' Place down point
   GisCreatePoint 0, 0, 0, "Circle", 0, 1
   ' Place down text
   GisCreateBoxText 250, 0, 0, 250, "No=" & Str(X)
   ' Place group down at a random location within the display
   GisPlaceGroup XX, YY, 0
   ' Add attribute to group
   GisSetInt SIS_OT_CURITEM, 0, "VehicleNo&", X + 1
   ' Close item to confirm changes
   GisCloseItem
   ' Scan editable items to find group and place in list
   VehicleNo = "VehicleNo&=" & Str(X)
   GisCreatePropertyFilter "Group", VehicleNo
   GisScan list, "E", "Group", ""
Next X

' Create a "*Sprites" list from all groups
GisCreateListFromOverlay 1, "*Sprites"

' Move the 10 groups around the screen 10 times
' using Redraw SIS_CURRENTSWD + SIS_QUICK_REDRAW option to stop flicker
For I = 1 To 10
   For Y = 1 To 10
      list = "List" & Str(Y)
      GisMoveList list, (Int(Rnd * 1000) - 500),
        (Int(Rnd * 1000) - 500), 0, 0, 1
   Next Y
   GisRedraw SIS_CURRENTSWD + SIS_QUICK_REDRAW
Next I
```

```
    GisRedraw SIS_CURRENTSWD
    GisRelease

End Sub
```

◆ Pan across the map base

This example shows you how to pan across the map with user-given snap positions.

```
Private Sub BtnGeneralPanMap_Click()
    Dim arg As Long
    Dim DisplayExtent As String
    Dim x1 As Double, y1 As Double, z1 As Double
    Dim x2 As Double, y2 As Double, z2 As Double
    Dim dx As Double, dy As Double
    ' Pan the Map Base by to user defined Snaps

    ' Populate prompt box on SIS window informing the user what to do
    GisPrompt "Snap reference point"

    ' Find positions to pan from and to
    ' First snap
    Do
        arg = GisGetPosEx(x1, y1, z1)
    Loop Until arg = SIS_ARG_ESCAPE Or arg = SIS_ARG_POSITION

    If arg = SIS_ARG_ESCAPE Then
        GisRelease
        Me.WindowState = vbNormal
        Exit Sub
    End If

    ' Inform the user what to do next
    GisPrompt "Snap location to move to"

    ' Second snap
    Do
        arg = GisGetPosEx(x2, y2, z2)
    Loop Until arg = SIS_ARG_ESCAPE Or arg = SIS_ARG_POSITION

    If arg = SIS_ARG_ESCAPE Then
        GisRelease
        Exit Sub
    End If

    ' Calculate distance moved
    dx = x2 - x1
    dy = y2 - y1

    ' Calculate extent of current window
    DisplayExt = GisGetDisplayExtent
```

```
    ' Move to new panned extent
    GisSplitExtent x1, y1, z1, x2, y2, z2, DisplayExt
    GisSetViewExtent x1 - dx, y1 - dy, z1, x2 - dx, y2 - dy, z2

    GisRelease
End Sub
```

◆ Pan across the map base by a given amount

This example shows you how to pan a map by a given amount in a pre-defined direction. In this example, the current window is panned by one quarter of its current size in a north-westerly direction.

```
Private Sub BtnGeneralPanByAmount_Click()
    Dim x1 As Double, y1 As Double, z1 As Double
    Dim x2 As Double, y2 As Double, z2 As Double
    Dim dx As Double, dy As Double

    ' Pan the current window 1/4 screen to the North West
    ' Calculate extent of current view
    DisplayExt = GisGetViewExtent
    GisSplitExtent x1, y1, z1, x2, y2, z2, DisplayExt

    ' Calculate the distance to move
    dx = (x2 - x1) / 4
    dy = (y2 - y1) / 4

    ' Move to new view
    GisSetViewExtent x1 + dx, y1 + dy, z1, x2 + dx, y2 + dy, z2

    GisRelease

End Sub
```

◆ Zoom the map base

This example shows you how to zoom into a map by a given amount. There are various methods of zooming. Here, we ask the user to enter a value.

```
Private Sub BtnGeneralZoomByAmount_Click()
    Dim ZoomFactor As String
    Dim DisplayScale As Double
    Dim dx As Double, dy As Double

    ' Zoom in by a user defined factor on screen centre
    ZoomFactor = InputBox("ZoomIn By Factor", "Zoom In", "2")
    Displayscale = GisGetFlt(SIS_OT_WINDOW, 0, "_displayScale#")
    GisSetFlt SIS_OT_WINDOW, 0, "_displayScale#", _
      Displayscale / Val(ZoomFactor)
    GisRelease

    Exit Sub
```

```
    ' Below are examples of the System zoom

    ' System zoom in by a factor of two
    GisDoCommand "AComZoomIn2"

    ' System zoom out by a factor of two
    GisDoCommand "AComZoomOut"

    ' System zoom in by a factor
    GisDoCommand "AComZoomIn"
End Sub
```

■ Graphics

The API provides two ways in which you can create graphics under program control.

The first and simplest is to invoke the Cadcorp SIS system commands for graphics creation. These are the commands generally found on the **Construct** menu in Cadcorp SIS Map Manager, Map Editor, and Map Modeller.

The second way is to use the methods provided in the API.

Both ways have their advantages, and examples of both are given here.

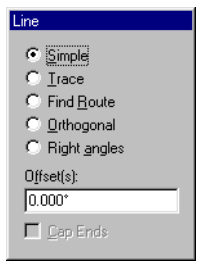
■ Cadcorp SIS system commands

◆ GisLink

Why would you want to write a GisLink program to call a command which is already available on the Cadcorp SIS menu?

One reason is that you might want to place the command within your own menu structure, rather than rely on your user negotiating the standard menus. In the following example, the **Construct>Line** and **Construct>Area** commands are removed from Map Editor, and a custom Draw menu added. This new menu offers only three simple commands for creating Points, Lines, and Areas.

Cadcorp SIS Map Editor provides the AComLineEx command for drawing lines, and the AComAreaEx command for drawing areas. These commands display a non-modal dialog from which the user can select the method by which they want to draw a line or area:

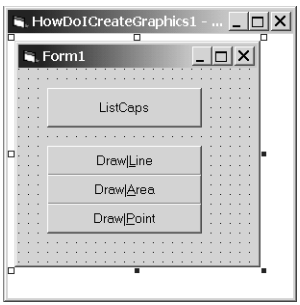


In this example, we will use the simpler AComLine and AComArea commands, which are the equivalent of the Simple option above.

First, remove the Line and Area menu items from Map Editor, then add three new commands. This should be written in the Click event of the button captioned ListCaps:

```
Private Sub cmdListCaps_Click()
  GisAllowCommands SIS_COM_REMOVE, "AComLineEx,AComAreaEx"
  GisAddCommand "&Draw|&Line", "Constructs a new line item", "Item", 0, _
   -1, "", ""
  GisAddCommand "&Draw|&Area", "Constructs a new area item", "Item", 0, _
   -1, "", ""
  GisAddCommand "&Draw|&Point", "Draws new point items", "Item", 0, -1, _
   "", ""
  GisRelease
End Sub
```

Next, add three buttons to your startup form, and caption each of them exactly as the command name:



Notice that Visual Basic does not underline the D in Draw, even though the ampersand (&) is present.

Next, add code to the `Click` event of each of these three command buttons:

```
Private Sub cmdLine_Click()
  GisSwitchCommand "AComLine"
  GisRelease
End Sub

Private Sub cmdArea_Click()
  GisSwitchCommand "AComArea"
  GisRelease
End Sub

Private Sub cmdPoint_Click()
  GisSwitchCommand "AComPoint"
  GisRelease
End Sub
```

◆ Cadcorp SIS Control

Applications written using the Cadcorp SIS Control offer only the functionality you as a programmer choose to provide. In many cases, the functionality required is the same that available in Map Manager/Map Editor/Modeller, so the easiest way to provide it is by invoking the Cadcorp SIS system commands.

Here is an example of the **Construct>Line**, **Construct>Area** and **Construct>Point** commands added to a Cadcorp SIS Control application in the form of three command buttons. This example uses the extended AComLineEx and AComAreaEx commands, rather than the simplified commands used in the GisLink example above.

```
Private Sub cmdLine_Click()
   Sis.DoCommand "AComLine"
End Sub

Private Sub cmdArea_Click()
   Sis.DoCommand "AComArea"
End Sub

Private Sub cmdPoint_Click()
   Sis.DoCommand "AComPoint"
End Sub
```

◆ GisLink and Cadcorp SIS Control

The Cadcorp SIS API provides several dozen functions for creating graphical items. The following examples describe seven of these functions:

- `MoveTo`
- `LineTo`
- `StoreAsLine`
- `StoreAsArea`
- `BulgeTo`
- `CreateRectangle`
- `CreatePoint`

There are no significant differences in the way you would use these methods in a GisLink customisation or a Cadcorp SIS Control application. These examples use the `GisLink` prefix to each method, and include the `GisRelease` method. If you are programming for the Cadcorp SIS Control, you do not need the prefix or `GisRelease`.

◆ Straight lines

The following code shows you how to draw graphics using `GisMoveTo` and `GisLineTo`. A 3 metre wide cross shape is drawn, centred on a user-given position.

The `GisMoveTo` and `GisLineTo` methods are best used when you have a defined shape to draw, or if you are reading x, y, and z values from an external source, such as a file of comma-separated values.

```
Private Sub cmdMoveToLineTo_Click()
   Dim lSnap As Long
   Dim X As Double, Y As Double, Z As Double

   GisPrompt "Position for centre of the shape"

   ' Loop until the user has snapped a position
   Do
      lSnap = GisGetPosEx(X, Y, Z)
   Loop Until lSnap = SIS_ARG_POSITION

   ' Draw cross shape
   GisMoveTo X - 1.5, Y + 0.5, 0
```

```
    GisLineTo X - 0.5, Y + 0.5, 0
    GisLineTo X - 0.5, Y + 1.5, 0
    GisLineTo X + 0.5, Y + 1.5, 0
    GisLineTo X + 0.5, Y + 0.5, 0
    GisLineTo X + 1.5, Y + 0.5, 0
    GisLineTo X + 1.5, Y - 0.5, 0
    GisLineTo X + 0.5, Y - 0.5, 0
    GisLineTo X + 0.5, Y - 1.5, 0
    GisLineTo X - 0.5, Y - 1.5, 0
    GisLineTo X - 0.5, Y - 0.5, 0
    GisLineTo X - 1.5, Y - 0.5, 0
    GisLineTo X - 1.5, Y + 0.5, 0

    GisStoreAsArea
    GisRelease

End Sub
```

The `GisStoreAsArea` method converts the linework to an area item. If you want to keep the lines, use `GisStoreAsLine` instead.

When the graphics have been created, the newly created item will become current. When an item is current in Visual Basic, you are able to set and retrieve its properties and attributes, and access its geometry.

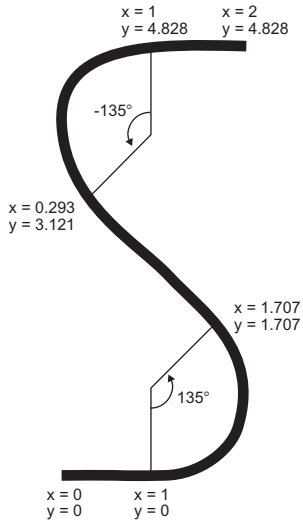
Cadcorp SIS Control and the status bar

The `GisPrompt` method displays a prompt in the status bar of Cadcorp SIS applications. Because the Cadcorp SIS Control does not provide a status bar, you must write your own code to prompt the user to enter a position.

◆ Curves

The API provides the `GisBulgeTo` method for creating lines or areas containing circular arcs. This can be incorporated into your code in the same way as `GisLineTo`, except that the `GisBulgeTo` method requires an additional argument, the arc angle. This angle must be given in radians in an anticlockwise direction.

The following code creates this S shape:



There are 2π radians in 360˚ (π radians = 180˚), so you must multiply degrees by π/180 to obtain radians:

$$135° \times \frac{\pi}{180} = 2.356\text{radians}$$

```
GisMoveTo 0, 0, 0
GisLineTo 1, 0, 0
GisBulgeToTo 2.356, 1.707, 1.707
GisLineTo 0.293, 3.121, 0
GisBulgeTo -2.356, 1, 4.828
GisLineTo 2, 4.828, 0
GisStoreAsLine
GisRelease
```

◆ Rectangles

To create a rectangular area item you do not have to specify the co-ordinates of all four corners of the rectangle. The `Rectangle` method allows you to specify the minimum and maximum X and Y co-ordinates to create the rectangle in a single command. The following code creates a 3m by 1m rectangle centred at 0, 0:

```
GisRectangle -0.5, -1.5, 0.5, 1.5
```

The `Rectangle` method lets you to create only flat (2 dimensional) shapes, whereas the `MoveTo`/`LineTo` methods accept a Z co-ordinate for the third dimension.

◆ Placing symbols

To place a symbol at a given position on the map, use the `Point` method, supplying the position, the required symbol (shape), the angle of rotation, and the scale.

Remember that the size of a symbol on a map depends not only on the scale of the symbol, but also on the dataset scale. The following example uses the Star shape from the standard Cadcorp SIS library. This was created to appear best at dataset scales of around 1:1000 or 1:2000 when placed at a scale of 1. If the dataset scale were 1:10 000, the scale of the Star shape would need to be 0.1 to achieve the same visual appearance. Additionally, setting a negative scale will cause the symbol to appear the same size on the screen (or on a print) irrespective of the display scale. In other words, as the user zooms out, the symbol stays the same size.

The following example creates a point at 0, 0, 0, assigns a shape to it, and sets it at an angle of 45 degrees with a negative scale to fix its screen size:

```
Dim Pi As Double
Pi = 3.1415926
GisCreatePoint 0, 0, 0, "Star", Pi / 4, -2
GisRelease
```

■ Text

◆ GisLink and Cadcorp SIS Control

The Cadcorp SIS API provides four methods for creating text:

- `CreateText`
- `CreateBoxText`
- `CreateBoxLabel`
- `CreateLineText`

The `CreateText` method is available to all levels of the Cadcorp SIS Control, the Cadcorp SIS Active Server Component, and all Cadcorp SIS applications except Map Viewer.

The remaining three text methods are available to the Manager and Modeller levels of the Cadcorp SIS Control, the Cadcorp SIS Active Server Component, and all Cadcorp SIS applications except Map Viewer.

There are no significant differences in the way you would use these methods in a GisLink customisation or a Cadcorp SIS Control application. The examples in this chapter use the `GisLink` prefix to each method, and include the `GisRelease` method. If you are programming for the Cadcorp SIS Control, you do not need the prefix or `GisRelease`.

◆ Create Point Text

The AComText command in the Cadcorp SIS applications displays a dialog enabling the user to enter text, setting its font, size, and justification before interactively placing the text on the map. There are many occasions when you want to place text on the map under program control. The `CreateText` method provides a simple means of placing a text string at a known position:

```
GisCreateText 0, 0, 0, "Sample Text"
```

An example of the use of the `CreateText` method is to label land parcels with the area of the parcel in Hectares. A Hectare (abbreviated Ha.) is 10 000 square metres. The label is to be in 12 point Times New Roman, justified so that the position is at the bottom left of the text. The example assumes that the land parcel to be labelled has been selected by the user, and the label is to be placed at the origin of the area:

```
Dim dArea As Double
Dim dHa As Double, dX As Double, dY As Double

' make the selected item current:
GisOpenSel 0
' retrieve the area in square metres:
dArea = GisGetFlt(SIS_OT_CURITEM, 0, "_area#")
' convert to hectares:
dHa = dArea / 10000
' retrieve the centroid of the area item
dX = GisGetFlt(SIS_OT_CURITEM, 0, "_ox#")
dY = GisGetFlt(SIS_OT_CURITEM, 0, "_oy#")
' create the text:
GisCreateText dX, dY, 0, Format(dHa, "00.00") & " Ha."
' set the attributes of the text:
GisSetInt SIS_OT_CURITEM, 0, "_text_alignH&", SIS_LEFT
GisSetInt SIS_OT_CURITEM, 0, "_text_alignV&", SIS_BOTTOM
GisSetStr SIS_OT_CURITEM, 0, "_font$", "Times New Roman"
GisSetInt SIS_OT_CURITEM, 0, "_point_height&", 12
' apply the changes:
GisUpdateItem
GisRelease
```

◆ Create box text

Box text differs from point text in that it has a ‘geographical’ size, rather than a point size. Box text is created in a very similar way to point text, except a height in metres is required for the text:

```
GisCreateBoxText dX, dY, 0, 10, "Example Text"
```

◆ Rotate box text

Point text inherits some of the properties of a point item. For example, users can place point text at an angle. You can set this angle programmatically for points and point text by setting the `_angleDeg#` property.

Box text does not have this property, so it cannot simply be rotated, other than by the user with the AComRotate command. An easy way to programmatically rotate box text is to temporarily convert it to point text, set its rotation angle, then convert it back to box text:

```
' create some box text for this example:
GisCreateBoxText dX, dY, 0, 10, "Example Text"
' make it selected so the ACom… command can see it:
GisSelectItem
' convert it to point text:
GisCallCommand "AComBoxToText"
' re-open the item to make it current again:
GisOpenSel 0
' rotate it
GisSetFlt SIS_OT_CURITEM, 0, "_angleDeg#", 45
' convert it back to box text:
GisCallCommand "AComTextToBox"
' deselect it (optional):
GisSelectItem
GisRelease
```

Similarly, if box text has been rotated, you can temporarily convert it to point text to retrieve the angle of rotation, then convert it back to box text again.

◆ Create line text

Line text inherits many of the attributes of a line item (length, number of vertices, and so on) and many attributes of a text item (alignment, text, point height, and so on). In addition, it holds two attributes which control the display of text along the line:

`_draw_line&` is the line itself to be drawn, or just the text?

`_even_spacing&` should letters be evenly spaced along the line?

To create line text, use the `CreateLineText` method, then set the attributes to change the appearance of the line text:

```
' open the selected item (assuming it to be a line item)
GisOpenSel 0
' replicate the current item as line text
GisCreateLineText "Text along a wavy path"
GisSetInt SIS_OT_CURITEM, 0, "_even_spacing&", True
GisSetInt SIS_OT_CURITEM, 0, "_draw_line&", False
GisUpdateItem
GisRelease
```

The selected line is not converted to line text. It is retained and a new line text item is created. So you can create line text from any editable or hittable lines, without modifying the underlying data.

◆ Create box label text

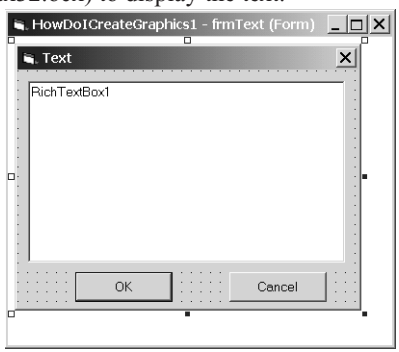
Box label text is similar to box text, but provides a leader line from the text to a specified position.

```
GisCreateBoxLabel 0, 0, 0, 3, "This is a label", 1, 1, 0
GisRelease
```

In this example, the 0, 0, 0 co-ordinates are for the origin of the text, ie its justification point. The second set of co-ordinates, 1, 1, 0 are the position of the end of the leader line. The beginning of the leader line is automatically placed at the middle of the nearest edge of the surrounding box.

◆ Editing text

To edit text under program control, modify the `_text$` property of the text item, whether it is point text, box text, line text or label text. If you want to provide users of your program with a dialog box in which they can edit the text, you should retrieve the value of this property, and its other text-related attributes, and use the Microsoft Rich Text Box Control (richtx32.ocx) to display the text.



The following example shows code for an 'edit text' command (although you would have to add some more code to produce a fully working text editor). The code assumes you have added a form to your project, and named it `frmText`. On that form, you should add a Rich Text Box Control (using **Project>Components…** on the Visual Basic menu).

```
Private Sub cmdEditText_Click()
  Dim sText As String, sFont As String, sClass As String
  Dim lJust As Long, lHeight As Long
  Dim bBold As Boolean, bItalic As Boolean

  GisOpenSel 0
  sText = GisGetStr(SIS_OT_CURITEM, 0, "_text$")
  sFont = GisGetStr(SIS_OT_CURITEM, 0, "_font$")
  bBold = GisGetInt(SIS_OT_CURITEM, 0, "_text_bold&")
  bItalic = GisGetInt(SIS_OT_CURITEM, 0, "_text_italic&")
  lJust = GisGetInt(SIS_OT_CURITEM, 0, "_text_alignH&")
  sClass = GisGetStr(SIS_OT_CURITEM, 0, "_class$")
  If sClass = "Text" Or sClass = "LineText" Then
    lHeight = GisGetInt(SIS_OT_CURITEM, 0, "_point_height&")
```

```
  Else
    ' assume 10pt text
    lHeight = 10
  End If

  With frmText.RichTextBox1
    .text = sText
    .font = sFont
    .SelStart = 0
    .SelLength = Len(.text)
    .SelBold = bBold
    .SelItalic = bItalic
    .SelFontSize = lHeight
    Select Case lJust
      Case SIS_LEFT
        .SelAlignment = rtfLeft
      Case SIS_CENTRE
        .SelAlignment = rtfCenter
      Case SIS_RIGHT
        .SelAlignment = rtfRight
    End Select
  End With

  frmText.Show vbModal, Me
  GisRelease
End Sub
```

If your program provided tools to modify the font, size, and so on of the text within the Rich Text Box Control, you apply these changes to the text within Cadcorp SIS by reversing the above process, using the `GisSetStr` and `GisSetInt` methods.

Text in Cadcorp SIS can be multi-line. To insert a line break into text in Visual Basic, use the pre-defined constant `vbCrLf`, which is a combination of the ASCII character codes 13 and 10. For example:

```
SNewText= "Line one" & vbCrLf & "Line two"
```

Cadcorp SIS does not support all the features of rich text format (RTF).

■ Windows

The Cadcorp SIS desktop applications - Map Viewer, Map Manager, Map Editor and Map Modeller - present a *multiple document interface* (MDI), enabling the user to display and manipulate several map views simultaneously. When customising Cadcorp SIS using GisLink, each of these map views is accessible to you, with one view at a time being regarded as the ‘current window’.

Using the Cadcorp SIS Control, each instance of the Control within an application is a window. If you are writing an application which uses multiple Cadcorp SIS Controls, the name you give each control enables you to access it as the current window. If your application presents a multiple document interface, where there are potentially many ‘child’ forms, each containing a Cadcorp SIS Control, you must use the techniques provided in your chosen programming language to keep track of which control is the current window.

◆ The SIS_OT_WINDOW Object

When using GisLink to customise a Cadcorp SIS desktop application, the 'current window' is the map view which has focus. To access the properties of the current window use the `SIS_OT_WINDOW` object. The following code retrieves the number of overlays in the current window, and the position of the default overlay within the (zero-based) list of overlays.

GisLink example

```
lNumOverlays = GisGetInt(SIS_OT_WINDOW, 0, "_nOverlay&")
lDefault = GisGetInt(SIS_OT_WINDOW, 0, "_nDefaultOverlay&")
```

Cadcorp SIS Control example

```
lNumOverlays = Sis.GetInt(SIS_OT_WINDOW, 0, "_nOverlay&")
lDefault = Sis.GetInt(SIS_OT_WINDOW, 0, "_nDefaultOverlay&")
```

The second argument of the `GetInt` method is not required here, and should be set to zero.

◆ The SIS_OT_OVERLAY Object

Having retrieved the number of overlays in the current window, the properties of each overlay can be accessed using the `SIS_OT_OVERLAY` object. The second argument of the `GetInt` and `GetStr` methods determines the position in the overlays list of the overlay to be queried.

The following code retrieves the name and status of each overlay, and the serial number of the dataset contained by the overlay. The code shown is for GisLink customisations. If you are using the Cadcorp SIS Control you should modify your code accordingly.

```
lNumOverlays = GisGetInt(SIS_OT_WINDOW, 0, "_nOverlay&")
For lCount = 0 To lNumOverlays -1
  sName = GisGetStr(SIS_OT_OVERLAY, lCount, "_name$")
  lStatus = GisGetInt(SIS_OT_OVERLAY, lCount, "_status&")
  lSerial = GisGetInt(SIS_OT_OVERLAY, lCount, "_nDataset&")
Next lCount
The SIS_OT_DATASET Object
```

The overlay name retrieved above is the name which appears in the workspace window and the Overlays dialog in Cadcorp SIS desktop applications. If the dataset contained by the overlay is an external (*.bds) file, you may also want to retrieve the full pathname of the file. For this you will need to use the `SIS_OT_DATASET` object, referring to it by the serial number retrieved above.

```
lNumOverlays = GisGetInt(SIS_OT_WINDOW, 0, "_nOverlay&")
For lCount = 0 To lNumOverlays -1
 sName = GisGetStr(SIS_OT_OVERLAY, lCount, "_name$")
 lStatus = GisGetInt(SIS_OT_OVERLAY, lCount, "_status&")
 lSerial = GisGetInt(SIS_OT_OVERLAY, lCount, "_nDataset&")
 bExternal = GisGetInt(SIS_OT_OVERLAY, lCount, "_bExternal&")
 If bExternal = True then
  sFilename = GisGetStr(SIS_OT_DATASET, lSerial, "_name$")
 End If
Next lCount
```

◆ Opening, saving, and closing windows

A map window is generally referred to as a saved window definition, or SWD, even when it has not actually been saved yet. The methods for opening, saving and closing SWDs are different using GisLink and the Cadcorp SIS Control.

To create a new, empty SWD using GisLink, use the `GisSwdNew` method. This new map window will not contain any overlays, and will adopt the current default map projection.

There is no equivalent method for use with the Cadcorp SIS Control, because the Control itself *is* an empty SWD.

To open a saved SWD using GisLink, use the `GisSwdOpen` method. This will open the named SWD as a new map window in the SIS desktop application, restoring the viewing extent to display the same area of map as when it was saved:

```
GisSwdOpen "C:\Temp\temp.swd", 0
```

Because the Cadcorp SIS Control is an empty SWD, the equivalent method is `LoadSwd`, which loads the nominated SWD file into the Cadcorp SIS Control, replacing any SWD previously displayed:

```
Sis.LoadSwd "C:\Temp\temp.swd"
```

Note that an SWD loaded into the Cadcorp SIS Control is a copy of the file. The file is not locked to other users. Only when an attempt is made to save the SWD is a check performed to gain write access. The Cadcorp SIS Control provides the `SaveSwd` method, which is the equivalent of a ‘save as’ process:

```
Sis.SaveSwd "C:\Temp\temp.swd"
Sis.SaveSwd "C:\Temp\temp2.swd"
```

GisLink programmers should use the `SwdSave` and `SwdSaveAs` methods:

```
GisSwdSave
GisSwdSaveAs "C:\Temp\temp2.swd"
```

■ Overlays

In many bespoke applications, you will want to access the overlays list of the current window to add, remove, or modify the overlays.

◆ Add backdrops and internal overlays

Using either GisLink or the Cadcorp SIS Control, the API provides a number of methods for adding particular types of overlay. These methods mirror the functionality offered to users of the desktop products through the Add Overlay Wizard. Some methods, such as those for adding a 'backdrop' overlay or adding an 'internal' overlay, require just a name and a position in the overlays list to insert it:

```
GisCreateBackdropOverlay 0, "GB National Grid"
GisCreateInternalOverlay "My Overlay", 1
```

When an overlay is created and inserted at a particular position, the overlay, if any, which occupied that position will be moved down the list, as will all subsequent overlays in the list. The overlay which is last on the list is the last to be drawn on screen.

◆ Add file-based overlays

Map data is often provided as a single file: Cadcorp SIS Base Dataset (bds); ESRI Shape File (*.shp); MapInfo Export (*.mif); raster image (*.bmp, *.jpg, *.tif, and so on).

All of these file-based datasets can be added as an overlay using the `InsertDataset` method:

```
GisInsertDataset "C:\Temp\Lakes.bds", 0
```

When a BDS file is inserted, its status is set to editable, if possible. If another user already has ownership of the file, or it is set to read-only, its status will be hittable. All other file-based datasets will be given hittable status.

◆ Add indexed overlays

Many GIS applications rely on the use of index datasets which enable many datasets to be contained within a single overlay. The `CreateIndexOverlay` method enables you to add an overlay which indexes all files in a directory which match a specific naming pattern. If the filename enables Cadcorp SIS to unambiguously determine the naming convention, you do not need to specify the namer to be used.

The following example creates an overlay containing an index dataset to display all GB Ordnance Survey LandLine files in the specified directory:

```
GisCreateIndexOverlay 2, "C:\Temp\OS\ss1234ne.ntf", "", _
  SIS_INDEX_OUTLINES + SIS_INDEX_PYRAMID
```

Here, position 2 is specified for the overlay. If there are fewer than two overlays already in the current window, the overlay will be placed at the lowest available position. If there are already more than two overlays, the new overlay will be inserted at position 2, and the remainder will be moved down the list.

The file path should exist, but the specific file (ss1234ne.ntf) is only a hint to Cadcorp SIS to enable the namer to be identified. If the specified path contains files which follow the example naming convention, the API method will succeed.

The `SIS_INDEX_OUTLINES` and `SIS_INDEX_PYRAMID` flags are constants defined in the GisLink.bas (GisLink) or SisConst.bas (Cadcorp SIS Control) file. In this example,

both flags are used, causing the resulting overlay to contain a 'gateway' item of each tile representing its extent, and to include files which are within the same hierarchical 'family'. In the case of GB Ordnance Survey LandLine files, this will include the 1:1250, 1:2500 and 1:10 000 scale map files.

◆ Add overlays which display database information

Several methods are provided for displaying overlays containing data which are stored in an external database. Each of these methods requires a *recordset* to be created, which is then used as one of the arguments.

To create an overlay which displays geo-referenced points directly from an external database, the `CreateDbPointOverlay` method is provided. This is the programmatic equivalent of the View Points overlay.

The `CreateDbBlobOverlay` method enables you to create an overlay in which graphics are stored as Binary Large Objects (Blobs) in a database. This is the programmatic equivalent of the View Blobs overlay created using the Overlays Wizard in the Cadcorp SIS desktop products.

One of the arguments required by these methods is a recordset defined using the `DefineRecordset` method.

The syntax of the `DefineRecordset` method is:

```
DefineRecordset (rs As String, connect As String, tables As String, columns
As String, aliases As String, sqlwhere As String)
```

The *`rs`* argument is the name of the recordset to be created or replaced. This recordset is specific to the Cadcorp SIS programming environment, and cannot be accessed using Microsoft ADO or other data manipulation tools.

The *`connect`* string argument enables Cadcorp SIS to connect to the database containing the data to be mapped. The methods you can use to connect to a database are:

- DAO (Microsoft Data Access Objects)
- ODBC (Open Database Connectivity)
- Oracle (direct driver)
- ADO (Microsoft ActiveX Data Objects)

The *`tables`* argument is a comma-separated list of database table names, corresponding to the *`columns`* argument.

The *`columns`* argument is a comma-separated list of database columns containing the data.

The *`aliases`* argument is a comma-separated list of names by which the data will be referred in Cadcorp SIS, ie the 'attribute name' of the data. These aliases must each be suffixed by $, &, or # to indicate its data type.

The *`sqlwhere`* argument is optional. This enables you to join tables or otherwise select a subset of data.

DAO

Using DAO the *`connect`* string should be DAO, followed by a semi-colon, followed by the pathname of the database:

```
"DAO;C:\Temp\MyDatabase.mdb"
```

ODBC

Using ODBC the *connect* string should be ODBC, followed by a semi-colon, followed by a combination of the following components:

```
DSN=DataSourceName
DATABASE=DatabasePathName
UID=UserName
PWD=Password
LOGINTIMEOUT=seconds
```

Components should be separated by semi-colons. For example:

```
ODBC;DSN=Fred;UID=MyName;PWD=topsecret
```

If the DSN exists, a connection will be established. The user name and password components are necessary only if the database to which the DSN refers requires them. If they are not required, you can omit these components altogether, or provide an empty string:

```
ODBC;DSN=Fred;UID=;PWD=
```

or

```
ODBC;DSN=Fred
```

If the DSN does not exist, you can create one on the fly by providing full connection details:

```
ODBC;DSN=Anything;DATABASE=C:\test.mdb;UID=MyName;PWD=topsecret
```

If insufficient information is provided, the standard Windows ODBC dialog will be displayed for the user to enter the required information.

Oracle

Using Oracle, the connect string should be Oracle, and the following components:

Server=*HostString*

User=*UserName*

Password=*Password*

Components should be separated by semi-colons. For example:

```
Oracle;Server=LPG;User=MyName;Password=topsecret
```

ADO

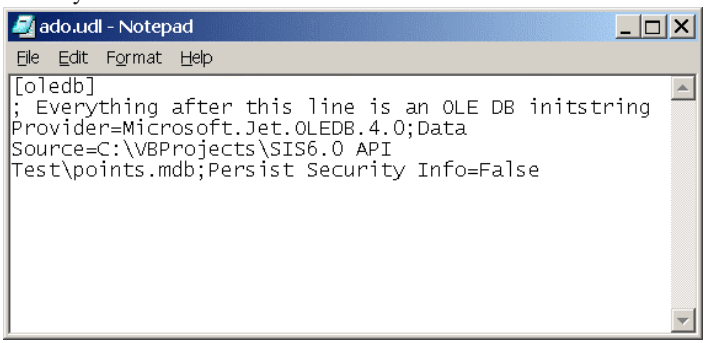
Using ADO, the *connect* strings hould be ADO, followed by a semi-colon, followed by the ADO Connection string:. For example:

```
ADO; Provider=Microsoft.Jet.OLEDB.4.0;Data Source=
C:\Temp\MyDatabase.mdb;Persist Security Info=False
```

A quick way to generate an ADO connection string is to create the connection using a Universal Data Link:

1. Create an empty text file, with any name, and change its file extension from *.txt to *.udl.
2. Double-click on this file in Explorer to display the Data Link Properties dialog.
3. Use this dialog to build the database connection, and click the Test Connection button.
4. If the connection succeeds, close the dialog.

5 Open this file using Notepad. The ADO connection string can now be copied and pasted into your code.



6 Delete the UDL file after use, because password information is also stored here.

For details on creating ADO connection strings, refer to your programming language's documentation.

◆ DefineRecordset: example 1

Here is an example of an Access database table, containing five columns:

Points : Table

ID	Easting	Northing	Data1	Data2
1	0	0	Absolute Zero	Sample1
2	1000	0	B	Sample2
3	2000	0	C	Sample3
4	3000	0	D	Sample4
5	4000	0	E	Sample5
6	0	1000	F	Sample6
7	0	2000	G	Sample7
8	0	3000	H	Sample8
9	0	4000	I	Sample9
11	4000	4000	TopCorner	Sample10
(AutoNumber)	0	0		

Record: 1 of 10

To create a View Points overlay displaying these records as points on the map, use `CreateDbPointOverlay` like this:

```
GisCreateDbPointOverlay 0, "*APrjNatGrid", "rsPoints", "Point", 1, 2, 0, -1,
-1, -1, 2000000
```

To create a View Points overlay displaying these records with one of the columns shown as text on the map, use the following code:

```
GisCreateDbPointOverlay 0, "*APrjNatGrid", "rsPoints", "Text", 1, 2, 0, -1,
-1, -1, 2000000
```

The arguments for this example are as follows:

`0`	the position in the overlays list at which to insert the overlay
`"*APrjNatGrid"`	the named projection of the stored point co-ordinates
`rsPoints`	the name of the recordset created using `DefineRecordset`
`"Point" or "Text"`	the column which is to be used as the Text is set within the `DefineRecordset` method
`1`	the index in the recordset *`columns`* argument of the x co-ordinate column
`2`	the index in the recordset *`columns`* argument of the y co-ordinate column
`0`	the index in the recordset *`columns`* argument of the item ID column. If the value is –1, Cadcorp SIS generates the item IDs automatically.
`-1`	the index in the recordset *`columns`* argument of the item spatial reference column. If the value is –1, no spatial reference is being supplied.
`-1`	the index in the recordset *`columns`* argument of the z co-ordinate column. A value of –1 indicates that points are 2D, and the Z values will be set to zero
`-1`	this parameter is currently ignored. For future compatibility, use –1.
`2000000`	the span used in the spatial reference

`rsPoints` is the name of the recordset created using `DefineRecordset`.

The `connect` string argument is shown below in three forms: using DAO, ODBC, and ADO:

DAO
```
sConnect = "DAO;C:\VBProjects\SIS6.0 API Test\points.mdb"
```

ODBC
```
sConnect = "ODBC;DSN=TestDSN"
```
created in Control Panel - Administrative Tools - ODBC (Data Sources)

or
```
sConnect = "ODBC;DSN=anything;DATABASE=C:\VBProjects\SIS6.0
API Test\points.mdb;UID=;PWD="'
```
(DSN created on the fly)

ADO
```
sConnect = "ADO;Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:\VBProjects\SIS6.0 API Test\points.mdb;Persist
Security Info=False"
```

all connections
```
SIS.DefineRecordset "rsPoints", sConnect,
"Points,Points,Points,Points,Points",
"ID,Easting,Northing,Data1,Data2",
"ID&,X#,Y#,_text$,Data2$", ""
```

The column named `Data1` is aliased as `_text$`, so the the `CreateDbPointOverlay` method creates a text item rather than a point.

◆ DefineRecordset: example 2

Here is an example of an Access database table, created by Cadcorp SIS Map Editor using the Add Overlay Wizard. The table contains five items:

Table1 : Table

ITEMID	ITEMVERSION	SPATIALREFERENCE	ITEMBLOB	TESTDATA
1	1	6d2d054cf5749eae	Long binary data	
2	2	6d2d054cf5700d4c	Long binary data	
3	3	6d2d054cf570350f	Long binary data	
4	4	6d2d054cf5748faf	Long binary data	
5	5	6d2d054cf5747e34	Long binary data	

Record: 3 of 5

The `DefineRecordset` method to create a recordset for use by `CreateDbBlobOverlay` is shown below:

```
GisDefineRecordset "rsBlobs", "DAO;" & dbPath,
"Table1,Table1,Table1,Table1,Table1",
"ITEMID,ITEMVERSION,SPATIALREFERENCE,ITEMBLOB,TESTDATA",
"ID&,Version&,SpatialReference$,Blob$,TestData$", ""
```

This example uses a DAO connection, but an ODBC or ADO connection could equally be used, as described above.

This recordset can now be used to create the overlay which displays this table:

```
GisCreateDbBlobOverlay 0, "*APrjNatGrid", "rsBlobs", SIS_BLOB_SIS, 3, 0, 1,
2, -1, -1, 2000000
```

This will create a View Blobs overlay at position 0 in the overlays list. When creating an overlay in this way, you must know the structure of the database table, so that you can set the correct arguments for the `DefineRecordset` and `CreateDbBlobOverlay` methods.

`SIS_BLOB_SIS`	a constant defined in SisConst.bas
`3`	the binary data (the graphics) are in column 3 of the table
`0`	the item ID is in column 0 of the table
`1`	the item version is in column 1 of the table

`2`	the spatial reference is in column 2 of the table
`-1`	(max and min scale thresholds cannot be set in the current release)
`2000000`	the spatial reference string encodes a position and a radius which together describe an extents circle The span used when calculating a spatial reference must be big enough to cover all of the possible co-ordinates. A smaller span will give spatial references with a finer resolution. The spatial reference does not affect the accuracy or resolution of the positions of the item it is associated with, only the accuracy of whether or not the item is loaded in a particular view. The worst that can happen with a coarse resolution is that extra items are loaded.

◆ Create an editable Blobs overlay

The above example shows how to add an overlay to view Blobs stored in an external database. The `CreateDbOverlay` method enables you to create an overlay to view and edit Blobs stored in an external database. To use this method, you must know the name of the database table storing the Blobs, the projection used, and the span of the spatial reference. A typical Blob table is shown below:

Blobs : Table

ITEMID	ITEMVERSION	SPATIALREFERENCE	ITEMBLOB
1	7	12cb2cb2cb29a515	Long binary data
2	3	09a69a69a153a774	Long binary data
3	12	6d34d34d34d5d040	Long binary data
4	17	09a69a69a6a22649	Long binary data

Record: 2 of 4

To add this as an editable Blobs overlay at position 0, use the following code. This example uses an ADO connection string:

```
sConnect = "ADO;Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:\VBProjects\SIS6.0 API Test\v6test.mdb;Persist Security
Info=False"

GisCreateDbOverlay 0, 0, False, sConnect, "Blobs", "*APrjNatGrid",
SIS_BLOB_SIS, 2000000
```

◆ Change the name of an overlay

You can change the name of an overlay within an SWD without affecting the filename of the underlying dataset. An overlay name is merely an alias, and is stored within the SWD file. If it is an internal overlay, the name *and* the data are stored within the SWD. Set the `_name$` property:

```
GisSetStr SIS_OT_OVERLAY, pos, "_name$", "Industrial Zones"
```

Substitute the index position of the overlay for *`pos`*.

◆ Set the scale thresholds of an overlay

Overlays can be set only to display within minimum and/or maximum viewing scales. These scale thresholds are stored within the SWD, and do not affect the display of the underlying dataset in any other SWDs.

Set the *_scalemin#* and *_scalemax#* properties:

```
GisSetFlt SIS_OT_OVERLAY, pos, "_scalemin#", 1000
GisSetFlt SIS_OT_OVERLAY, pos, "_scalemax#", 15000
```

◆ Find the filename of an overlay's dataset

If an overlay contains a Base Dataset (BDS) file, you must first obtain the *serial number* of the overlay's dataset. The serial number can be obtained from the `_nDataset&` property of an overlay, or from the `GetDataset`, `GetDatasetContainer` or `FindExternalDataset` methods. The serial number cannot be relied upon to be identical in each session, or if the dataset is removed and re-added. The number returned should therefore not be stored for long-term use.

Get the overlay's `_nDataset&` property, then get its dataset's `_name$` property:

```
lSerial = GisGetInt (SIS_OT_OVERLAY, pos, "_nDataset&")
sFilename = GisGetStr (SIS_OT_DATASET, lSerial, "_name$")
```

◆ Remove an overlay

Removing an internal overlay will delete its graphics, because internal overlays are stored within the SWD itself. Before removing an overlay containing an external (BDS) file, be sure to save the BDS first.

```
GisRemoveOverlay pos
```

◆ Replicate an overlay

To achieve the equivalent of the Cadcorp SIS **Edit>Replicate** command, you should first capture the items on the source overlay into a named list. You should then create and/or select the overlay which is to be the destination, and use the `CopyListItems` method. For more about named lists, ➲page 257, **Named lists**.

This code copies all items on Overlay 1 onto Overlay 2:

```
GisCreateListFromOverlay 1, "AllItems"
GisSetInt SIS_OT_WINDOW, 0, "_nDefaultOverlay&", 2
GisCopyListItems "AllItems"
```

◆ Move an overlay

Overlays can be moved up or down in the overlays list. The first overlays in the list are drawn on screen before later overlays. This may cause an overlay to obscure other data.

To move overlay 6 to the top of the list:

```
GisReorderOverlay 6, 0
```

Other overlays are shuffled down to accommodate this move.

◆ Zoom to the extent of an overlay

The Cadcorp SIS desktop interface allows users to select Zoom Overlay on the local menu in the workspace window. To provide the same functionality programmatically, you first need to retrieve the extent of all items in the overlay. This is returned as a comma-separated string of the X, Y, and Z values of the lower left and upper right of the enclosing rectangle. You should use the `SplitExtent` method to separate this into numerical values which can be passed to the `SetViewExtent` method:

```
GisCreateListFromOverlay pos, "AllItems"
SExtent = GisGetListExtent("AllItems")
GisSplitExtent X1, Y1, Z1, X2, Y2, Z2, sExtent
GisSetViewExtent X1, Y1, Z1, X2, Y2, Z2
```

For details of the `CreateListFromOverlay` and `GetListExtent` methods, ➲page 257, **Named lists**.

The above code will zoom to the extent of all visible items in the overlay, respecting any filter or locus which may be applied to the overlay. To zoom to the extent of all items in a dataset, whether they are visible or not, you should use the `GetDatasetExtent` method:

```
lSerial = GisGetInt (SIS_OT_OVERLAY, pos, "_nDataset&")
sExtent = GisGetDatasetExtent(lSerial)
GisSplitExtent X1, Y1, Z1, X2, Y2, Z2, sExtent
GisSetViewExtent X1, Y1, Z1, X2, Y2, Z2
```

◆ Find the projection of a dataset

The `GetDatasetProjection` method retrieves a dataset's projection and stores it in the current named object library under a definable name. This does not really tell you what the projection actually is.

Dataset projections can be retrieved in OpenGIS Well Known Text (WKT) format, which defines every parameter of a projection. This text string can then be examined to find the name of the projection.

A GB National Grid Backdrop dataset has the following WKT description of its projection:

```
PROJCS["OSGB 1936.British National Grid",
GEOGCS["Latitude/Longitude.OpenGIS.OSGB_1936",
DATUM["OSGB_1936",
SPHEROID["anon",6377563.396,299.324964600004]],
PRIMEM["Greenwich",0],
UNIT["degrees",0.0174532925199433]],
PROJECTION["Transverse_Mercator"],
```

```
PARAMETER["Central_Meridian",-2],
PARAMETER["False_Easting",400000],
PARAMETER["False_Northing",-100000],
PARAMETER["Latitude_of_Origin",49],
PARAMETER["Scale_Factor",0.999601272],
UNIT["m",1]]
```

The name of the projection can be used in methods which require a projection, such as:

`SetAxesPrj`, `SetDefaultPrj`, `SetDatasetPrj`, and `SetViewPrj`.

Use the Visual Basic `Split` function to extract the text from between the first pair of quotes:

```
SerialNo = GisGetInt (SIS_OT_OVERLAY, pos, "_nDataset&")
WKT = GisGetStr (SIS_OT_DATASET, SerialNo, "_projection$")
ProjArray = Split(WKT, Chr(34))
Proj = ProjArray(1)
```

This will return OSGB 1936.British National Grid in the above example.

The following statement will return Latitude/Longitude.OpenGIS.OSGB_1936, which can equally well be used in methods requiring a projection:

```
Proj = ProjArray(3)
```

◆ Find the dataset containing the selected item

Open the selected item:

```
GisOpenSel 0
SerialNo = GisGetDataset
```

◆ Find an overlay which contains a BDS dataset

Sometimes you will know the filename of a dataset, and want to find which overlay in the list contains this dataset. You will first need to obtain the serial number of the dataset. Then you use the `FindDatasetOverlay` method to search the overlays list:

```
SerialNo = GisFindExternalDataset ("C:\MyData\Rivers.bds")
Pos = GisFindDatasetOverlay (SerialNo, -1, True)
```

◆ Restore the overlay list settings

In some applications you will want to make a particular overlay current and editable, so the user can add data to it, then restore its status to its original state. You will also want to restore the user's current (default) overlay.

For example, your application requires the user to draw an area item on the Farms overlay, but previously the user has been drawing Lines on the Rivers overlay. The initial state of the overlays list might be as shown below:

Position	Name	Status	Current
0	GB National Grid	visible	No
1	LandLine	hittable	No
2	Farms	visible	No
3	Rivers	editable	Yes

Your application will set Farms to editable, and make it the default overlay. After the user has completed the task, your application should restore Rivers as the editable current overlay, and return Farms to its previous state.

The following code shows how to do this:

```
' Set up the variables to be used:
Dim lNumOverlays As Long
Dim lDefault As Long
Dim lCount As Long
Dim lStatus As Long
Dim lOverlays() As Long
Dim sName As String

' Get the number of overlays, and remember the default overlay:
lNumOverlays = GisGetInt(SIS_OT_WINDOW, 0, "_nOverlay&")
lDefault = GisGetInt(SIS_OT_WINDOW, 0, "_nDefaultOverlay&")

' Store the status of each overlay in an array:
ReDim lOverlays(lNumOverlays)
For lCount = 0 To lNumOverlays - 1
 lStatus = GisGetInt(SIS_OT_OVERLAY, lCount, "_status&")
 lOverlays(lCount) = lStatus
Next lCount

' If the overlay is "Farms" make it current and editable:
For lCount = 0 To lNumOverlays - 1
 sName = GisGetStr(SIS_OT_OVERLAY, lCount, "_name$")
 If sName = "Farms" Then
  GisSetInt SIS_OT_OVERLAY, lCount, "_status&", SIS_EDITABLE
  GisSetInt SIS_OT_WINDOW, 0, "_nDefaultOverlay&", lCount
  Exit For
 End If
Next lCount
```

After the user has added graphics and so on to the Farms overlay you will need to restore the settings:

```
For lCount = 0 To lNumOverlays - 1
 GisSetInt SIS_OT_OVERLAY, lCount, "_status&", lOverlays(lCount)
Next lCount
GisSetInt SIS_OT_WINDOW, 0, "_nDefaultOverlay&", lDefault
```

■ Object properties

Whether you are customising a Cadcorp SIS product (Map Manager, Map Editor, Map Modeller) using the GisLink interface, or creating a standalone GIS application using the Cadcorp SIS Control, you will need to set or retrieve the properties of graphical items. ➲page 235, **Text**

To set the font of the current text item:

```
GisSetStr SIS_OT_CURITEM, 0, "_font$", "Times New Roman"
```

To retrieve the bold setting of the current text item:

```
bBold = GisGetInt(SIS_OT_CURITEM, 0, "_text_bold&")
```

The `SetStr` and `GetInt` methods used in these examples can operate on Cadcorp SIS objects. The `SIS_OT_CURITEM` object represents the current item, ie the item which has been opened by your program using `OpenItem`, `OpenSel`, or `OpenList`, or has just been created using one of the graphical methods such as `CreateText` or `CreateRectangle`.

Cadcorp SIS provides fourteen objects which your programs can access, and they give you extensive control over the Cadcorp SIS environment and the components of the environment.

The Cadcorp SIS object types (`SIS_OT_`) are listed in the following table.

`SIS_OT_CURITEM`	the current open item
`SIS_OT_DEFITEM`	the default item. Whenever a new item is created by a Cadcorp SIS command it takes its default properties from those of the default item. The default item properties are not applied to items created using GisLink or Cadcorp SIS Control API methods.
`SIS_OT_DATASET`	Cadcorp SIS understands many different datasets, eg AutoCAD DXF or Ordnance Survey NTF, and each of these datasets has its own properties. A dataset is contained by an overlay, and is referenced by a serial number held by the overlay.
`SIS_OT_OVERLAY`	This provides access to the properties of each overlay in a window. An overlay is referenced by its list position in the window.
`SIS_OT_WINDOW`	If you are customising using GisLink, the window is the currently selected map window, 3D window, or table window. If you are using the Cadcorp SIS Control, the window is the control itself.
`SIS_OT_NOL`	Cadcorp SIS has a set of named object libraries which contain named object library classes, eg pen, brush, filter objects, and so on. These named object library classes are used throughout Cadcorp SIS.

`SIS_OT_FTABLE`	The properties of individual feature codes within a feature table can be set and queried using the `SIS_OT_FTABLE` constant.
`SIS_OT_SCHEMA`	Schema objects consist of a number of columns, each of which has a formula which is used to evaluate values on items. For example, when viewing an overlay in the table window, each row is equivalent to an item, and each column comes from the overlay schema.
`SIS_OT_SCHEMACOLUMN`	You can set and query the properties of individual columns within a schema using the `SIS_OT_SCHEMACOLUMN` constant.
`SIS_OT_THEME`	Cadcorp SIS uses named theme objects to control the display, eg brush, pen, shape, of items, depending on item properties, and also to annotate items, eg with bar charts, or pie charts.
`SIS_OT_THEMECOMPONENT`	Several types of theme consist of several components, eg blocks in a Bar Charts theme, slices in a Pie Charts theme, and so on. Each of these components has its own properties.
`SIS_OT_PRINTER`	Printer properties control printer settings used by the method `SendPrint`. You can set and query these using the `SIS_OT_PRINTER` constant.
`SIS_OT_SYSTEM`	Cadcorp SIS has several system variables, which are global to all windows: eg `_SnapTolerance&` is the tolerance used for snapping, measured in screen pixels, and `_ColSelection&` is the colour used for drawing selected items
`SIS_OT_OPTION`	Cadcorp SIS has several Boolean (True or False) system options, which are global to all windows: eg `_bFlickerDisplay&` sets whether the selected items should be flickered, and `_bShowMapTips&` sets whether MapTips should be shown when the cursor hovers over an item in a map window.

The API methods for accessing the properties of these object types fall into three categories.

Methods to read the value(s) of properties

`GetStr`	read a string (textual) property, eg `_pen$`
`GetInt`	read a (long) integer property, eg `_nDefaultOverlay&`
`GetFlt`	read a (double) floating point property, eg `_area#`
`GetStrW`	read the Unicode value of a string property
`GetListItemStr`	read the value of a string property on an item in a named list
`GetListItemInt`	read the value of a long integer property on an item in a named list
`GetListItemFlt`	read the value of a floating point property on an item in a named list
`GetPropertyDescription`	read the textual description of a property

Methods to set the value(s) of properties

If the property does not exist, it will be created.

`SetStr`	set the value of a string (textual) property
`SetInt`	set the value of a (long) integer property
`SetFlt`	set the value of a (double) floating point property
`SetStrW`	set the Unicode value of a string property
`SetListStr`	set the value of a string property of several items at once
`SetListInt`	set the value of an integer property of several items at once
`SetListFlt`	set the value of a floating point property of several items at once
`DescribeProperty`	set the textual description of a property

Methods to remove properties

`RemoveAtt`	removes a user-defined attribute from the current open item.
`RemoveProperty`	removes a user-defined property from an object.

You can retrieve a list of properties held on any of the fourteen `SIS_OT_` objects by querying one of four special properties:

`_properties$`	contains a space-separated list of all the properties of an object

`_properties_edit$`	contains a space-separated list of all the editable properties of an object
`_members$`	contains a space-separated list of all the system properties of an object. These are the properties which are prefixed with the underscore (_) character.
`_attributes$`	contains a space-separated list of the user-defined properties of an object

◆ SIS_OT_CURITEM

See ➲page 235, **Text**, for this example of retrieving the area of the current item:

```
dArea = GisGetFlt(SIS_OT_CURITEM, 0, "_area#")
```

If you are writing for the Cadcorp SIS Control the code should read:

```
dArea = Sis.GetFlt(SIS_OT_CURITEM, 0, "_area#")
```

The three methods for retrieving property values (`GetStr`, `GetInt`, and `GetFlt`) each require three arguments:

`objectType`	integer	the object type holding the property, in this case `SIS_OT_CURITEM`
`nObject`	long integer	the object number. Because there is only ever one current item, this value is not used and should be set to zero.
`propertyName`	string	the property name

System properties, ie properties which Cadcorp SIS automatically manages, are always prefixed with an underscore (_) character. User-defined properties, known as attributes, must not be prefixed with an underscore. This argument is a string, and should be enclosed in quotes. All properties, system and user-defined, must be suffixed with a $, &, or # character, which defines the data type of the property.

Because this example uses the `GetFlt` method, the property must be a floating point property, with a hash (#) suffix, and the return value must be assigned to a double variable or used as a double in an expression.

Using another example from the section about working with text (➲page 235, **Text**), the following line sets the horizontal alignment (justification) of the current text item:

```
GisSetInt SIS_OT_CURITEM, 0, "_text_alignH&", SIS_LEFT
```

`SIS_LEFT` is a pre-defined constant declared in the GisLink.bas module or the SisConst.bas module of your application. `SIS_LEFT` and its colleagues are defined as follows:

```
' Horizontal Text alignment
Global Const SIS_LEFT = 0
Global Const SIS_RIGHT = 2
Global Const SIS_CENTRE = 6
```

For details of the GisLink module, ➲Chapter 2: **"Customising with GisLink"**.

All properties of the current item can be read, but only certain properties can be altered. For example, it is possible to change the text-related settings of a text item, eg `_font$`, `_text_alignH&`, but it is not possible to change the `_area#` value of an area item, because this is automatically calculated from the item's geometry.

To add your own properties, known as attributes, to the current item, set the property. If the property does not exist, it will be created:

```
GisSetInt SIS_OT_CURITEM, 0, "MaxSpeedLimit&", 50
```

This will add or change the attribute `MaxSpeedLimit&` to 50.

To add a textual description to this attribute, use the `DescribeProperty` method:

```
GisDescribeProperty "MaxSpeedLimit$", "Maximum Speed"
```

An item can be opened (made current) irrespective of the status of its overlay. You are able to alter property values on items which reside in editable, hittable, visible or invisible overlays.

◆ SIS_OT_DEFITEM

The default item is an abstract item which has properties such as pen, brush, and so on. When a new item such as a line or area is created, using any of the Cadcorp SIS commands, the new item inherits the properties of the default item. Setting the pen, brush, and shape properties of the default item is analogous to the user setting the pen, brush, and shape using the Style toolbar in the Cadcorp SIS user interface.

Items created using `LineTo`, `CreateRectangle`, `CreateText`, and so on do not take on the properties of the default item.

Refer to the relevant sections of this manual for examples and explanations of the properties of `SIS_OT_` objects.

■ Named lists

Named lists are used in both GisLink and Cadcorp SIS Control applications as a means of passing and retrieving a list of items to and from many API methods. Named lists are referred to by a textual name provided by the programmer. For example, to build a list of all the currently selected items, use the `CreateListFromSelection` method:

```
GisCreateListFromSelection "SelectedItems"
```

You can then refer to this list by its name, and operate on it using many API methods.

Named lists persist throughout a Cadcorp SIS or Cadcorp SIS Control session, but can become out of date during a session if items are deleted or datasets removed. As a general rule, you should not rely upon named lists as anything other than temporary storage, and it is good practice to remove the list when you have finished:

```
GisEmptyList "SelectedItems"
```

The `EmptyList` method removes items from the list, and deletes the list. The items themselves are not removed. To delete all items in a list you can use the `Delete` method:

```
GisDelete "SelectedItems"
```

To find the number of items in a list, you can use the `GetListSize` method, although many methods which return a named list also return the number of items found:

```
lNumItems = GisGetListSize ("SelectedItems")
lNumFound = GisScan ("Farms", "V", "FarmFilter", "CountyLocus")
```

All the spatial searching methods (`Scan…`) return a named list of the items found by the search. This includes the ability to spatially search a named list, retrieving the result into the same list or a new one.

Named lists can also be added to each other, subtracted from each other, extended or reduced, even spatially moved as a group.

For more information on the various scanning methods, ➲page 275, **Spatial searches**

◆ Add attributes to items in a list

Your application may need to add an attribute to several items. By putting these items in a named list there are methods for adding attributes in bulk, and for querying attributes of items within the list. For example, your application may need to add a Consulted attribute to a group of address points on the map, to indicate that these addresses have been sent a consultation letter with regard to a planning application. The user selects the address points using normal Cadcorp SIS selection methods. Your code can then create a named list of the selected items and add the attribute to all items in the list:

```
GisCreateListFromSelection "SelectedItems"
GisSetListStr "SelectedItems", "Consulted$", "YES"
```

If the user has made the selection using a selection fence, the list may contain many items which are not address points. Assuming the address points hold a Postcode attribute, you could create a filter in order to ensure the list excludes all items which are not addresses. You can then modify the list using this filter:

```
GisCreateListFromSelection "SelectedItems"
GisCreatePropertyFilter "Postcodes", "Exists(""Postcode$"")"
GisScanList "Addresses", "SelectedItems", "Postcodes", ""
```

Notice the use of double quotes (`""`) within the filter formula. For more information on creating and using filters, ➲page 261, **Filters**.

The `ScanList` method enables you to build a new list based upon an existing one. In this example, the Addresses list is built by scanning the SelectedItems list. If you no longer require the original list, you can scan the list into itself, for example:

```
GisScanList " SomeItems", "SomeItems", "Postcodes", ""
```

This will reduce the SomeItems list to contain only the items which pass the filter. This technique is similar to the following Visual Basic statement, where a variable value is modified by a formula containing the variable itself:

```
MyNumber = MyNumber - 5
```

Another technique for adding attributes to items in a list is to cycle through each item in the list, open each in turn and adding attributes one by one:

```
GisCreateListFromSelection "SelectedItems"
lNumItems = GisGetListSize("SelectedItems")
For lCount = 0 To LNumItems - 1
 GisOpenList "SelectedItems", lCount
 GisSetStr SIS_OT_CURITEM, 0, "Example$", "SomeData"
Next lCount
```

The first method is much simpler if all items are to receive the same attribute value.

◆ Retrieve attribute values from items in a list

A similar technique can be used for retrieving attribute values, ie to step through each item in the list. The API provides methods which return an attribute's value from an item in a list:

```
sValue = GisGetListItemStr("SelectedItems", lCount, "Postcode$")
lValue = GisGetListItemInt("SelectedItems", lCount, "Bedrooms&")
dValue = GisGetListItemFlt("SelectedItems", lCount, "_area#")
```

◆ Combine lists

Often you will want to combine two lists to build a resultant list. The `CombineLists` method enables you to use Boolean operations on two lists.

For example, if you have used one of the `Scan…` methods to retrieve into a named list all areas which are designated as private property. Another `Scan…` has produced a list of all areas which are woodland. A third `Scan…` has produced a list of all areas where Spotted Owls are know to be nesting.

These three lists (Private, Woods, and Owls) can be combined in several ways:

Create a list of all areas

```
GisCombineLists "Areas", "Private", "Woods", SIS_BOOLEAN_OR
```

Notice the use of Boolean OR to collect items which are in either list into a grand total.

Create a list of areas which are privately owned woodland

```
GisCombineLists "PrivWoods", "Private", "Woods", SIS_BOOLEAN_AND
```

The AND requires that each item must appear in both lists.

Create a list of areas which are either woodland or private, but excluding private woodland

```
GisCombineLists "WoodOrPri", "Private", "Woods", SIS_BOOLEAN_XOR
```

This is known as exclusive or.

Create a list of woodland where Spotted Owls do not nest:

```
GisCombineLists "NoOwls", "Woods", "Owls", SIS_BOOLEAN_DIFF
```

This creates a list containing items which are in Woods but not in Owls.

◆ Move items using a list

You can use a named list to move one or more items. For example, you may be writing an application which displays the location of vehicles in a town. Each vehicle is represented by a symbol (a point item with a shape attribute) and a text label.

Assuming the symbol and the label have a reference number attribute, you can add each vehicle and its label to a list, and then move the list to its new location.

When you use the `MoveList` method, notice that the co-ordinates for the move are relative to the existing position of the items. If the new location is being read from a resource file, you must compare the new position with the current position to calculate the relative move. For simplicity, the following example assumes the calculation has been made, and that the move is to be 10 metres north, 5 metres east.

```
' Find the required vehicle graphics and label:
GisCreatePropertyFilter "Ref", "RefNo&=2346"
GisScan "Vehicle", "E", "Ref", ""
GisMoveList "Vehicles", 5, 10, 0, 1, 0
```

The last two arguments of the method enable a scale and/or rotate of the graphics at the same time as the move.

◆ Zoom to a list of items

When you have collected items into a list, perhaps as the result of a spatial search, you may want to zoom the map to display all of the items. There are two ways to achieve this, the choice depending on whether you are able to select the items in the list.

If the items in the list are on a hittable or editable overlay, and it is not important to retain the user's current selection, select all items on the list, then call the Cadcorp SIS command AComZoomSelect:

```
GisDeselectAll
GisSelectList "SomeItems"
GisCallCommand "AComZoomSelect"
```

If the items are on a visible overlay but are not hittable or editable, you must get the extent of items in the list first, then zoom to this extent. The next example uses the `GetListExtent` method to retrieve the extent of the items, then `SplitExtent` to convert the result into X, Y, and Z values:

```
sExtent = GisLetListExtent("SomeItems")
GisSplitExtent X1, Y1, Z1, X2, Y2, Z2, sExtent
GisSetViewExtent X1, Y1, Z1, X2, Y2, Z2
```

◆ Highlight items in a list

When your program zooms to display the items, you may want to highlight them for the user's attention. To do this, use the `DrawList` method, which enables you to draw all items on the list with a bold appearance. The changed appearance lasts only until the view is refreshed, when the items return to their original appearance. You can set the pen, brush, shape, and font of the items. The following example displays lines in a 1mm thick red pen, fill areas in a light red colour, displays all point items as a solid circle, and displays text in the Arial Black font:

```
GisDrawList SIS_CURRENTWINDOW, "SomeItems", "P_SOLID_255:0:0_200R_0",
"B_SO_255:128:128_255:255:255", "Circle", "Arial Black"
```

The brush style should be defined as solid, so that the temporary shade obliterates the underlying shade. Transparent brush styles will result in rather murky and unexpected colours.

■ Filters

Spatial searching and manipulating named lists are made considerably more powerful using filters. A filter is a non-spatial criterion or set of criteria which must be met for an item to be added to a list. In another section (➲page 257, **Named lists**) we give an example of a simple scan of the map to return a list of items considered to be Farms:

```
lNumFound = GisScan ("Farms", "V", "FarmFilter", "CountyLocus")
```

This scan operation collects into a list all visible items which pass the FarmFilter, provided they are within the zone named CountyLocus. For more detail on using a locus within the spatial searching methods, ➲page 275, **Spatial searches**.

Filters are created and given a textual name, and can be reused throughout the time your application is running. They can then be used, as above, as a parameter of a spatial search, or to restrict the display of items on an overlay to only those items which pass the filter. Filters can also be combined to form more complex criteria than could easily be defined in a single filter.

The Cadcorp SIS API provides five different methods for defining filters, and additional methods to combine or add to existing filters. The five different types of filter are listed below:

Filter type	What the filter will pass
Class Tree	only items that belong to specified item classes
Feature	only items that have specified feature codes
Link	only items with one of a specified list of item IDs
Property	only items with a particular property value
Value List	only items that have a specified value for an integer property

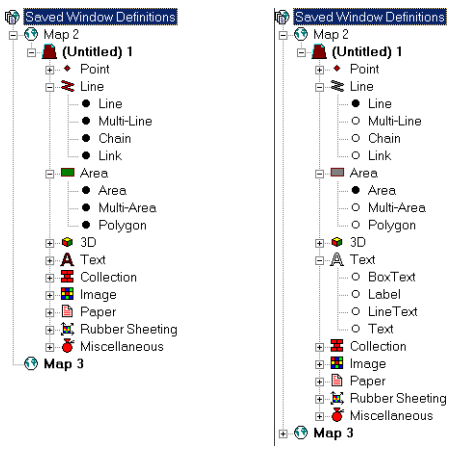
◆ GisLink and Cadcorp SIS Control

There are no significant differences in the way you would use these methods in a GisLink customisation or a Cadcorp SIS Control application. The examples in this section use the GisLink prefix to each method, and include the GisRelease method. If you are programming for the Cadcorp SIS Control, you don't need the prefix or GisRelease.

◆ Class Tree filters

Users of the Cadcorp SIS desktop applications will be familiar with the class tree, as displayed in the workspace window. All graphics and text belong to the category of item.

Items are then separated into classes, such as points, lines, and areas. In turn, classes can have sub-classes. These classes can be switched on or off to define a filter. Items will pass the filter only if the class to which they belong is switched on:



Filters can be created by constructing a string which defines the classes to included or excluded. For example, to allow all items except areas, the string would be:

```
+Item -Area
```

To disallow all items but allow areas, the string would be:

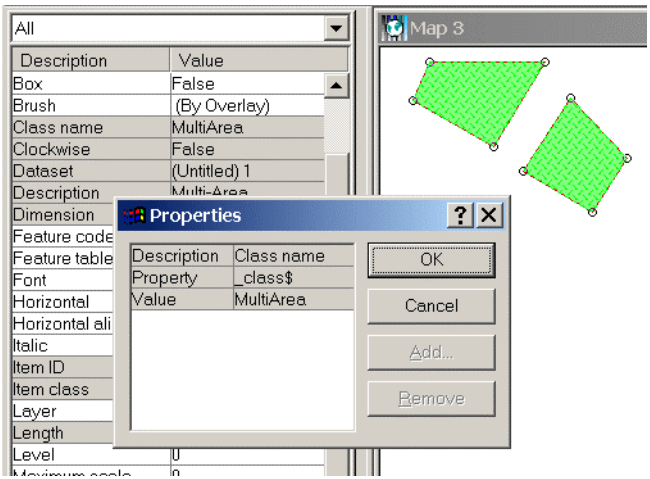
```
-Item +Area
```

By prefixing the class name with a plus sign (+) or a minus sign (–), you can specify the full class hierarchy.

Using the `CreateClassTreeFilter` method and GisLink, you might write code as follows to scan for area items:

```
GisCreateClassTreeFilter "Areas", "-Item +Area"
```

However, this code would not allow multi-area items through the filter. A multi-area is several area items grouped together by using the Boolean Union command. It is considered by Cadcorp SIS to be a single entity in every way, but consists of several discrete loops. Multi-areas have the class name MultiArea (note the absence of a hyphen). You can always check the actual class name of an item in the Properties tab of the workspace window in Cadcorp SIS:




```
GisCreateClassTreeFilter "Areas", "-Item +Area +MultiArea"
```

◆ Feature filters

If a feature table is assigned to an item, the item can have one of that table's feature codes assigned to it, which automatically controls the display properties of the item. An example of feature-coded data is the GB Ordnance Survey LandLine data. Using a feature filter you are able to include or exclude items which belong to specified feature groups. Feature filters differ from class tree filters in that an empty filter is first created, then clauses are added to it one by one.

Below is example code which creates a feature filter. This filter passes only those items which could possibly form part of the boundary of agricultural land, ie it excludes overhead power lines, dotted lines of underpasses, and so on:

```
GisCreateFeatureFilter "PermittedLandLine", "Land-Line"
GisChangeFeatureFilter "PermittedLandLine", 0, SIS_FEATUREEXCLUDE _
  + SIS_FEATURECASCADE
GisChangeFeatureFilter "PermittedLandLine", 1, SIS_FEATUREINCLUDE
GisChangeFeatureFilter "PermittedLandLine", 4, SIS_FEATUREINCLUDE
GisChangeFeatureFilter "PermittedLandLine", 21, SIS_FEATUREINCLUDE
```

and so on for codes 30, 32, 35, 36, 52, 59, 71, 72, 374, 375, and 376.

◆ Link filters

To create a link filter, provide a space-separated list of item IDs. For example:

```
GisCreateLinkFilter "OddNumbers", "1 3 5 7 9"
```

Take care when using a Link Filter, because if there is more than one overlay, there may well be more than one item with the same ID, because IDs are unique only within their own dataset. You should therefore use a link filter only when using the `ScanOverlay` method, where the overlay is specified, or where you are certain this ambiguity is avoided.

An example of using a Link Filter is where your application progressively builds a list of items selected by the user. When the list is complete, you then want to perform an operation on all items in bulk.

◆ Property filters

Property filters are probably the most frequently used filters in GisLink or Cadcorp SIS Control applications. A property filter requires a formula as an argument. See the Cadcorp SIS on-line help for full details of the types of formula you can create.

A typical formula consists of a property name, a comparison operator, and a value. For example, the formula used in a property filter to pass only items that have an area greater than 50 square metres would be:

```
"_area# > 50"
```

Any item property can be used in a formula, including user-defined attributes:

```
"SalesTurnover# >= 5000.00"
```

Where the formula uses a numerical property (integer or float), the value on the right hand side of the operator is simply a number, with or without a decimal point. If the property is a string, Cadcorp SIS requires the value to be enclosed in double quotes. Programmers familiar with Visual Basic will know that when a string is constructed which requires double quotes, the double quote character (ASCII 34) needs to be entered twice, so that Visual Basic knows it should be used literally rather than as a means of enclosing the entire string:

```
"CropType$ = ""Wheat"""
```

Visual Basic will store the above string as:

```
CropType$ = "Wheat"
```

When you include this formula into the `CreatePropertyFilter` method, the code is:

```
GisCreatePropertyFilter "MyFilter", "CropType$ = ""Wheat"""
```

A frequent requirement is to select or filter only items which have a particular attribute assigned to them. For example, you may wish to filter for all items which have the `CropType$` attribute, irrespective of what the crop may be. You can achieve this using the `Exists` formula expression. `Exists` will return True (–1) if an item has that attribute, or False (0) if not:

```
GisCreatePropertyFilter "Crops", "Exists(""CropType$"")"
```

Notice the use of double quotes within the formula.

Because a property filter can use almost all the functionality of Cadcorp SIS formulae, it is possible to build filters of considerable complexity. To define a property filter which filters all items over 50 square metres, and which have a CropType which is not blank and is not Cabbages:

```
sFormula = "CropType$ <> """" And CropType$ <> ""Cabbages"" And _area# > 50
GisCreatePropertyFilter "NotCabbages", sFormula
```

◆ Value list filters

A value list filter is for use specifically with integer properties. Like the feature filter, you must create the filter first, then add or remove lists of permissible values. The link filter example above creates a filter passing only items whose `_id&` values are 1, 3, 5, 7, or 9. A value list filter could be created and numbers added to it or removed from it to produce a filter with the same effect:

```
GisCreateValueListFilter "OddNumbers", "_id&"
GisChangeValueListFilter "OddNumbers", SIS_FILTERADD, _
  "1,2,3,4,5,6,7,8,9,10"
GisChangeValueListFilter " OddNumbers ", SIS_FILTERREMOVE, "2,4,6,8,10"
```

Note that a link filter requires a space-separated list of ID numbers, whereas a value list filter will accept a list separated by spaces, commas, tabs, or newlines, and will operate on any integer properties:

```
GisCreateValueListFilter "SmallHouses", "NumBedrooms&"
GisCreateValueListFilter "BigHouses", "NumBedrooms&"
GisChangeValueListFilter "SmallHouses", SIS_FILTERADD, "1,2,3"
GisChangeValueListFilter "BigHouses", SIS_FILTERADD, "4,5,6,7"
[GisCreatePropertyFilter "Mansions", "NumBedrooms& > 7"]
```

◆ Where do filters go?

When a filter is created, it is stored in a named object library (NOL). By default this library is the temporary library available in all Cadcorp SIS sessions. At the end of the session, whether this is a Cadcorp SIS desktop session (using Map Manager, for example) or a Cadcorp SIS Control application session, the temporary library is automatically deleted and any filters or other data in the library are lost. If a file-based NOL is made current, filters created by your application will be stored in this, and if the library is saved at the end of the session, any named filters you have created will be retained for future use.

Cadcorp SIS desktop applications, customisable using GisLink, have the concept of a workspace which stores system settings, and can also act as a library. It is therefore possible that your application will add several filters to the user's named object library or workspace. Filters can be permanently deleted using the `DeleteNolObject` method. For details on managing NOLs and their content, ➲page 270, **Named object libraries**.

◆ Combine filters

It is sometimes too difficult or impossible to define a filter by using a single filter method. The API provides the `CombineFilter` method which combines the action of two filters into one. You can specify the way in which the two filters are combined, in the same way that the `CombineLists` method enables you to combine named lists (➲page 257, **Named lists**). A compound filter can be created from two filters which themselves were created using any of the filter methods described above. The resultant filter can in turn be combined with another filter.

There are significant differences between combining filters and combining lists. When combining lists, a new list is created containing items from the two constituent lists. When combining filters, no items are collected into a list. The new filter can be used and re-used to collect items into a list, and can be stored in a library for re-use over a long period of time.

The example of combining named lists given in another section (➲page 257, **Named lists**) shows how three lists, Private, Woods, and Owls, can be combined in several ways to produce subsidiary lists of items. The same resultant lists could be created without the need for intermediate lists by using combined filters.

The Private named list of items might be created using a Property Filter with the `ScanOverlay` method:

```
GisCreatePropertyFilter "Private", "Owner$ = ""Private"""
lNumFound = GisScanOverlay("Private", 1, "Private", "")
```

The Woods named list might be created using a similar method:

```
GisCreatePropertyFilter "Woods", "Category$ = ""Woods"""
lNumFound = GisScanOverlay("Woods", 1, " Woods", "")
```

The Combining Lists example uses the Boolean AND operation to create a list of Private Woodland.

The same list could be created by combining the filters, then performing a single scan. Reducing the number of scans will noticeably speed up your application:

```
GisCreatePropertyFilter "Private", "Owner$ = ""Private"""
GisCreatePropertyFilter "Woods", "Category$ = ""Woods"""
GisCombineFilter "PrivWoods", "Private", "Woods",SIS_BOOLEAN_AND
lNumFound = GisScanOverlay("Private", 1, " PrivWoods", "")
```

To create a list of woodland where owls' nests have been observed:

```
GisCreatePropertyFilter "Woods", "Category$ = ""Woods"""
GisCreatePropertyFilter "Owls", "Exists(""OwlNests&"")"
```

```
GisCombineFilter "OwlWoods", "Owls", "Woods", SIS_BOOLEAN_AND
lNumFound = GisScanOverlay("OwlWoods", 1, " OwlWoods", "")
```

To reverse the effect of the Owls filter, use the Boolean DIFFERENCE operation between an empty filter and the Owls filter:

```
GisCombineFilter "NoOwls", "Owls", "", SIS_BOOLEAN_DIFF
lNumFound = GisScanOverlay("OwlFree", 1, " NoOwls", "")
```

These compound filters do not store any items, only the criteria for selecting them, so they can be used and re-used as the data on your map changes.

■ Groups

A group is an entity which can be created only using GisLink or the Cadcorp SIS Control. Users of the Cadcorp SIS desktop products (Map Manager, for example) are not able to create groups, but can explode a group into its constituent items using the local command Explode.

If you do not want users to be able to explode a group, you should exclude the command from the user interface:

```
GisAllowCommands SIS_COM_REMOVE, "AComExplodeGroup"
```

A group is a collection of items which are manipulated as a single entity by the user. Groups cannot be edited, other than to move, stretch, rotate, or delete them, or to add attributes or set style properties. If you want to prevent users from stretching or rotating groups, you should remove the AComStretch and AComRotate commands.

Groups offer three main benefits:

- multiple items are treated as a single entity, and their component items cannot be altered
- a group can be attached to the user's cursor for placement
- groups can be sub-classed to give them their own class name

◆ Build a group of existing items

The `CreateGroupFromItems` method enables you to create a group of all items in a named list (➲page 257, **Named lists**), with the option to delete the original constituent items. Normally you would provide a name for the group class, but it is possible to create a group without a class name:

```
GisCreateListFromSelection "Selected"
GisCreateGroupFromItems "Selected", True, "MyGroup"
```

or

```
GisCreateGroupFromItems "Selected", True, ""
```

In the above example, the selected items are grouped as a single entity, and the original items are deleted. The user can now manipulate the group, or explode the group back to its constituents. The constituent items retain any properties and attribute data they held, and these properties are restored if the group is exploded.

◆ Build a group from scratch

All the API methods which create graphics or text normally create items which are individually editable by the user. If a group is created first, new items will become part of the group, and will remain grouped until the user chooses to explode the group:

```
GisRegisterGroupType "TestGroup"
GisCreateGroup "TestGroup"
GisMoveTo 0, 0, 0
GisLineTo 10, 0, 0
GisLineTo 10, 10, 0
GisLineTo 0, 10, 0
GisLineTo 0, 0, 0
GisStoreAsArea
GisRelease
```

This example creates a 10 metre square area item. The square will be attached to the user's cursor, requiring a position to locate the corner of the square, and a second position to orient it. If the user presses the Enter key instead of giving a second position, the square is oriented along the x-axis.

The co-ordinates are all relative to the tip of the user's cursor.

Although this group contains only a single item, it remains a group, and as such the user will be unable to perform the editing functions associated with an area item.

The following code creates two concentric squares, with the word SQUARE placed in the centre. All three items are built as a group, and can be manipulated only as a whole:

```
GisRegisterGroupType "ColouredSquares"
GisCreateGroup "ColouredSquares"
GisMoveTo -5, -5, 0
GisLineTo 5, -5, 0
GisLineTo 5, 5, 0
GisLineTo -5, 5, 0
GisLineTo -5, -5, 0
GisStoreAsLine
GisSetStr SIS_OT_CURITEM, 0, "_pen$", "Red"
GisSetStr SIS_OT_CURITEM, 0, "WhatAmI$", "Small Red Square"
GisMoveTo -10, -10, 0
GisLineTo 10, -10, 0
GisLineTo 10, 10, 0
GisLineTo -10, 10, 0
GisLineTo -10, -10, 0
GisStoreAsLine
GisSetStr SIS_OT_CURITEM, 0, "_pen$", "Blue"
GisCreateBoxText 0, 0, 0, 1, "SQUARES"
GisSetInt SIS_OT_CURITEM, 0, "_text_alignH&", SIS_CENTRE
GisSetInt SIS_OT_CURITEM, 0, "_text_alignV&", SIS_MIDDLE
GisSetStr SIS_OT_CURITEM, 0, "_pen$", "Green"
GisRelease
```

As each component is created, you can set its display properties. Any attributes added to a component of a group (`WhatAmI` in this example) will not be accessible to the user or to any querying methods until the group is exploded.

To add attributes to the group itself, you must monitor the AComPlaceGroup command. This is the command that occurs automatically when the user places the group. When the second (orientation) snap is given, or when the user presses the Enter key, the End trigger is fired. If you are using GisLink you should register the End trigger of the AComPlaceGroup command. If you are writing an application using the Cadcorp SIS Control, you will need to monitor the End action of the AComPlaceGroup command in the CommandAction event of the Cadcorp SIS Control.

For information about Cadcorp SIS triggers and events, ➲Chapter 2:**"Customising with GisLink", GisLink Triggers** and ➲page 25, **Cadcorp SIS Control events**.

◆ Add attributes to the group: GisLink

1. Add a button to your main form and set its caption to PlaceGroupEnd.
2. Register a trigger in the ListCaps button's Click event:

```
GisRegisterTrigger "AComPlaceGroup::End", "PlaceGroupEnd"
```

3. Add code to the trigger button's Click event:

```
GisOpenSel 0
GisSetStr SIS_OT_CURITEM, 0, "GroupAttr$", "ExampleText"
GisUpdateItem
GisRelease
```

◆ Add attributes to the group: Cadcorp SIS Control

Add this code to the CommandAction event of the Cadcorp SIS Control:

```
If comname = "AComPlaceGroup" And action = "End" then
 Sis.OpenSel 0
 Sis.SetStr SIS_OT_CURITEM, 0, "GroupAttr$", "ExampleText"
 Sis.UpdateItem
End If
```

◆ Explode a group on placement

Groups are often used as a way of putting graphics on the cursor for the user to place. Once the items are placed you then want to explode the group leaving ordinary Cadcorp SIS items on the map. To do this, give an empty string as the argument for the `CreateGroup` method. There is therefore no need to register the group first:

```
GisCreateGroup ""
GisMoveTo -5, -5, 0
GisLineTo 5, -5, 0
GisLineTo 5, 5, 0
GisLineTo -5, 5, 0
GisLineTo -5, -5, 0
GisStoreAsLine
GisRelease
```

■ Named object libraries

Cadcorp SIS uses named object libraries (NOLs) to store a range of named object classes. The Cadcorp SIS API provides a number of methods for creating, adding and manipulating named object libraries.

Below is a list of the named object classes, showing the API method(s) used to create each object:

Class	Purpose	Method(s)
`Block`	A block object is a collection of graphics and text that can be manipulated as though they were a single entity. Think of it as a component that may be called up by name and placed onto the map.	`CreateBlock`
`Brush`	A brush has a colour and a fill type, eg Solid, Hatched, etc, and defines the way in which filled item interiors, area and polygon item interiors, opaque text items, etc, are drawn.	`DefineNolObject`
`Colour-set`	A colour-set is a series of values with associated colours, and is used by grid items to indicate heights, densities or other values by graduated colour shades.	`DefineNolObject`
`Feature Table`	Cadcorp SIS uses named feature table objects as a fast and efficient way of controlling the display (brush, pen, shape, for example) of items.	`LoadFeatureTable` `StoreFeatureTable`

`Filter`	A filter is an object which has rules which it uses to pass some items, but fail others.	`CreateClassTreeFilter`
		`CreateFeatureFilter`
		`CreatePropertyFilter`
		`CreateLinkFilter`
		`CreateValueListFilter`
		`CreateCombinedFilter`
		`CombineFilter`

`Geoid Datum`	A Geoid Datum is a set of parameters defining co-ordinate systems for local parts of the Earth or for all of the Earth. Different datums have been produced and revised over time. They are used to produce a better local fit of a spheroid to the actual shape of the Earth (the geoid).	`DefineNolDatum`
`Graticule Style`	A graticule style stores all of the styling and setup of a graticule item.	no API methods

Class	Purpose	Method(s)
`Item`	A named item can be stored in a named object library. The named item stores an item, and a projection which can be used to place the Item in the world. Items are frequently used to store map backdrops.	`DefineNolItem`

`Locus`	A locus is a named object which includes (passes) or excludes (fails) items based on their position and geometry. A locus object is the spatial equivalent of an filter object.
	Locus objects use geometry tests to decide whether or not an item passes or fails the locus.
		`CreateLocusFromItem`
		`CreateBufferLocusFromItems`
		`CreateCircleLocus`
		`CreateRectLocus`

`Pen`	A pen has a thickness, a colour and a style, such as Solid or DashDot, and defines the way in which Item line geometry such as line items and area item boundaries are drawn.	`DefineNolObject`
`Print Template`	A print template is a pre-defined page layout, which, as well as the view of the map, can also include other 'furniture', such as page borders, logos, scale bar, north point, and graticule items. The view of the map is defined using a photo item.	`DefineNolPrintTemplate`
`Projection`	A projection is a method of projecting positions in the world onto screen or paper. The surface of the Earth is curved, but paper is flat. These two facts mean that paper maps will always be slightly distorted. Different projections are designed to minimise the distortion in different ways. Some projections show orientation accurately, some show areas accurately, and so on.	`DefineNolObject` `DefineNolPrjLatLon`
`Schema`	A schema object controls the display of data-oriented parts of the user interface, eg the table window.	`StoreSchema`

Class	Purpose	Method(s)
`Shape`	A shape is a pre-defined symbol that may be assigned as an attribute of any point item. Then, whenever the point item is drawn you see a copy of the shape. The point item can optionally store information that scales and rotates the shape.	`DefineNolShape`
`Theme`	A theme controls the display, such as brush, pen, and shape, of items depending on item properties, and also annotates items, eg with bar charts or pie charts.	`StoreTheme`
`Toolbar Definition`	A toolbar definition defines the contents of a toolbar.	no API methods
`View`	A view stores the extents, scale and projection of map window.	`DefineNolView`

◆ Add a NOL

By default, every Cadcorp SIS session, whether running a desktop product such as Map Editor or an application which uses the Cadcorp SIS Control, makes a standard and temporary NOL available. The standard NOL is read-only, and is in fact an amalgamation of all NOLs in the Libraries folder within the Cadcorp SIS application folder. The temporary NOL is read/write and is initially empty. Any named objects created during a session will be stored in the temporary NOL, unless another NOL is nominated as current. The temporary NOL is not saved when the session is ended. The Cadcorp SIS desktop applications provide the ability to cut, copy, and paste items between NOLs.

When a user wishes to create and store any of the above named objects, they would normally use a NOL file to which they have write access. If you are customising Cadcorp SIS desktop products using GisLink, or writing applications using the Cadcorp SIS Control, you can use the methods for creating, opening, closing, and saving NOL files.

◆ Create and insert NOLs

The `NolCreate` method creates a new, empty NOL file, and requires a full pathname to the folder in which the file is to be created:

```
GisNolCreate "C:\MyLibraries\Example.nol"
```

To attach this NOL to the current session, you must insert it, specifying the file, the position in the session's list of NOLs, and a flag to indicate whether it is to be read-only:

```
GisNolInsert "C:\Libs\test.nol", 0, False
```

The number of NOLs currently in use can be found using the `GetNumNol` method:

```
lNumNols = GisGetNumNol
```

This method always returns a value of at least 2, because there will always be a standard and temporary NOL. If the user has a workspace open in a Cadcorp SIS desktop product, this is also returned as a NOL, because workspaces can also store named objects.

Only one NOL may be current at a time. The API refers to this as the default NOL, and you can retrieve its name from the `_DefaultNol$` system property. This returns the full pathname if the default NOL is file-based, or the word `(temporary)` or `(workspace)`:

```
sDefaultNol = GisGetStr(SIS_OT_SYSTEM, 0, "_DefaultNol$")
```

To set a newly inserted NOL as default, use the `SetStr` method:

```
GisSetStr SIS_OT_SYSTEM, 0, "_DefaultNol$", "C:\Libs\test.nol"
```

When closing a NOL using the `NolClose` method, you must specify the number of the NOL to be closed, and flag whether changes are to be saved first:

```
GisNolClose 0, True
```

To retrieve the name of a NOL, use the `SIS_OT_NOL` object, specifying the number of the NOL:

```
sNolName = GisGetStr(SIS_OT_NOL, 1, "_name$")
```

◆ Examine the contents of a NOL

You can use the `SIS_OT_NOL` object to retrieve a tab-separated list of any of the named object classes in a NOL. Refer to named object library properties in Cadcorp SIS on-line help for a full list of the NOL properties.

To retrieve a list of named pens in the NOL which is at position 1:

```
sPenList = GisGetStr(SIS_OT_NOL, 1, "_listPen$")
```

The `NolCatalog` method enables you to retrieve a tab-separated list of a named object class from either the current NOL, or all NOLs:

```
sCurrentPens = GisNolCatalog("APen", True)
sAllPens = GisNolCatalog("APen", False)
```

To find the colour of a named pen in this list, you can use the `GetImplicitNolObject` method to retrieve the definition of the pen. (Refer to Pen (Named Object) in Cadcorp SIS on-line help for a full description of implicit pen definitions.) From this definition you can retrieve the red, green, and blue values of the pen.

The following example retrieves the colour of a pen named Forest:

```
sPenDef = GisGetImplicitNolObject("APen", "Forest")
```

This might return a string such as:

```
P_SOLID_69:150:61_0R_0
```

Use standard programming techniques to extract the red (69), green (150) and blue (61) components of this definition. The Visual Basic `Split` function could be used twice: firstly to split the string into an array using the underscore character as a separator, and secondly to split the third element into an array using the colon as a separator:

```
Dim sPenDef As String
Dim aPenDef() As String
Dim aColour() As String
Dim iRed As Integer
Dim iGreen As Integer
Dim iBlue As Integer
sPenDef = GisGetImplicitNolObject("APen", "Forest")
aPenDef = Split(sPenDef, "_")
aColour = Split(aPenDef(2), ":")
iRed = Val(aColour(0))
iGreen = Val(aColour(1))
iBlue = Val(aColour(2))
GisRelease
```

◆ Named views

Cadcorp SIS enables named views to be created, stored in, and recalled from a NOL. When a named view is created using the `DefineNolView` method, the current map view's extent, scale, and projection are captured and stored in the default NOL. If the default NOL is the temporary NOL, it is not possible to save the named view for use in future Cadcorp SIS sessions or for use by other users. If a file-based NOL is set as the default, views can be saved and then recalled by all users who have read access to the file.

Named views are useful to users who need to navigate quickly to known locations on the map, such as administrative areas or planning zones.

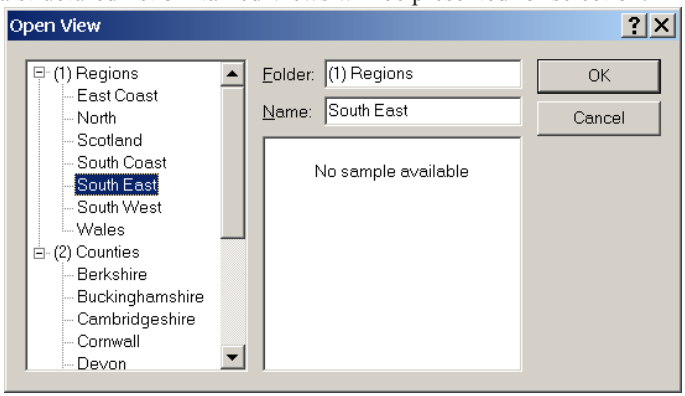
The code below inserts a read/write NOL file at position 2 (after temporary and standard), sets it as default and saves the current view as London. The Nol is then saved to disk:

```
GisNolInsert "C:\Libs\Admin.nol", 2, False
GisSetStr SIS_OT_SYSTEM, 0, "_DefaultNol$", "C:\Libs\Admin.nol"
GisDefineNolView "London"
GisNolSave 2
```

Named views, like other named objects in a NOL, can be stored in a hierarchical structure, by adding a full stop (period) in the name:

```
GisDefineNolView "(1) Regions.South East"
```

When a user of a Cadcorp SIS desktop product uses the **Map>View>Recall** command, a structured list of Named Views will be presented for selection:



The number within the folder name forces Regions to be alphabetically listed before Counties.

This hierarchy can have many levels. For example:

```
GisDefineNolView "Regions.South East.Counties.Hertfordshire"
GisDefineNolView "Regions.South East.Counties.Surrey"
```

If you recall a named view using the API method `RecallNolView`, the view extent, projection and scale of the map window are set immediately.

```
GisRecallNolView "Regions.South East.Counties.Surrey"
```

◆ Named shapes

Users of the Cadcorp SIS desktop products can build named shape objects. A shape can be assigned to a point item for display as a symbol. You can use the API method `DefineNolShape` to create named shapes from a list of items. ➲page 257, **Named lists**

The following code collects the selected items into a list and creates a named shape from them. The origin of the shape is set to the centre of the extent of the items:

```
GisCreateListFromSelection "ItemList"
sExtent = GisGetListExtent("ItemList")
GisSplitExtent X1, Y1, Z1, X2, Y2, Z2, sExtent
XX = (X1 + X2)/2
YY = (Y1 + Y2)/2
GisDefineNolShape "New Shape", XX, YY, 0, 1
```

■ Spatial searches

One of the most powerful capabilities of any GIS system is the ability to perform spatial searches. Cadcorp SIS provides extensive functionality to search map data spatially. The API methods available to GisLink and Cadcorp SIS Control developers

enable custom applications to be written which make full use of these spatial searching techniques.

The `Scan…` methods are mentioned in ➲page 257, **Named lists** and ➲page 261, **Filters**. These methods enable you to write code to scan map data for items which meet set criteria. The criteria can be specific to the API method being used, or by additional arguments to these methods called Filters and Loci.

The Cadcorp SIS API provides the following methods for performing spatial searches:

- `Scan`
- `ScanDataset`
- `ScanOverlay`
- `ScanPointContainers`
- `ScanGeometry`
- `ScanList`
- `Snap2D`

Each of these methods populates a named list with the items found, and returns the number of items found. (`Snap2D` returns a comma-separated string of the position snapped.)

◆ Scan

The `Scan` method enables you to scan all items in the map view, without the need to specify the dataset, the overlay, or a geometry to scan within. The method requires you to specify the status of items to be scanned, ie editable, hittable, visible, or invisible.

`E`	returns editable items
`H`	returns editable and hittable items
`V`	returns editable, hittable, and visible items
`I`	returns all items

You can specify the scan criteria by providing a filter and/or a locus:

```
lNumItems = GisScan("aList", "H", "FarmFilter", "CountyLocus")
```

For information about filters, ➲page 261, **Filters**.

◆ Locus

A locus is an object which includes (passes) or excludes (fails) items, based on their position and geometry. A locus object is the spatial equivalent of a filter object. Think of a locus as a ‘virtual’ geometry, which also specifies how items should interact with it.

For example, the `CreateLocusFromItem` method enables you to create a locus from the current open item. The method requires the geometric test to be specified. The following code creates a locus from the current open item, setting the locus to return items which are inside but not crossing the locus boundary:

```
GisCreateLocusFromItem "aLocus", SIS_GT_CONTAIN, SIS_GM_GEOMETRY
```

This locus is created in the current named object library, and can be re-used in any of the spatial searching methods. For details on managing libraries and their content, ➲page 270, **Named object libraries**.

You can re-use the same spatial extent of a locus with a different geometry test by using the `ChangeLocusTestMode` method:

```
GisChangeLocusTestMode "aLocus", SIS_GT_TOUCH, SIS_GM_GEOMETRY
```

Refer to Locus Methods in the Cadcorp SIS on-line Help for details on the other Locus methods.

◆ ScanDataset

The `Scan` method searches all overlays in the current window. If your application requires a more specific scan, use the `ScanDataset` method. This allows you to search for items by specifying the dataset serial number.

```
lNumItems = GisScanDataset("aList", lSerial, "aFilter","aLocus")
```

For details of the dataset serial number, ➲page 241, **Overlays**.

◆ ScanOverlay

An overlay may have a filter or locus applied to restrict the display of items. The `ScanOverlay` method will search only through items which pass the overlay filter and locus. Any filter or locus used as part of the scan will be additional to the filtering applied to the overlay itself.

◆ ScanPointContainers

This method allows you to provide a position and search for all items which intersect it. For example, a location can be retrieved from a database of postal addresses, and your application can find items of a certain type which contain or enclose this position, such as County or Postcode District:

```
GisScanPointContainers "District", 234567, 117886, 0, "PostalDistricts", ""
```

This example uses the `ScanPointContainers` method as a procedure call rather than as a function, so the parentheses are not required. This is because, in this case, we are going to receive only a single item – the postal district – into the list, so we are not interested in the return value of the function, which is the number of items found.

◆ ScanGeometry

The `ScanGeometry` method uses the current open item as a test item, and scans for editable or hittable items which meet a specified spatial test. The method requires a geometry test and a geometry mode as arguments, as well as the optional filter and locus arguments.

In Cadcorp SIS, a geometry test is provided as a constant specifying a test for the spatial relationship between the test items and all other items allowed by the filter and locus. The geometry tests follow the OpenGIS Consortium (OGC) specification, and are intended to cover all possible spatial relationships between points, lines, and areas.

Despite the fact that these shapes occupy two-dimensional space, they are each referred to as having a dimension. (Do not confuse this with the 2D flat world or the 3D solid world.) Points have a dimension of 0, Lines a dimension of 1, and Areas a dimension of 2.

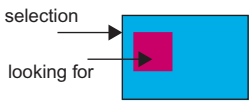
The geometry tests compare a pair of geometries (the test item against each candidate item). The Interior, Boundary, and Exterior of the test item is compared to the same features of the candidate item, resulting in a 3 by 3 matrix of interactions. This matrix is then evaluated against the chosen geometry test.

The geometry tests are:

`Contain`	the interior of the candidate item must be strictly inside the test item
`Cross`	if the items are line items, then they must intersect without being tangential, otherwise their interiors must intersect, with the test item going outside the candidate item
`Touch`	the item's interiors must be disjoint and their boundaries must intersect
`Disjoint From`	the items must be completely separate, with daylight between them
`Equal To`	the two item's geometry must be the same
`Within`	the interior of the test item must be strictly inside the candidate item
`Intersect`	the items must not be disjoint, so they must have a point in common. This is the fastest, and most common test.
`Overlap`	if the two items are line items, they must be tangential, and neither should contain the other. Otherwise, their interiors must intersect, with neither containing the other.
`Crossed By`	if the items are line items, then they must intersect without being tangential. Otherwise, their interiors must intersect, with the candidate item going outside the test item.

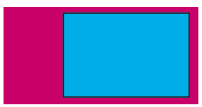
The following tables describe the spatial tests used in Cadcorp SIS.

For example, in the first row, you have selected the rectangular area item, and you want to find the square area item which lies inside it. You can use the Contain, Cross, and Intersect tests. In these tables, the double arrow (↔) at the top of a column indicates that the test item is also returned in the results. So, if you use Contain or Intersect, the test item (a rectangle in this example) is returned.

Test	Contain	Cross	Touch	Disjoint	Equal	Intersect	Overlap	Crossed by	Within
	↔				↔	↔			↔
	✔	✔	.	.	.	✔	.	.	.

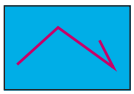
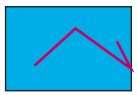
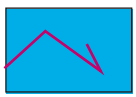
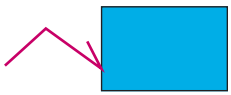
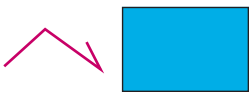
Selected item: area - candidates to find: areas

In this table, the selected item is an area item, and the tests are looking for area items.

Test	Contain ↔	Cross	Touch	Disjoint	Equal ↔	Intersect ↔	Overlap	Crossed by	Within ↔
	✔	✔	·	·	·	✔	·	·	·
	✔	✔	·	·	·	✔	·	·	·
	✔	✔	·	·	·	✔	·	·	·
	·	✔	·	·	·	✔	✔	✔	·
	·	·	✔	·	·	✔	·	·	·
	·	·	·	✔	·	·	·	·	·
	·	·	·	·	·	✔	·	·	✔

Selected item: area – candidates to find: lines

In this table, the selected item is an area item, and the tests are looking for line items.

Test	Contain ↔	Cross	Touch	Disjoint	Equal ↔	Intersect ↔	Overlap	Crossed by	Within ↔
	✔	✔	·	·	·	✔	·	·	·
	✔	✔	·	·	·	✔	·	·	·
	✔	✔	·	·	·	✔	·	·	·
	·	✔	·	·	·	✔	✔	✔	·
	·	·	✔	·	·	✔	·	·	·
	·	·	·	✔	·	·	·	·	
	·	·	·	✔	·	·	·	·	·

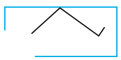
Selected item: area – candidates to find: points

In this table, the selected item is an area item, and the tests are looking for point items.

Test	Contain ↔	Cross	Touch	Disjoint	Equal ↔	Intersect ↔	Overlap	Crossed by	Within ↔
	✔	✔	·	·	·	✔	·	·	·
	·	✔	✔	·	·	✔	·	·	·
	·	·	·	✔	·	·	·	·	·

Selected item: line – candidates to find: lines

In this table, the selected item is a line item, and the tests are looking for line items.

Test	Contain ↔	Cross	Touch	Disjoint	Equal ↔	Intersect ↔	Overlap	Crossed by	Within ↔
	·	·	·	✔	·	·	·	·	·
	·	✔	·	·	·	✔	·	✔	·
	·	·	✔	·	·	✔	·	·	·
	·	·	✔	·	·	✔	·	·	·
	·	·	✔	·	·	✔	·	·	·
	·	✔	·	·	·	✔	·	✔	·
	·	·	·	✔	·	·	·	·	·

Test	Contain ↔	Cross	Touch	Disjoint	Equal ↔	Intersect ↔	Overlap	Crossed by	Within ↔
	·	✔	·	·	·	✔	·	✔	·
	·	·	·	·	·	✔	✔	·	·
	✔	·	·	·	·	✔	·	·	·

Selected item: line – candidates to find: areas

In this table, the selected item is a line item, and the tests are looking for area items.

Test	Contain ↔	Cross	Touch	Disjoint	Equal ↔	Intersect ↔	Overlap	Crossed by	Within ↔
	·	·	·	✔	·	·	·	·	·
	·	·	✔	·	·	✔	·	·	·
	·	·	✔	·	·	✔	·	·	·
	·	·	✔	·	·	✔	·	·	·
	·	✔	·	·	·	✔	✔	✔	·
	·	·	·	✔	·	·	·	·	·
	·	·	✔	·	·	✔	·	·	·
	·	·	·	·	·	✔	·	✔	✔

Selected item: line – candidates to find: points

In this table, the selected item is a line item, and the tests are looking for point items.

Test	Contain ↔	Cross	Touch	Disjoint	Equal ↔	Intersect ↔	Overlap	Crossed by	Within ↔
	·	·	·	✔	·	·	·	·	·
	✔	✔	·	·	·	✔	·	·	·
	·	·	✔	·	·	✔	·	·	·

Selected item: point – candidates to find: areas

In this table, the selected item is a point item, and the tests are looking for area items.

Test	Contain ↔	Cross	Touch	Disjoint	Equal ↔	Intersect ↔	Overlap	Crossed by	Within ↔
	·	·	·	·	·	✔	·	✔	✔
	·	·	✔	·	·	✔	·	·	·
	·	·	·	✔	·	·	·	·	·

Selected item: point – candidates to find: lines

In this table, the selected item is a point item, and the tests are looking for line items.

Test	Contain ↔	Cross	Touch	Disjoint	Equal ↔	Intersect ↔	Overlap	Crossed by	Within ↔
	·	·	·	✔	·	·	·	·	·
	·	·	✔	·	·	✔	·	·	·
	·	·	·	·	·	✔	·	✔	✔
	·	·	·	✔	·	·	·	·	·

Selected item: point – candidates to find: points

In this table, the selected item is a point item, and the tests are looking for point items.

Test	Contain	Cross	Touch	Disjoint	Equal	Intersect	Overlap	Crossed by	Within
	↔				↔	↔			↔
×	✔	·	·	·	✔	✔	·	·	✔
× ×	·	·	·	✔	·	·	·	·	·

In addition to the geometry test, the `ScanGeometry` method also requires a testing mode argument.

A very rapid search can be performed by scanning for the origin of the candidate items. This is a single point, eg the centroid of an area item.

A more accurate search can be performed by scanning for the extents of the candidate items. An item's extent is the bounding rectangle of all of the item's graphics.

The most accurate search is to scan for the geometry of the candidate items. Comparing the spatial interaction of complex shapes can be time consuming, so it is important to choose the most appropriate testing mode.

To search for all address points in the currently selected area, the code might be as follows:

```
GisOpenSel 0
lNumItems = GisScanGeometry ("AddressList", SIS_GT_CONTAIN, SIS_GM_ORIGIN, _
   "AddressFilter", "")
```

The `SIS_GT_CONTAIN` geometry test will return all address points which are wholly within the area. Points which lie on the boundary will not be selected. To find points which are inside or on the boundary, use the `SIS_GT_INTERSECT` geometry test:

```
lNumItems = GisScanGeometry ("AddressList", SIS_GT_INTERSECT, _
  SIS_GM_ORIGIN, "AddressFilter", "")
```

The AddressList will be populated by all items which pass the test. In both of the examples above, the test area itself passes the test: it contains and intersects itself. You should always set a filter to ensure that only the items you are searching for are returned. For example:

```
GisCreateClassTreeFilter "AddressFilter", "-Item +Point"
```

◆ ScanList

For information about named lists, ➲page 257, **Named lists**.

The `ScanList` method enables you to scan the items in a named list, using a filter and/or a locus to create a subset of items:

```
GisCreateListFromSelection "SelectedItems"
GisCreatePropertyFilter "Postcodes", "Exists(""Postcode$"")"
GisScanList "Addresses", "SelectedItems", "Postcodes", ""
```

Notice the use of double quotes (`""`) within the filter formula. For more information on creating and using Filters, ➲page 261, **Filters**.

The `ScanList` method enables you to build a new list based upon an existing one. In this example, the Addresses list is built by scanning the SelectedItems list. If you no longer require the original list, you can scan the list into itself. For example:

```
GisScanList " SomeItems", "SomeItems", "Postcodes", ""
```

This reduces the SomeItems list so that it contains only the items which pass the filter. This technique is similar to the following Visual Basic statement, where a variable value is modified by a formula containing the variable itself:

```
MyNumber = MyNumber - 5
```

◆ Snap2D

The `Snap2D` method enables you to find the item nearest to a specified position. You are required to provide the position, the search radius to be used, the status of items to be searched (Editable or All), and a list of snap codes to be used. As with all the `Scan…` methods, you may also add a filter and/or a locus to restrict the search.

For example, to find the nearest address point to the position 416479,193923 within a 10 metre radius, use the following code:

```
GisCreatePropertyFilter "Address", "Exists(""Postcode$"")"
sPos = GisSnap2D(416479, 193923, 10, False, "P", "Address", "")
```

This method returns a comma-delimited string of the actual position of the address point, eg “416476.456,193927.552,0.000”. You should then use the `SplitPos` method to separate this into numbers:

```
GisSplitPos X, Y, Z, sPos
```

◆ Create a list of areas which overlap a test area

A common requirement for a spatial search is to find all areas which overlap a test area. For example, suppose that the test area represents a Planning Application. The problem is to find all other Planning Applications which apply to some or all of the same area of land. Areas which are adjacent to and touching the test area are not required.

The problem is that none of the nine geometry tests return exactly what is required. The Overlap test finds areas only if they partially overlap, not those which contain or are contained by the test area. The Contain test returns areas only if they are wholly inside the test area. The Intersect test finds what we want, but also finds abutting areas. The Crossing test does not find areas which completely encompass the test area.

The solution is to perform two spatial searches, and then combine the resulting two lists. Because the Intersect test finds all the areas we require, but also finds abutting areas, we can use this to build the first list:

```
GisScanGeometry "IntersectAreas", SIS_GT_INTERSECT, SIS_GM_GEOMETRY, _
   "PlanningFilter", ""
```

The Touch test finds only areas which touch but do not overlap the test area:

```
GisScanGeometry "TouchingAreas", SIS_GT_TOUCH, SIS_GM_GEOMETRY, _
   "PlanningFilter", ""
```

The two lists can be combined by subtracting the second from the first:

```
GisCombineLists "Areas", "IntersectAreas", "TouchingAreas", _
   SIS_BOOLEAN_DIFF
```

Alternatively, the Crossing test finds all the areas except those which completely encompass the test area. The Within test finds these, so the two lists can be combined by adding them:

```
GisScanGeometry "CrossingAreas", SIS_GT_CROSS, SIS_GM_GEOMETRY, _
   "PlanningFilter", ""
GisScanGeometry "WithinAreas", SIS_GT_WITHIN, SIS_GM_GEOMETRY, _
   "PlanningFilter", ""
GisCombineLists "Areas", " CrossingAreas ", " WithinAreas ", SIS_BOOLEAN_OR
```

In Boolean logic, the OR operator has the effect of adding.

◆ Find areas outside but close to an area

Suppose that an area of land is to be developed, and your program must find all properties which lie within 150 metres of the development area, to notify them of the possibility of noise and dust. The properties within the development area are not required because those residents have already been notified of the plans.

To solve this problem, create a buffer to specify the 150 metre zone. It would be possible to perform two searches, and subtract the results of one from the other. But you can make a single search by creating the buffer as a locus, using this locus to restrict the search. By using the Disjoint test to find areas outside the development area, you can obtain a list of items which are within or overlapping the buffer but outside the development area.

1 First, put the development area into a named list:

```
GisCreateListFromSelection "DevArea"
```

2 Now create a 150 metre buffer locus around it, without deleting the area itself:

```
GisCreateBufferLocusFromItems "DevArea", False, "Buffer", 150, 0
```

3 Set the locus test mode so that it is restricted to items crossing it:

```
GisChangeLocusTestMode "Buffer", SIS_GT_CROSS, SIS_GM_GEOMETRY
```

4 Finally, make the development area current and perform the spatial search, for areas disjoint from it but inside or overlapping the buffer locus:

```
GisOpenList "DevArea", 0
GisScanGeometry "Properties", SIS_GT_DISJOINT, SIS_GM_GEOMETRY, "", _
   "Buffer"
```

■ Schemas

◆ Create and apply a schema

This code shows you how to create and apply a schema for the current overlay. It uses the ID, Pen, Length, Container Area, and Date properties.

```
Private Sub btnSchemaCreate_Click()

  ' Create a scheme and apply to current overlay
  ' Create a blank schema
  GisLoadSchema ""
  ' Build up the columns for the schema
  ' Add Integer attribute
  GisInsertSchemaColumn "_id&", 0
  ' Add String attribute
  GisInsertSchemaColumn "_pen$", 1
  ' Add Floating point attribute
  GisInsertSchemaColumn "_length#", 2
  ' Add Formula to find the ID of the container on overlay "Areas" of an item
  GisInsertSchemaColumn "FindContainer(" & Chr(34) & "Areas" & Chr(34) _
    & ")._id&", 3
  ' Add Formula time item selected
  GisInsertSchemaColumn "FormatDate(Date()," & Chr(34) _
    & "Time: %X" & Chr(34) & ")", 4

  ' Set the column names
  GisSetStr SIS_OT_SCHEMACOLUMN, 0, "_description$", "Items ID Number"
  GisSetStr SIS_OT_SCHEMACOLUMN, 1, "_description$", "Pen"
  GisSetStr SIS_OT_SCHEMACOLUMN, 2, "_description$", "Length"
  GisSetStr SIS_OT_SCHEMACOLUMN, 3, "_description$", "Container ID"
  GisSetStr SIS_OT_SCHEMACOLUMN, 4, "_description$", "Time"

  ' Set Column for MapTip to 0
  GisSetInt SIS_OT_SCHEMA, 0, "_nMapTipColumn&", 0

  ' Save and assign schema to the current overlay
  GisStoreSchema "Schema1"
```

```
  GisSetOverlaySchema (GisGetInt(SIS_OT_WINDOW, 0, "_nDefaultOverlay&")), _
    "Schema1"
  GisRelease
End Sub
```

◆ View a schema in a table window

This code shows you how to display a table window, showing the schema for the current overlay.

```
Private Sub btnSchemaTable_Click()

  ' Create a table window and display the schema for the current overlay
  ' Create a table window
  GisSwdNewWindowTable

  ' Set overlay at position one
  currentOverlayIndex = GisGetInt(SIS_OT_WINDOW, 0, "_nDefaultOverlay&")
  GisSetInt SIS_OT_WINDOW, 0, "_nOverlayShow&", currentOverlayIndex

  ' Tile Windows Horizontally
  GisWndTileHorizontal

  GisRelease
End Sub

  ' Add Formula time item selected
  GisInsertSchemaColumn "FormatDate(Date()," & Chr(34) & "Time: %X" _
    & Chr(34) & ")", 4

  ' Set the column names
  GisSetStr SIS_OT_SCHEMACOLUMN, 0, "_description$", "Items ID Number"
  GisSetStr SIS_OT_SCHEMACOLUMN, 1, "_description$", "Pen"
  GisSetStr SIS_OT_SCHEMACOLUMN, 2, "_description$", "Length"
  GisSetStr SIS_OT_SCHEMACOLUMN, 3, "_description$", "Container ID"
  GisSetStr SIS_OT_SCHEMACOLUMN, 4, "_description$", "Time"

  ' Set Column for MapTip to 0
  GisSetInt SIS_OT_SCHEMA, 0, "_nMapTipColumn&", 0

  ' Save and assign schema to the current overlay
  GisStoreSchema "Schema1"
  GisSetOverlaySchema (GisGetInt(SIS_OT_WINDOW, 0, "_nDefaultOverlay&")), _
    "Schema1"
  GisRelease
End Sub
```

◆ View a schema in a table window

This code shows you how to display a table window, showing the schema for the current overlay.

```
Private Sub btnSchemaTable_Click()
```

```
  ' Create a table window and display the schema for the current overlay
  ' Create a table window
  GisSwdNewWindowTable

  ' Set overlay at position one
  currentOverlayIndex = GisGetInt(SIS_OT_WINDOW, 0, "_nDefaultOverlay&")
  GisSetInt SIS_OT_WINDOW, 0, "_nOverlayShow&", currentOverlayIndex

  ' Tile Windows Horizontally
  GisWndTileHorizontal

  GisRelease
End Sub
```

■ Themes

◆ Create a bar chart theme

This example shows you how to create a theme on the current overlay, displaying a bar chart of the X and Y co-ordinates of the origin of each item.

```
Private Sub btnThemeBar_Click()
 ' a bar chart theme for the X,Y values

 GisCreateBarTheme 2
 GisSetStr SIS_OT_THEME, 0, "_title$", "Test Bar Chart Theme"

 ' Set the ThemeComponent for the fomula, 0 to _ox#, 1 to _oy#, 2 to etc
 GisSetStr SIS_OT_THEMECOMPONENT, 0, "_formula$", "_ox#"
 GisSetStr SIS_OT_THEMECOMPONENT, 1, "_formula$", "_oy#"

 ' Set the ThemeComponent for the titles, 0 to X-axis, 1 to Y-axis, 2 to etc
 GisSetStr SIS_OT_THEMECOMPONENT, 0, "_title$", "X-axis"
 GisSetStr SIS_OT_THEMECOMPONENT, 1, "_title$", "Y-axis"

 ' Set the ThemeComponent for the Colour, 0 to X-axis, 1 to Y-axis, 2 to etc
 GisSetStr SIS_OT_THEMECOMPONENT, 0, "_brush$", "B_SO_255:0:0"
 GisSetStr SIS_OT_THEMECOMPONENT, 1, "_brush$", "B_SO_0:0:255"

 ' Set the Scale Function for Height of the Bars
 GisSetInt SIS_OT_THEME, 0, "_function&", SIS_FUNCTION_LOG10

 ' Set the "at value" according to the approximate value of _ox# or _oy#
 GisSetFlt SIS_OT_THEME, 0, "_value#", 300000

 ' Close and Save New Theme Style
 GisStoreTheme "TestBarTheme"

' Recall Theme on current overlay
 GisInsertOverlayTheme GisGetInt(SIS_OT_WINDOW, 0, "_nDefaultOverlay&"), _
  "TestBarTheme", 0
 GisRedraw SIS_CURRENTWINDOW

 GisRelease
End Sub
```

◆ Create a pie chart theme

This code shows you how to create a theme on the current overlay, displaying a pie chart of the X and Y co-ordinates of the origin of each item.

```
Private Sub btnThemePie_Click()

 ' a pie chart theme for the X,Y values

 GisCreatePieTheme 2
 GisSetStr SIS_OT_THEME, 0, "_title$", "Test Pie Chart Theme"

 ' Set the ThemeComponent for the fomula, 0 to _ox#, 1 to _oy#, 2 to etc
 GisSetStr SIS_OT_THEMECOMPONENT, 0, "_formula$", "_ox#"
 GisSetStr SIS_OT_THEMECOMPONENT, 1, "_formula$", "_oy#"

 ' Set the ThemeComponent for the titles, 0 to X-axis, 1 to Y-axis, 2 to etc
 GisSetStr SIS_OT_THEMECOMPONENT, 0, "_title$", "X-axis"
 GisSetStr SIS_OT_THEMECOMPONENT, 1, "_title$", "Y-axis"

 ' Set the ThemeComponent for the Colour, 0 to X-axis, 1 to Y-axis, 2 to etc
 GisSetStr SIS_OT_THEMECOMPONENT, 0, "_brush$", "B_SO_255:0:0"
 GisSetStr SIS_OT_THEMECOMPONENT, 1, "_brush$", "B_SO_0:0:255"

 ' Set the Scale Function for size of shapes
 GisSetInt SIS_OT_THEME, 0, "_function&", SIS_FUNCTION_CONSTANT

 ' close and save new theme
 GisStoreTheme "TestPieTheme"
 GisInsertOverlayTheme GisGetInt(SIS_OT_WINDOW, 0, "_nDefaultOverlay&"), _
  "TestPieTheme", 0
 GisRedraw SIS_CURRENTWINDOW

 GisRelease
End Sub
```

◆ Create an extruded theme

This code shows you how to create a theme on the current overlay for the item ID values multiplied by 10. You can create an extruded theme only if you have the 3D capabilities of Cadcorp SIS Map Modeller.

```
Private Sub btnThemeExtruded_Click()

 ' an extruded chart theme for the _id&

 GisCreateExtrudeTheme "_id&*10"
 GisSetStr SIS_OT_THEME, 0, "_title$", "Test Extruded Chart Theme"

 ' Close and Save New Theme
 GisStoreTheme "TestExtrudedTheme"
 GisInsertOverlayTheme GisGetInt(SIS_OT_WINDOW, 0, "_nDefaultOverlay&"), _
    "TestExtrudedTheme", 0
```

```
  GisRedraw SIS_CURRENTWINDOW

  GisRelease
End Sub
```

◆ Create a label theme

This code shows you how to create a theme on the current overlay displaying a label of the ID of each item.

```
Private Sub btnThemeLabel_Click()
  ' Create a label theme for _id&

  GisCreateLabelTheme "_id&"
  GisSetStr SIS_OT_THEME, 0, "_title$", "Test Label Chart Theme"
  GisSetInt SIS_OT_THEME, 0, "_point_height&", 15
  GisSetInt SIS_OT_THEME, 0, "_text_opaque&", True
  GisSetInt SIS_OT_THEME, 0, "_text_italic&", True

  ' Text box will take on overlay brush for the background colour
  GisSetStr SIS_OT_THEME, 0, "_brush$", "B_SO_255:255:255"

  ' Close and Save New Theme
  GisStoreTheme "TestLabelTheme"
  GisInsertOverlayTheme GisGetInt(SIS_OT_WINDOW, 0, "_nDefaultOverlay&"), _
    "TestLabelTheme", 0
  GisRedraw SIS_CURRENTWINDOW

  GisRelease
  Me.WindowState = vbNormal

End Sub
```

◆ Create a range theme

This code shows you how to create a theme on the current overlay, setting the brush of each item according to the value of its X co-ordinate. You will need an overlay displaying areas for testing this program.

```
Private Sub btnThemeRange_Click()
   Dim DefOverlay  As Long
   Dim howMany As Integer
   Dim x As Integer
   Dim themeList As String
   Dim Title As String
  ' create a range theme for _ox#

  ' Find Min Max Values for _ox#
  minValue = 0
  maxValue = 0

  GisScanOverlay "OverlayItems", GisGetInt(SIS_OT_WINDOW, 0, _
    "_nDefaultOverlay&"), "", ""
  For x = 0 To GisGetListSize("OverlayItems") - 1
     GisOpenList "OverlayItems", x
```

```
    ' Find Min Value
    ox = GisGetFlt(SIS_OT_CURITEM, 0, "_ox#")
    If ox < minValue Then minValue = ox
    If ox > maxValue Then maxValue = ox
  Next
  GisEmptyList "OverlayItems"

  ' Increment for ranges
  NoOfRanges = 5
  RangeIncrement = (maxValue - minValue) / NoOfRanges

  GisCreateRangeTheme "_ox#", NoOfRanges
  GisSetStr SIS_OT_THEME, 0, "_title$", "Test Range Chart Theme"
  GisSetInt SIS_OT_THEME, 0, "_point_height&", 15
  GisSetInt SIS_OT_THEME, 0, "_text_opaque&", True
  GisSetInt SIS_OT_THEME, 0, "_text_italic&", True

  ' Set values for Ranges
  GisSetFlt SIS_OT_THEMECOMPONENT, 0, "_value#", minValue

  For x = 1 To NoOfRanges - 1
    GisSetFlt SIS_OT_THEMECOMPONENT, x, "_value#", minValue + _
     (RangeIncrement * x)
  Next

  ' Close and Save New Theme
  GisStoreTheme "TestRangeTheme"
  GisInsertOverlayTheme GisGetInt(SIS_OT_WINDOW, 0, "_nDefaultOverlay&"), _
   "TestRangeTheme", 0
  GisRedraw SIS_CURRENTWINDOW

  GisRelease
End Sub
```

◆ Create a graduated theme

This code shows you how to create a theme on the current overlay, displaying a circle at the origin of each item, graduated in size according to the value of its ID.

```
Private Sub btnThemeGraduated_Click()
 ' create a graduated theme for _id&

 GisCreateGraduatedTheme "_id&"
 GisSetStr SIS_OT_THEME, 0, "_title$", "Test Graduated Theme"
 GisSetInt SIS_OT_THEME, 0, "_point_height&", 15
 GisSetInt SIS_OT_THEME, 0, "_text_italic&", True
 GisSetStr SIS_OT_THEME, 0, "_graduated_shape$", "Circle"
 GisSetStr SIS_OT_THEME, 0, "_graduated_brush$", "B_SO_0:255:255"

 ' close and save new theme
 GisStoreTheme "TestGraduatedTheme"
 GisInsertOverlayTheme GisGetInt(SIS_OT_WINDOW, 0, "_nDefaultOverlay&"), _
   "TestGraduatedTheme", 0
```

```
  GisRedraw SIS_CURRENTWINDOW
  GisRelease

End Sub
```

◆ Edit an existing theme

This code shows you how to edit an existing theme. The program changes the colours and the scale function of the theme TestPieTheme.

```
Private Sub btnEditTheme_Click()

  ' Edit a theme
  ' This will not affect themes of the same name in use.
  ' Only new instances of the theme will be affected.
  ' For instant effect, use GetOverlayTheme

  ' Make the chosen theme current
  GisLoadTheme "TestPieTheme"

  ' Set the ThemeComponent for the Colour, 0 to X-axis, 1 to Y-axis, 2 to etc
  GisSetStr SIS_OT_THEMECOMPONENT, 0, "_brush$", "B_SO_255:255:0"
  GisSetStr SIS_OT_THEMECOMPONENT, 1, "_brush$", "B_SO_0:255:255"

  ' Set the Scale Function for size of shapes
  GisSetInt SIS_OT_THEME, 0, "_function&", SIS_FUNCTION_LOG10

  ' Close and Save New Theme
  GisStoreTheme "TestPieTheme"

  GisRelease
  Me.WindowState = vbNormal

End Sub
```

◆ Edit a theme at a known position

This code shows you how to edit a theme at a known position in the list. The program changes the colours and the scale function of the `TestPieTheme`. The theme is modified in the NOL, but you will not see the effects if it is in use on an overlay.

```
Private Sub btnEditThemeKnown_Click()

  ' Edit a theme at a known position in list.

  GisGetOverlayTheme 1, "TestPieTheme", 0
  ' 1 is the overlay position, 0 is the index of the Theme

  GisLoadTheme "TestPieTheme"

  ' Set the ThemeComponent for the Colour, 0 to X-axis, 1 to Y-axis, 2 to etc
  GisSetStr SIS_OT_THEMECOMPONENT, 0, "_brush$", "B_SO_255:128:0"
  GisSetStr SIS_OT_THEMECOMPONENT, 1, "_brush$", "B_SO_0:128:255"

  ' Set the Scale Function for size of shapes
  GisSetInt SIS_OT_THEME, 0, "_function&", SIS_FUNCTION_SQUAREROOT
```

```
  ' Close and Save New Theme
  GisStoreTheme "TestPieTheme"

  ' Remove Theme Old Theme add New Theme
  GisRemoveOverlayTheme 1, 0
  GisInsertOverlayTheme 1, "TestPieTheme", 0

  GisRedraw SIS_CURRENTWINDOW
  GisRelease

End Sub
```

◆ Find out the number of themes on an overlay

This code shows you how to find out the number and names of the themes on the current overlay.

```
Private Sub btnThemesHowMany_Click()
  Dim DefOverlay  As Long
  Dim howMany As Integer
  Dim x As Integer
  Dim themeList As String
  Dim Title As String

  ' How many themes and their titles
  DefOverlay = GisGetInt(SIS_OT_WINDOW, 0, "_nDefaultOverlay&")
  howMany = GisGetInt(SIS_OT_OVERLAY, DefOverlay, "_nTheme&")

   For x = 0 To howMany - 1
    GisGetOverlayTheme 1, "EMPTY", x
    GisLoadTheme "EMPTY"
    Title = GisGetStr(SIS_OT_THEME, 0, "_title$")
    themeList = themeList & vbCrLf & x & ". " & Title
   Next x

  If themeList <> "" Then
   MsgBox themeList, vbInformation, "Themes in Current Overlay"
  Else
   MsgBox "The Current Overlay does not contain any Themes", _
     vbInformation, "Themes in  Current Overlay"
  End If

  GisRelease
End Sub
```

◆ Remove a theme

This code shows you how to remove a theme from the current overlay.

```
Private Sub btnThemeRemove_Click()
   Dim DefOverlay  As Long
   Dim howMany As Integer
   Dim x As Integer
```

```
    ' Remove a theme or Themes

    DefOverlay = GisGetInt(SIS_OT_WINDOW, 0, "_nDefaultOverlay&")
    howMany = GisGetInt(SIS_OT_OVERLAY, DefOverlay, "_nTheme&")

    For x = howMany - 1 To 0 Step -1
      GisRemoveOverlayTheme DefOverlay, x
    Next x

    MsgBox howMany & " Themes removed", vbInformation

    GisRelease
End Sub
```

■ Printing

◆ Printing with Wizards

Printing maps is an essential part of a GIS application. Users of the Cadcorp SIS desktop applications such as Cadcorp SIS Map Manager will be familiar with using the Print Wizard to format a map view onto a print template, with tools to add scale bars, north points, grids, and titling.

This functionality is available to the GisLink customiser by invoking the Print Wizard commands, AComPrintTemplate or AComPrintTemplateQuick. Developers using the Cadcorp SIS Control have similar functionality available by using the `AComPrintTemplateWizard` or `AComPrintTemplateWizardQuick` methods.

Cadcorp SIS Control applications cannot use the standard Print Wizard, for two reasons:

- the Print Setup button in the Wizard cannot be used with the Cadcorp SIS Control, for technical reasons
- the Wizard provides a Create SWD option, which is incompatible with the Cadcorp SIS Control environment

In other respects, the dialogs displayed to the user are identical.

When calling the Print Wizard from a Cadcorp SIS Control application, the resulting print will be sent to the default printing device. If you wish to provide the user with the option to select a printer, this is best achieved by displaying the Print Setup dialog first. (Refer to the documentation of your chosen programming tool for details on displaying Print Setup.)

◆ Printing using GisLink

The Print Wizard enables the user to choose a print template, position the map view within the template, then, optionally, add secondary graphics such as a scale bar and grid. The API provides the following methods to create secondary graphics:

- `CreateGraticule`
- `CreateKeyMap`
- `CreateNorthPoint`
- `CreateScaleBar`
- `GetOverlayThemeLegend / CreateItem`

- `PhotoGrid`

If you are not calling the Print Wizard, you can place a map view onto a template using GisLink as follows:

1. Compose the current map view. This holds the composition of the view ready for placing within a print template.
2. Create a new empty SWD ready to receive the map view:

```
SwdNew
```

3. Place the print template in the new SWD. The photo area of this template will be filled with the previously composed view.

```
PlacePrintTemplate
```

4. Print the template.

```
GisCompose
GisSwdNew
GisPlacePrintTemplate "A4 Landscape", 0, 1250
GisDoCommand "AComPrint"
GisRelease
```

◆ Using Cadcorp SIS Control

You can use a similar sequence when programming for the Cadcorp SIS Control. Because you cannot use the `SwdNew` method in the Cadcorp SIS Control, we suggest that you place a second instance of the control on your main form, with its `Visible` property set to False. When the current map view (displayed in the first Cadcorp SIS Control) is composed, the `PlacePrintTemplate` method is used on the second control, which is then made visible:

```
Sis.Compose
Sis2.PlacePrintTemplate "A4 Landscape", 0, 1250
Sis.Visible = False
Sis2.Visible = True
```

However, the Cadcorp SIS Control cannot call the `AComPrint` command, so you must use your programming language's functionality to display the Windows Print dialog, and retrieve the chosen printer from this.

In Visual Basic, the simplest method is to retrieve a handle to the Device Context of the selected printer. The device context can then be passed to the Cadcorp SIS Control's `SIS_OT_PRINTER` object, and the `SendPrint` method used to print the composed template to the printer.

Once the map is printed, it is good practise to delete the device context. To do this you will need to declare the Windows GDI function `DeleteDC` in a module in your application:

```
Public Declare Function DeleteDC Lib "gdi32" (ByVal hdc As Long) As Long
```

You will also need to add the Windows Common Dialog component (ComDlg32.OCX) to your application.

1 First compose the map view and place it in a second Cadcorp SIS Control:

```
Private Sub cmdCompose_Click()
   Sis.Compose
   Sis2.PlacePrintTemplate "A4 Landscape", 0, 1250
   Sis.Visible = False
   Sis2.Visible = True
End Sub
```

2 Next display the Print dialog to allow the user to select a printer. You should retrieve the device context `hDC` and the number of copies required. You should pass the device context to the `SIS_OT_PRINTER` object in hexadecimal, then call the `SendPrint` method for each copy required:

```
Private Sub cmdPrint_Click()
   Dim lHandle As Long
   Dim iCopies As Integer
   Dim iPrint As Integer

   With dlgPrint
      .CancelError = True
      .PrinterDefault = False
      .Flags = cdlPDReturnDC
      On Error GoTo cancelled
      .ShowPrinter
      On Error GoTo 0
      lHandle = .hdc
      iCopies = .Copies
   End With

   Sis2.SetStr SIS_OT_PRINTER, 0, "_device$", "&H" & Hex(lHandle)
   For iPrint = 1 To iCopies
      Sis2.SendPrint "", "", "", SIS_PRINTCAPS_QUERY, 1
   Next iPrint
   DeleteDC (lHandle)

cancelled:
End Sub
```

Appendix 1

Availability of methods

■ **Introduction** 299
■ **Availability of methods in Cadcorp SIS products** 299

■ Introduction

This appendix lists the methods in Cadcorp SIS in alphabetical order, showing in which products they are available.

Each method is described in ➲Chapter 7: **"Methods"**.

■ Availability of methods in Cadcorp SIS products

Method	MM	ME	MD	OV	OM	OD	ASC
`Activate`				✓	✓	✓	
`ActivateWnd`	✓	✓	✓				
`AddCommand`	✓	✓	✓	✓	✓	✓	
`AddToList`	✓	✓	✓	✓	✓	✓	✓
`AllowCommands`	✓	✓	✓		✓	✓	
`BezierTo`		✓	✓			✓	✓
`BulgeTo`	✓	✓	✓		✓	✓	✓
`CallCommand`	✓	✓	✓				
`CanDoCommand`	✓	✓	✓	✓	✓	✓	✓
`ChangeFeatureFilter`	✓	✓	✓		✓	✓	✓
`ChangeLocusTestMode`	✓	✓	✓		✓	✓	✓
`ChangePrjUnits`	✓	✓	✓		✓	✓	✓
`ChangeValueListFilter`	✓	✓	✓		✓	✓	✓
`CleanLines`		✓	✓			✓	✓
`CloseDataset`	✓	✓	✓		✓	✓	✓
`CloseIndexDatasetTile`	✓	✓	✓		✓	✓	✓
`CloseItem`	✓	✓	✓	✓	✓	✓	✓
`CombineFilter`	✓	✓	✓		✓	✓	✓
`CombineLists`	✓	✓	✓		✓	✓	✓
`CombineLocus`		✓	✓			✓	✓
`CompactDataset`	✓	✓	✓	✓	✓	✓	✓
`Compose`	✓	✓	✓	✓	✓	✓	✓
`Copy`	✓	✓	✓	✓	✓	✓	

Method	MM	ME	MD	OV	OM	OD	ASC
CopyFeatureCode		✓	✓			✓	✓
CopyListItems	✓	✓	✓		✓	✓	✓
CopyThemeComponent	✓	✓	✓		✓	✓	✓
CreateAreaFromLines	✓	✓	✓	✓	✓	✓	✓
CreateAssembly		✓	✓			✓	✓
CreateBackdropOverlay	✓	✓	✓	✓	✓	✓	✓
CreateBarTheme	✓	✓	✓	✓	✓	✓	✓
CreateBds	✓	✓	✓		✓	✓	✓
CreateBitmap	✓	✓	✓		✓	✓	✓
CreateBlock		✓	✓			✓	✓
CreateBoolean		✓	✓			✓	✓
CreateBoundary		✓	✓			✓	✓
CreateBoxLabel	✓	✓	✓		✓	✓	✓
CreateBoxText	✓	✓	✓		✓	✓	✓
CreateBufferFromItems		✓	✓			✓	✓
CreateBufferLocusFromItems		✓	✓			✓	✓
CreateCircle	✓	✓	✓		✓	✓	✓
CreateCircleLocus		✓	✓			✓	✓
CreateClassTreeFilter	✓	✓	✓		✓	✓	✓
CreateCombinedFilter	✓	✓	✓		✓	✓	✓
CreateContourTheme			✓			✓	✓
CreateConvexHull		✓	✓			✓	✓
CreateDataSourceOverlay	✓	✓	✓	✓	✓	✓	✓
CreateDbBlobOverlay	✓	✓	✓		✓	✓	✓
CreateDbOverlay	✓	✓	✓		✓	✓	✓
CreateDbPointOverlay	✓	✓	✓		✓	✓	✓
CreateDbTable	✓	✓	✓		✓	✓	✓
CreateDisplacement		✓	✓			✓	✓
CreateDotTheme	✓	✓	✓	✓	✓	✓	✓
CreateDoubleBoolean		✓	✓			✓	✓
CreateEllipse	✓	✓	✓	✓	✓	✓	✓
CreateExtrudeTheme			✓			✓	✓
CreateExtrusion			✓			✓	✓
CreateFeatureFilter	✓	✓	✓		✓	✓	✓
CreateFlowTheme			✓			✓	✓
CreateFormulaGrid			✓			✓	✓
CreateGraduatedTheme	✓	✓	✓	✓	✓	✓	✓
CreateGraticule	✓	✓	✓		✓	✓	✓
CreateGridFromQZone			✓			✓	✓
CreateGroup	✓	✓	✓		✓	✓	✓
CreateGroupFromItems	✓	✓	✓		✓	✓	✓
CreateIndexCoverage	✓	✓	✓		✓	✓	✓
CreateIndexOverlay	✓	✓	✓	✓	✓	✓	✓
CreateIndividualTheme	✓	✓	✓	✓	✓	✓	✓

Method	MM	ME	MD	OV	OM	OD	ASC
CreateInsert	✓	✓	✓		✓	✓	✓
CreateInternalOverlay	✓	✓	✓	✓	✓	✓	✓
CreateIsoRoute		✓	✓			✓	✓
CreateItem	✓	✓	✓	✓	✓	✓	✓
CreateItemB	✓	✓	✓		✓	✓	✓
CreateItemFromLocus		✓	✓			✓	✓
CreateKeyMap	✓	✓	✓		✓	✓	✓
CreateLabelTheme	✓	✓	✓	✓	✓	✓	✓
CreateLineText	✓	✓	✓		✓	✓	✓
CreateLinkFilter	✓	✓	✓		✓	✓	✓
CreateListFromOverlay	✓	✓	✓	✓	✓	✓	✓
CreateListFromSelection	✓	✓	✓		✓	✓	
CreateLocusFromItem	✓	✓	✓		✓	✓	✓
CreateNorthPoint	✓	✓	✓		✓	✓	✓
CreateOpenGisSqlOverlay	✓	✓	✓		✓	✓	✓
CreatePhaseOverlay	✓	✓	✓		✓	✓	✓
CreatePhoto	✓	✓	✓		✓	✓	✓
CreatePieTheme	✓	✓	✓	✓	✓	✓	✓
CreatePoint	✓	✓	✓	✓	✓	✓	✓
CreatePropertyFilter	✓	✓	✓	✓	✓	✓	✓
CreateQZoneFromGrid			✓			✓	✓
CreateRangeTheme	✓	✓	✓	✓	✓	✓	✓
CreateRectLocus		✓	✓			✓	✓
CreateRectangle	✓	✓	✓	✓	✓	✓	✓
CreateReliefTheme	✓	✓	✓	✓	✓	✓	✓
CreateRubberSheet		✓	✓			✓	✓
CreateScaleBar	✓	✓	✓		✓	✓	✓
CreateScatterGrid			✓			✓	✓
CreateSurface			✓			✓	✓
CreateText	✓	✓	✓	✓	✓	✓	✓
CreateThiessen			✓			✓	✓
CreateTin			✓			✓	✓
CreateTopoTheme		✓	✓			✓	✓
CreateValueListFilter	✓	✓	✓		✓	✓	✓
DefineNolDatum	✓	✓	✓		✓	✓	✓
DefineNolItem		✓	✓			✓	✓
DefineNolItemFromLocus		✓	✓			✓	✓
DefineNolObject		✓	✓			✓	✓
DefineNolPrintTemplate	✓	✓	✓		✓	✓	✓
DefineNolPrjLatLon	✓	✓	✓		✓	✓	✓
DefineNolPrjTm	✓	✓	✓		✓	✓	✓
DefineNolShape		✓	✓			✓	✓
DefineNolView	✓	✓	✓		✓	✓	✓
DefineRecordset	✓	✓	✓		✓	✓	✓

Method	MM	ME	MD	OV	OM	OD	ASC
Delete	✓	✓	✓	✓	✓	✓	✓
DeleteItem	✓	✓	✓	✓	✓	✓	✓
DeleteNolObject		✓	✓			✓	✓
DescribeProperty	✓	✓	✓		✓	✓	✓
DeselectAll	✓	✓	✓	✓	✓	✓	✓
DigitiserSnap		✓	✓				
DoCommand	✓	✓	✓	✓	✓	✓	
DrawList	✓	✓	✓	✓	✓	✓	
EmptyGroup	✓	✓	✓		✓	✓	✓
EmptyList	✓	✓	✓	✓	✓	✓	✓
EnsureOpenWithin	✓	✓	✓	✓	✓	✓	✓
EvaluateFlt	✓	✓	✓	✓	✓	✓	✓
EvaluateInt	✓	✓	✓	✓	✓	✓	✓
EvaluateStr	✓	✓	✓	✓	✓	✓	✓
Exit	✓	✓	✓				
ExplodeOverlayTheme	✓	✓	✓		✓	✓	✓
Export	✓	✓	✓		✓	✓	✓
ExportBds	✓	✓	✓		✓	✓	
ExportBmp	✓	✓	✓		✓	✓	
ExportFeatureTable		✓	✓			✓	✓
ExportJpeg	✓	✓	✓		✓	✓	
ExportPdf	✓	✓	✓		✓	✓	
ExportPng	✓	✓	✓		✓	✓	
ExportVrml			✓			✓	
ExportWmf	✓	✓	✓		✓	✓	
FindDatasetOverlay	✓	✓	✓		✓	✓	✓
FindExternalDataset	✓	✓	✓		✓	✓	✓
Get3DEye			✓			✓	
Get3DLook			✓			✓	
GetAxesAngle	✓	✓	✓		✓	✓	✓
GetAxesFromLatLonHgt	✓	✓	✓	✓	✓	✓	✓
GetAxesPrj	✓	✓	✓	✓	✓	✓	✓
GetAxesType	✓	✓	✓	✓	✓	✓	✓
GetBlob		✓	✓			✓	✓
GetBlobB		✓	✓			✓	✓
GetBlobExtent	✓	✓	✓		✓	✓	✓
GetCommandTick	✓	✓	✓		✓	✓	
GetCoordExtent	✓	✓	✓		✓	✓	✓
GetCoordString	✓	✓	✓	✓	✓	✓	✓
GetDataset	✓	✓	✓		✓	✓	✓
GetDatasetContainer	✓	✓	✓		✓	✓	✓
GetDatasetExtent	✓	✓	✓		✓	✓	✓
GetDatasetPrj	✓	✓	✓		✓	✓	✓
GetDisplayExtent	✓	✓	✓	✓	✓	✓	

Method	MM	ME	MD	OV	OM	OD	ASC
GetErrorString				✓	✓	✓	
GetExtent	✓	✓	✓		✓	✓	✓
GetFeatureFilterBranches	✓	✓	✓		✓	✓	✓
GetFeatureTableBranches		✓	✓			✓	✓
GetFlt	✓	✓	✓	✓	✓	✓	✓
GetGeomAngleFromLength	✓	✓	✓		✓	✓	✓
GetGeomDim	✓	✓	✓		✓	✓	✓
GetGeomLength	✓	✓	✓		✓	✓	✓
GetGeomLengthUpto	✓	✓	✓		✓	✓	✓
GetGeomNumPt	✓	✓	✓		✓	✓	✓
GetGeomNumSeg	✓	✓	✓		✓	✓	✓
GetGeomPosFromLength	✓	✓	✓		✓	✓	✓
GetGeomPt	✓	✓	✓		✓	✓	✓
GetGeomSegAxis	✓	✓	✓		✓	✓	✓
GetGeomSegBulge	✓	✓	✓		✓	✓	✓
GetGeomSegShape	✓	✓	✓		✓	✓	✓
GetGeomSelfIntersection	✓	✓	✓		✓	✓	✓
GetGeomTgtFromLength	✓	✓	✓		✓	✓	✓
GetGridItemValue			✓			✓	✓
GetHook	✓	✓	✓		✓	✓	✓
GetImplicitNolObject	✓	✓	✓		✓	✓	✓
GetInt	✓	✓	✓	✓	✓	✓	✓
GetLatLonHgtFromAxes	✓	✓	✓	✓	✓	✓	✓
GetListExtent	✓	✓	✓		✓	✓	✓
GetListSize	✓	✓	✓	✓	✓	✓	✓
GetNumAscClients							✓
GetNumGeom	✓	✓	✓		✓	✓	✓
GetNumNol	✓	✓	✓	✓	✓	✓	✓
GetNumSel	✓	✓	✓	✓	✓	✓	
GetNumSwd	✓	✓	✓				
GetNumWnd	✓	✓	✓				
GetOverlayFilter	✓	✓	✓		✓	✓	✓
GetOverlayLocus		✓	✓			✓	✓
GetOverlaySchema	✓	✓	✓		✓	✓	✓
GetOverlayTheme	✓	✓	✓		✓	✓	✓
GetOverlayThemeLegend	✓	✓	✓	✓	✓	✓	✓
GetPhotoWorldPos	✓	✓	✓		✓	✓	✓
GetPos	✓	✓	✓				
GetPosEx	✓	✓	✓				
GetPropertyDescription	✓	✓	✓	✓	✓	✓	✓
GetSpatialReference		✓	✓			✓	✓
GetStr	✓	✓	✓	✓	✓	✓	✓
GetStrW				✓	✓	✓	✓
GetViewExtent	✓	✓	✓	✓	✓	✓	✓

Method	MM	ME	MD	OV	OM	OD	ASC
GetViewPos							✓
GetViewPosEx							✓
GetViewPrj	✓	✓	✓		✓	✓	✓
ImportDataset	✓	✓	✓		✓	✓	✓
ImportFeatureTable		✓	✓			✓	✓
InsertDataset	✓	✓	✓	✓	✓	✓	✓
InsertFeatureCode		✓	✓			✓	✓
InsertGeomPt		✓	✓			✓	✓
InsertOverlayTheme	✓	✓	✓	✓	✓	✓	✓
InsertSchemaColumn	✓	✓	✓		✓	✓	✓
IsAscLicensed							✓
IsoRoute		✓	✓			✓	✓
JoinLines		✓	✓			✓	✓
LineTo	✓	✓	✓	✓	✓	✓	✓
LoadFeatureTable		✓	✓			✓	✓
LoadSchema	✓	✓	✓		✓	✓	✓
LoadSwd				✓	✓	✓	✓
LoadTheme	✓	✓	✓	✓	✓	✓	✓
LocusIntersect		✓	✓			✓	✓
MeasureAzimuth		✓	✓			✓	✓
MeasureGreatCircle		✓	✓			✓	✓
MeasureRoute		✓	✓			✓	✓
Message	✓	✓	✓				
MetreFromStr	✓	✓	✓		✓	✓	
MoveAxes		✓	✓		✓	✓	
MoveList	✓	✓	✓		✓	✓	✓
MoveTo	✓	✓	✓	✓	✓	✓	✓
MultiRoute		✓	✓			✓	✓
NolCatalog	✓	✓	✓	✓	✓	✓	✓
NolClose	✓	✓	✓	✓	✓	✓	✓
NolCompact	✓	✓	✓		✓	✓	✓
NolCreate		✓	✓			✓	✓
NolInsert	✓	✓	✓	✓	✓	✓	✓
NolOwn	✓	✓	✓		✓	✓	✓
NolSave	✓	✓	✓		✓	✓	✓
OpenClosestItem	✓	✓	✓	✓	✓	✓	✓
OpenDatasetItem	✓	✓	✓		✓	✓	✓
OpenExistingDatasetItem	✓	✓	✓		✓	✓	✓
OpenFormulaItem	✓	✓	✓	✓	✓	✓	✓
OpenItem	✓	✓	✓	✓	✓	✓	✓
OpenList	✓	✓	✓	✓	✓	✓	✓
OpenOverlayItem	✓	✓	✓		✓	✓	✓
OpenSel	✓	✓	✓	✓	✓	✓	
OwnDataset	✓	✓	✓		✓	✓	✓

Method	MM	ME	MD	OV	OM	OD	ASC
Paste	✓	✓	✓	✓	✓	✓	
PasteFrom	✓	✓	✓	✓	✓	✓	
PhotoGrid	✓	✓	✓		✓	✓	✓
PlaceGroup	✓	✓	✓		✓	✓	✓
PlacePrintTemplate	✓	✓	✓	✓	✓	✓	✓
Prompt	✓	✓	✓				
RecallNolItem		✓	✓			✓	✓
RecallNolView	✓	✓	✓		✓	✓	✓
Redraw	✓	✓	✓	✓	✓	✓	
RedrawExtent	✓	✓	✓		✓	✓	
RefreshDataset	✓	✓	✓	✓	✓	✓	✓
RefreshDbTable	✓	✓	✓		✓	✓	✓
RegisterGroupType	✓	✓	✓		✓	✓	✓
RegisterTrigger	✓	✓	✓				
Release	✓	✓	✓				
ReleaseNA	✓	✓	✓				
RemoveAtt	✓	✓	✓		✓	✓	✓
RemoveCommand	✓	✓	✓		✓	✓	
RemoveFeatureCode		✓	✓			✓	✓
RemoveOverlay	✓	✓	✓	✓	✓	✓	✓
RemoveOverlayTheme	✓	✓	✓	✓	✓	✓	✓
RemoveProperty		✓	✓			✓	✓
RemoveSchemaColumn	✓	✓	✓		✓	✓	✓
Render							✓
ReorderOverlay	✓	✓	✓	✓	✓	✓	✓
RubberSheetRaster		✓	✓			✓	✓
SaveBitmap	✓	✓	✓		✓	✓	✓
SaveDataset	✓	✓	✓		✓	✓	✓
SaveSwd				✓	✓	✓	
Scan	✓	✓	✓	✓	✓	✓	✓
ScanDataset	✓	✓	✓	✓	✓	✓	✓
ScanGeometry	✓	✓	✓		✓	✓	✓
ScanList	✓	✓	✓	✓	✓	✓	✓
ScanOverlay	✓	✓	✓	✓	✓	✓	✓
ScanPointContainers		✓	✓			✓	✓
ScrollView	✓	✓	✓	✓	✓	✓	
SelectAll	✓	✓	✓	✓	✓	✓	
SelectItem	✓	✓	✓	✓	✓	✓	
SelectList	✓	✓	✓	✓	✓	✓	
SendPrint	✓	✓	✓	✓	✓	✓	✓
Set3DView			✓			✓	
SetAxesAngle		✓	✓			✓	✓
SetAxesGrid		✓	✓			✓	
SetAxesNormal		✓	✓			✓	✓

Method	MM	ME	MD	OV	OM	OD	ASC
SetAxesPrj	✓	✓	✓		✓	✓	✓
SetCombinedFilterClause	✓	✓	✓		✓	✓	✓
SetCoordUnits	✓	✓	✓	✓	✓	✓	
SetDatasetPrj	✓	✓	✓		✓	✓	✓
SetDefaultPrj	✓	✓	✓	✓	✓	✓	
SetFlt	✓	✓	✓	✓	✓	✓	✓
SetForegroundWindow	✓	✓	✓				
SetGeomPt	✓	✓	✓		✓	✓	✓
SetGeomSegBulge	✓	✓	✓		✓	✓	✓
SetGridItemValue			✓			✓	✓
SetInt	✓	✓	✓	✓	✓	✓	✓
SetListFlt	✓	✓	✓		✓	✓	✓
SetListInt	✓	✓	✓		✓	✓	✓
SetListStr	✓	✓	✓		✓	✓	✓
SetOverlayFilter	✓	✓	✓	✓	✓	✓	✓
SetOverlayLocus		✓	✓	✓	✓	✓	✓
SetOverlaySchema	✓	✓	✓		✓	✓	✓
SetPhotoWorldCentre	✓	✓	✓		✓	✓	✓
SetRubberTransform		✓	✓			✓	✓
SetStr	✓	✓	✓	✓	✓	✓	✓
SetStrW				✓	✓	✓	✓
SetUnits	✓	✓	✓		✓	✓	
SetViewExtent	✓	✓	✓	✓	✓	✓	✓
SetViewPrj	✓	✓	✓	✓	✓	✓	✓
SetupLink	✓	✓	✓				
ShowMenu	✓	✓	✓				
SimplifyGeom		✓	✓			✓	✓
Snap2d	✓	✓	✓		✓	✓	✓
SplitExtent	✓	✓	✓	✓	✓	✓	✓
SplitPos	✓	✓	✓	✓	✓	✓	✓
StoreAsArea	✓	✓	✓		✓	✓	✓
StoreAsLine	✓	✓	✓		✓	✓	✓
StoreFeatureTable		✓	✓			✓	✓
StoreSchema	✓	✓	✓		✓	✓	✓
StoreTheme	✓	✓	✓	✓	✓	✓	✓
StrFromMetre	✓	✓	✓		✓	✓	
SwdClose	✓	✓	✓				
SwdNew	✓	✓	✓				
SwdNewWindow	✓	✓	✓				
SwdNewWindow3D			✓				
SwdNewWindowTable	✓	✓	✓				
SwdOpen	✓	✓	✓				
SwdSave	✓	✓	✓				
SwdSaveAs	✓	✓	✓				

Method	MM	ME	MD	OV	OM	OD	ASC
SwitchCommand	✓	✓	✓				
TableNewWindow	✓	✓	✓				
Takeover	✓	✓	✓				
TickCommand	✓	✓	✓		✓	✓	
TopoClean		✓	✓			✓	✓
TopoCombineNamedSeeds		✓	✓			✓	✓
TopoConvertToArea		✓	✓			✓	✓
TopoConvertToChain		✓	✓			✓	✓
TopoConvertToLine		✓	✓			✓	✓
TopoConvertToPolygon		✓	✓			✓	✓
TopoCreateArea		✓	✓			✓	✓
TopoCreateBoolean		✓	✓			✓	✓
TopoCreateChain		✓	✓			✓	✓
TopoCreateEmptyNamedSeed		✓	✓			✓	✓
TopoCreateLine		✓	✓			✓	✓
TopoCreateLink		✓	✓			✓	✓
TopoCreateNamedSeed		✓	✓			✓	✓
TopoCreateNode		✓	✓			✓	✓
TopoCreatePolygon		✓	✓			✓	✓
TopoDeleteLink		✓	✓			✓	✓
TopoDeleteNamedSeed		✓	✓			✓	✓
TopoDeleteNode		✓	✓			✓	✓
TopoDeleteSeed		✓	✓			✓	✓
TopoEdgeFill		✓	✓			✓	✓
TopoFindRoute		✓	✓			✓	✓
TopoFloodFill		✓	✓			✓	✓
TopoGetLinkNode		✓	✓			✓	✓
TopoGetLinkNumSeed		✓	✓			✓	✓
TopoGetLinkSeed		✓	✓			✓	✓
TopoGetNamedSeedDataset		✓	✓			✓	✓
TopoGetNamedSeedLoopLink		✓	✓			✓	✓
TopoGetNamedSeedLoopSize		✓	✓			✓	✓
TopoGetNamedSeedNumLoop		✓	✓			✓	✓
TopoGetNodeLink		✓	✓			✓	✓
TopoGetNodeNumLink		✓	✓			✓	✓
TopoGrowNamedSeed		✓	✓			✓	✓
TopoIsChain		✓	✓			✓	✓
TopoIsPolygon		✓	✓			✓	✓
TopoMoveNode		✓	✓			✓	✓
TopoReverseSeed		✓	✓			✓	✓
TopoShrinkNamedSeed		✓	✓			✓	✓
TraceGeom		✓	✓			✓	✓
UpdateItem	✓	✓	✓		✓	✓	✓
UpdateWorkspaceWindow	✓	✓	✓				

Method	MM	ME	MD	OV	OM	OD	ASC
`WndArrangeIcons`	✓	✓	✓				
`WndCascade`	✓	✓	✓				
`WndTile`	✓	✓	✓				
`WndTileHorizontal`	✓	✓	✓				
`WorkspaceClose`	✓	✓	✓				
`WorkspaceNew`	✓	✓	✓				
`WorkspaceOpen`	✓	✓	✓				
`WorkspaceSave`	✓	✓	✓				
`ZoomExtent`	✓	✓	✓	✓	✓	✓	✓
`ZoomView`	✓	✓	✓	✓	✓	✓	✓

Appendix 2

Method summaries

- Introduction 309
- Application control methods (GisLink) 310
- Application state methods (GisLink) 310
- Blob methods 311
- Boolean methods 311
- Cadcorp SIS Active Server Component methods 312
- Cadcorp SIS Control methods 312
- Clipboard methods 312
- Command methods (Cadcorp SIS Control) 312
- Command methods (GisLink) 313
- Co-ordinate methods 313
- Database methods 314
- Dataset methods 315
- Drawing methods 316
- Feature table methods 317
- File methods 318
- Filter methods 318
- Group methods 319
- Item methods 319
- Locus methods 320
- Named list methods 321
- Named object library methods 321
- Overlay methods 323
- Print methods 324
- Property methods 325
- Querying methods 326
- Rubber sheeting methods 327
- Selection methods 327
- Spatial searching methods 327
- Table and schema methods 328
- Theme methods 328
- Topology methods 329
- Viewing methods 330

■ Introduction

This appendix lists the methods provided by Cadcorp SIS in functional groups. You can find the description of each method in ➲Chapter 7: **"Methods"**, which lists them in alphabetical order.

■ Application control methods (GisLink)

These methods control major application functionality. They offer control over saved window definition (*.swd) files, child windows within the main frame window, the visibility of the main frame menu, and control over program exit.

They are available only when using GisLink and Cadcorp SIS applications, not in the Cadcorp SIS Control or Cadcorp SIS Active Server Component.

`ActivateWnd`	activate a window by its number, making its SWD current
`Exit`	exit the session
`GetNumSwd`	get the number of different SWD files open
`GetNumWnd`	get the number of windows open
`Message`	show a message in the status bar of the main frame window
`Prompt`	set the prompt to show in the status bar of the main frame window
`ShowMenu`	set the visibility of the main menu
`SwdClose`	close all of the windows of the current SWD
`SwdNew`	create a new, empty SWD
`SwdNewWindow`	create a new window onto an existing SWD, with the given view
`SwdNewWindow3D`	create a new 3D window onto an existing SWD
`SwdNewWindowTable`	create a new Table window onto an existing SWD
`SwdOpen`	open an existing SWD file
`SwdSave`	save the current SWD
`SwdSaveAs`	rename and save the current SWD
`SwdNewFromSwt`	create a new SWD from a saved window template
`SwdSaveAsSwt`	save the current SWD as a saved window template file
`SwtNew`	create a new, empty saved window template
`SwtOpen`	open an existing saved window template file
`SwtClose`	close all the windows of the current SWT
`SwtSave`	save the current saved window template
`SwtSaveAs`	save the current saved window template with a different name
`UpdateWorkspaceWindow`	update the current SWD in the Workspace window
`WndArrangeIcons`	arrange any iconised windows in the main frame window
`WndCascade`	cascade any non-iconised windows in the main frame window
`WndTile`	vertically tile any non-iconised windows in the main frame window.
`WndTileHorizontal`	horizontally tile any non-iconised windows in the main frame window
`WorkspaceClose`	save and close the current workspace file
`WorkspaceNew`	create a new workspace file
`WorkspaceOpen`	open an existing workspace file
`WorkspaceSave`	save the current workspace file

■ Application state methods (GisLink)

These methods control the interaction between a GisLink customisation and a Cadcorp SIS application.

They are available only when using GisLink and Cadcorp SIS applications, not in the Cadcorp SIS Control or Cadcorp SIS Active Server Component.

`Release`	return control to the application
`ReleaseNA`	return control to the application, without activating it
`SetForegroundWindow`	make a Microsoft Visual Basic form the foreground window
`SetupLink`	make a connection from a GisLink customisation
`Takeover`	take over control from the application

■ Blob methods

These methods allow the customiser/application writer to access and use Cadcorp SIS items as Blob strings. The term Blob is used because many databases use Binary Large Objects to store strings over a certain length.

Blob strings provide a database-independent way of storing Cadcorp SIS items in a database. A Blob string is a string which completely encapsulates a Cadcorp SIS item, ie its geometry and properties. Blob strings in Cadcorp SIS format are not human readable and can be interpreted only by Cadcorp SIS.

A companion string to a Blob string is a spatial reference. A spatial reference is a sixteen character string which has encoded in it a position and a radius which together describe an extents circle. The spatial reference allows Cadcorp SIS to load only those Blob items whose extents overlap the view extents when viewing Blob items from a database, thus reducing database queries.

Spatial references can also be used in databases which contain X and Y columns and are viewed as point or text items. To populate an existing database of points with spatial references, first add a suitable column to the database (ie a column of 16 character strings). A GisLink customisation or Cadcorp SIS Control application can then be written to create a temporary point item for each row in the database, get its spatial reference, and write the spatial reference to the new column, before deleting the point item. The customisation/application could be written to be run as a batch job.

`CreateItem`	create an item from a Blob string
`CreateItemB`	create an item from a Blob data
`GetBlob`	get the Blob string of the current open item within a projection
`GetBlobB`	get a Blob string of the current open item, within a projection
`GetBlobExtent`	get the extents of a Blob string, within a projection
`GetOverlayThemeLegend`	get an overlay theme legend as a Blob string within a projection
`GetSpatialReference`	get the spatial reference for the current open item within a spanned cube in a projection
`GetSpatialReferenceFromExtent`	get the spatial reference for an extent within a spanned cube in a projection

■ Boolean methods

These methods allow Boolean operations to be carried out on all items.

`CreateBoolean`	create an area item by combining existing area items
`CreateDoubleBoolean`	execute a combination of Boolean operations

All Boolean operations can be carried out between all items. The results of the Boolean operation depend on the original items and on the Boolean operation used.

■ Cadcorp SIS Active Server Component methods

These methods are specific to the Cadcorp SIS Active Server Component and are thus not available in Cadcorp SIS applications or the Cadcorp SIS Control.

`GetNumAscClients`	get the number of current ASC clients
`GetViewPos`	get the position in the current view from a position and size in pixels
`GetViewPosEx`	get the position in the current view from a position and size in pixels, as a comma-separated string
`IsAscLicensed`	test whether the Active Server Component is licensed
`Render`	render the current view into an HTML stream

■ Cadcorp SIS Control methods

These methods are specific to the Cadcorp SIS Control, and are thus not available in Cadcorp SIS applications or the Cadcorp SIS Active Server Component.

`Activate`	activate the Cadcorp SIS Control
`GetErrorString`	get the string associated with a Cadcorp SIS error code
`LoadSwd`	replace the current SWD with the contents of an SWD file
`SaveSwd`	saves the current SWD to a file

■ Clipboard methods

These methods let you copy, cut, and paste Cadcorp SIS items to the Windows clipboard.

You should paste any copied or cut items immediately, because many user actions will overwrite the contents of the Windows clipboard.

`Copy`	copy the items in a named list to the clipboard
`Paste`	paste the contents of the Windows clipboard into the current overlay

■ Command methods (Cadcorp SIS Control)

These methods allow an application written using the Cadcorp SIS Control to start system commands and to add application-specific commands to, and remove system commands from, the local (usually right-mouse button) menu.

`AddCommand`	add an application-defined command to the menu
`AllowCommands`	add or removes commands from the menu
`CanDoCommand`	check whether or not a command can be executed
`DoCommand`	execute a command
`GetCommandTick`	get the tick state on an application-defined command
`RemoveCommand`	remove an application-defined command
`TickCommand`	set or clear a tick on an application-defined command

■ Command methods (GisLink)

The following methods allow a customisation written using GisLink to start system commands and to add application-specific commands to, and remove system commands from, both the main menu and the local, right-mouse menu.

GisLink customisations are also able to monitor the progress of system commands using triggers.

`AddCommand`	add an application-defined command to the menu
`AllowCommands`	add or removes commands from the menu
`CallCommand`	call a non-interactive command in the current SWD
`CanDoCommand`	check whether or not a command can be executed
`DoCommand`	execute a command
`GetCommandTick`	get the tick state of an application-defined command
`RegisterTrigger`	register a trigger button, which will be pressed when an event occurs
`RemoveCommand`	remove an application-defined command
`SetCommandBitmap`	
`SwitchCommand`	queue a command for the current SWD, and return immediately
`TickCommand`	set or clear a tick on an application-defined command

■ Co-ordinate methods

These methods offer control over the projection of the co-ordinate system, axes, and view.

Each map window (or Cadcorp SIS Control) remembers its own co-ordinate system.

When the user is running a command inside a map window, they must enter co-ordinates using the window's co-ordinate system, with the preferred units. The user can change their preferred units with the **Tools>Preferences** command.

The Cadcorp SIS Control programmer must also use the window's co-ordinate system, but the units are always metres, in Cartesian mode, or degrees, in Spherical mode. Any co-ordinates returned by API methods, or co-ordinates passed as an argument, will either be in metres or degrees, depending on which co-ordinate system is active. (Z co-ordinates are metres for both Cartesian and Spherical modes.)

`ChangePrjUnits`	copy a named Transverse Mercator projection, changing the units
`DefineNolDatum`	define a named geoid datum, using the standard seven Bursa-Wolf parameters to modify WGS 84
`DefineNolPrjLatLon`	define a named (Latitude,Longitude) projection
`DefineNolPrjTm`	define a named Transverse Mercator projection
`DigitiserSnap`	send a digitised position into the current callback command
`GetAxesAngle`	get the angle of the current axes
`GetAxesFromLatLonHgt`	get the Cartesian axes (x,y,z) position from latitude, longitude and height above sea-level
`GetAxesPrj`	get a copy of the axes projection
`GetAxesType`	get the type of the current axes (ie Cartesian or Spherical)
`GetCoordExtent`	get the extents corresponding to a co-ordinate format string
`GetCoordString`	get the string representation of a position

`GetDatasetPrj`	get a copy of a dataset projection
`GetGridItemValue`	get the value of the cell in the current open grid item at a position
`GetLatLonHgtFromAxes`	get the latitude, longitude and height above sea-level of a Cartesian axes (x,y,z) position
`GetPhotoWorldPos`	get the world position from a paper position within the current open photo item
`GetPos`	get a position from the user
`GetPosEx`	get a position from the user and returns the action taken
`GetViewPrj`	get a copy of the view projection
`MeasureAzimuth`	measure the azimuth between two positions.
`MeasureGreatCircle`	measure the Great Circle distance between two positions
`MetreFromStr`	get a metre dimension from a string, regardless of the units used in the string
`MoveAxes`	set the position of the Cartesian axes
`SetAxesAngle`	rotate the axes to an angle
`SetAxesGrid`	show/hide a grid of points or lines with optional snapping
`SetAxesNormal`	reset the axes to the origin and orientation of the underlying projection
`SetAxesPrj`	set the projection used by axes
`SetCoordUnits`	set the preferred angle, linear, area or volume units used in the user interface
`SetDatasetPrj`	set a dataset projection
`SetDefaultPrj`	set the default viewing and co-ordinate system projections
`SetPhotoWorldCentre`	set the centre of the view within the current open photo item
`SetUnits`	set the preferred units used in the user interface
`SetViewPrj`	set the projection of the current window's view
`SimplifyGeom`	simplify the geometry of the current open item
`Snap2D`	simulate a 2D snap, making the snapped item current, and returning the snapped position
`SplitExtent`	split a comma-delimited extents string into numbers
`SplitPos`	split a comma-delimited position string into numbers
`StrFromMetre`	format a metre dimension as a string in a chosen format

■ Database methods

These methods control functionality for using Cadcorp SIS in conjunction with databases.

Cadcorp SIS can use database tables in several ways:

- point or text items can be created from a table which has suitable X and Y co-ordinate columns
- items can be stored in a table as Blob strings and viewed as a dataset
- it can store a complete dataset in a table with item level locking, and existing items can be linked to rows of relational data in a table

`CreateDbBlobOverlay`	create an overlay which views Blobs stored in a data-

	base
`CreateDbOverlay`	create an overlay which stores editable Blobs in a database
`CreateDbPointOverlay`	create an overlay which views points stored in a database
`CreateDbTable`	create a named table which views data from a database
`CreateItem`	create an item from a Blob string
`CreateItemB`	create an Item from Blob data
`CreateOpenGisSqlOverlay`	create an overlay using an OpenGIS conformant database.
`DefineRecordset`	define a named recordset, for use with databases
`DeleteRecordset`	delete a named recordset
`GetBlob`	get the Blob string of the current open item within a projection
`GetOverlayThemeLegend`	get an overlay theme legend as a Blob string within a projection
`GetSpatialReference`	get the spatial reference for the current open item within a spanned cube in a projection
`GetSpatialReferenceFromExtent`	get the spatial reference for an extent within a spanned cube in a projection
`RefreshDbTable`	refresh a named table from its database

■ Dataset methods

➲Chapter 8: **"Examples", Overlays**, page 241

These methods offer control over datasets which are file-based. A file-based dataset is a file which contains graphical items. File-based datasets must be inserted or imported into an overlay to see the contents.

`CloseDataset`	close a dataset
`CloseIndexDatasetTile`	close a named dataset tile within an index dataset
`CompactDataset`	discard all undo actions and defragments the memory used by a dataset
`CreateBds`	create an empty BDS file
`EnsureOpenWithin`	force datasets in the current SWD to open any items within the given extents, at the given scale
`FindExternalDataset`	get the serial number of a dataset, which is already open
`GetDatasetContainer`	get the serial number of the dataset which contains the specified dataset, eg in an index dataset
`GetDatasetExtent`	get the extents of all of the items in a dataset
`GetDatasetPrj`	get a copy of a dataset projection
`ImportDataset`	import a dataset into the current SWD
`InsertDataset`	insert a dataset into the current SWD
`OwnDataset`	set the ownership of a dataset
`RefreshDataset`	make sure a dataset is up to date
`SaveDataset`	save a dataset if it has been modified
`SetDatasetPrj`	set a dataset projection

■ Drawing methods

These methods create graphics. In many cases it will be easier and more efficient to use the `CreateItem` method with an OpenGIS Well-Known-Text string, rather than a stream of `MoveTo`/`LineTo` calls.

➲Chapter 8: **"Examples", Graphics**, page 230

➲Chapter 8: **"Examples", Text**, page 235

`BezierTo`	draw a Bezier curve from the current drawing position
`BulgeTo`	draw an arc from the current drawing position
`CleanLines`	clean up line items, removing repeated vertices etc
`CreateAreaFromLines`	create an area, or areas, from the line items in a named list
`CreateAssembly`	create an assembly item from the items in a named list
`CreateBitmap`	create a bitmap item
`CreateBlock`	create a named block from the items in a named list
`CreateBoundary`	create an item from the boundary of the current open item
`CreateBoxLabel`	create a special box label text item, with a line pointing to a labelled location
`CreateBoxText`	create a box text item
`CreateCircle`	create a circular area item
`CreateConvexHull`	create the smallest possible item with convex geometry, which contains the current open item
`CreateEllipse`	create an elliptical area item
`CreateExtrusion`	create a surface item by extruding the current open area or line item
`CreateFormulaGrid`	create a grid item by combining named grid items using a formula
`CreateGraticule`	create a graticule item using the current open photo item
`CreateGridFromQZone`	create a grid item from the current open QZone item
`CreateInsert`	create an insertion of a block item
`CreateLineText`	create a line text item using the current open line item
`CreateNorthPoint`	create a north point item using the current open photo item
`CreatePhoto`	create a photo item in the current window, filling it with the composed window
`CreatePoint`	create a point item
`CreateQZoneFromGrid`	create a QZone item from the cells in the current open grid item, which are between two values
`CreateRectangle`	create a rectangular area item
`CreateScatterGrid`	create a grid item from the hook points of the items in a named list
`CreateSurface`	create a surface item from the current open area item
`CreateText`	create a point text item
`CreateThiessen`	create Thiessen area items from the hook points of the items in a named list
`CreateTin`	create a TIN from the hook points of the items in a named list
`FacetGeometry`	replace curved geometry segments with shorter straight segments
`InsertGeomPt`	insert a new vertex into geometry from the current

	open item
`JoinLines`	join line items within a tolerance
`LineTo`	draw a line from the current drawing position
`MoveTo`	set the current drawing position
`PasteFrom`	paste a file into the current SWD
`SetGeomPt`	set the position of a vertex in geometry from the current open item
`SetGeomSegBulge`	set the bulge of a segment in geometry from the current open item
`SetGridItemValue`	set the value in a cell of the current open grid item
`SnipGeometry`	snip away portions of the items inside or outside the current item
`StoreAsArea`	store the previous `MoveTo`/`LineTo` operations as an area item
`StoreAsLine`	store the previous `MoveTo`/`LineTo` operations as a line item
`TraceGeom`	create a line item by tracing geometry from the current open item

■ Feature table methods

These methods allow creation and editing of named feature table objects.

Cadcorp SIS uses named feature table objects as a fast and efficient way of controlling the display, eg brush, pen, shape, of items. Named feature table objects consist of a hierarchy of feature codes, each of which has styling information which graphical items with the feature code will use when they are drawn. The feature codes hierarchy allows features of a similar type, eg Motorways, A-Roads, B-Roads, to be grouped together for fast switching on and off.

`ChangeFeatureFilter`	modify feature code information in a named feature filter
`CopyFeatureCode`	copy an existing feature code into the currently loaded feature table
`CreateFeatureFilter`	create a named feature filter based on a named feature table
`ExportFeatureTable`	export a named feature table to a comma-separated file
`GetFeatureFilterBranches`	get the feature codes branching from a parent feature code in a named feature filter
`GetFeatureTableBranches`	get the feature codes branching from a parent feature code in the currently loaded feature table
`ImportFeatureTable`	import a named feature table from a comma-separated file
`InsertFeatureCode`	insert a new feature code into the currently loaded feature table
`LoadFeatureTable`	load a named feature table for editing
`RemoveFeatureCode`	remove an existing feature code from the currently loaded feature table
`StoreFeatureTable`	store the currently open feature table

■ File methods

These methods allow applications to load and save SWD files which are fully compatible with all Cadcorp SIS applications.

Some of these methods are only available in the Cadcorp SIS Control, not using GisLink and Cadcorp SIS applications.

`Export`	export data using a plug-in exporter
`ExportBds`	export the current view to a BDS file
`ExportBmp`	export the current view to a Windows bitmap (BMP) file
`ExportEcw`	export the current view to a ECW file
`ExportPdf`	export the current view to an Adobe Acrobat (PDF) file
`ExportJpeg`	export the current view to a JPEG file
`ExportPng`	export the current view to a PNG file
`ExportVrml`	export the current view to a VRML file
`ExportWmf`	export the current view to a Windows Metafile (WMF) file
`GetNumSwd`	get the number of different SWD files opened
`ImportDataset`	import a dataset into the current SWD
`LoadSwd`	replace the current SWD with the contents of an SWD file
`SaveSwd`	save the current SWD to a file
`SwdClose`	close all the windows of the current SWD
`SwdNew`	create a new, empty SWD
`SwdNewWindow3d`	create a new 3D window onto an existing SWD
`SwdNewFromSwt`	create a new SWD from a saved window template
`SwdNewWindow`	create a new window onto an existing SWD, with the given view
`SwdOpen`	open an existing SWD file
`SwdSave`	save the current open SWD
`SwdSaveAs`	rename and saves the current SWD
`SwdSaveAsSwt`	save the current SWD with a different name
`SwtClose`	close all of the windows of the current SWD, using the chosen savecode
`SwtNew`	create a new, empty saved window template
`SwtOpen`	open an existing saved window template file
`SwtSave`	save the current saved window template
`SwtSaveAs`	rename and save the current saved window template

■ Filter methods

➲Chapter 8: **"Examples", Filters**, page 261

These methods control named filter objects. A named filter is an object which has rules which it uses to pass some items, but fail others.

Named filter objects can be applied to overlays to control the visibility of the items in that overlay: only items which pass are drawn, all other items become temporarily invisible. In addition, many Cadcorp SIS methods such as `Scan` accept a named filter as an argument.

There are several different types of named filter which have different rules. Named Filter objects can be saved in, and loaded from, named object libraries. Named filter objects can be created, edited, and set using the methods below.

`ChangeFeatureFilter`	modify feature code information in a named feature filter
`ChangeValueListFilter`	change the values included in a named value-list filter
`CombineFilter`	create a named filter by combining two named filters using a Boolean operation
`CreateClassTreeFilter`	create a named class tree filter
`CreateCombinedFilter`	create a named combined class/property filter
`CreateFeatureFilter`	create a named feature filter based on a named feature table
`CreateLinkFilter`	create an empty named link filter
`CreatePropertyFilter`	create a named property filter
`CreateValueListFilter`	create an empty named value-list filter
`GetFeatureFilterBranches`	get the feature codes branching from a parent feature code in a named feature filter
`GetOverlayFilter`	get a copy of an overlay drawing filter
`SetCombinedFilterClause`	add a clause to a named combined class/property filter
`SetOverlayFilter`	apply a copy of a named filter to an overlay in the current SWD

■ Group methods

➲Chapter 8: **"Examples", Groups**, page 267

A group is a collection of items, which can be manipulated only as a single unit by the user. Users can explode a group to access its component items, but it cannot be re-assembled.

Typically, a GisLink customisation or Cadcorp SIS Control application would use group items to prevent the user from directly editing a collection of items. An example of a good use of group items would be parameterised shapes: the customisation or application could store the parameters of the shape as attributes of the group, and then provide the user with a dialog to edit those attributes. After the user has used this dialog to modify the parameters, the customisation or application could regenerate the geometry of the group, confident that no intermediate geometrical edits could have been done directly by the user.

`CreateGroup`	create an empty group item, using a previously registered group class
`CreateGroupFromItems`	create a group item from the items in a named list
`EmptyGroup`	empty the current open group item
`PlaceGroup`	place the current open group item, leaving it open
`RegisterGroupType`	register a sub-class of a group, which the user cannot directly modify

■ Item methods

These methods control the current open item. Many methods, such as `GetFlt`/`GetInt`/`GetStr`, `SetFlt`/`SetInt`/`SetStr`, `GetDataset`, `GetGeomPt`, and so on, operate on the current open item. Only one item can be open at any time.

The current open item is independent of selected items, although selected items may be opened.

`CloseItem`	close the current open item, stopping it being current
`DeleteItem`	delete the current open item
`GetDataset`	get the serial number of the current open item's dataset
`OpenClosestItem`	open the item closest to a 3D position, within a specified search radius
`OpenDatasetItem`	open the item in the named dataset with the given ID number
`OpenExistingDatasetItem`	open an item from an existing dataset with a given ID number
`OpenFormulaItem`	open an item within a dataset which matches a formula
`OpenItem`	open the item in the current dataset with the given ID number
`OpenOverlayItem`	open the item on an overlay with the given ID number
`Scan`	scan for items, storing any found in a named list
`ScanOverlay`	scan an overlay for items, storing any found in a named list
`UpdateItem`	update the current open item, leaving it current

■ Locus methods

These methods control named locus objects. A named locus is an object which includes or excludes items based on their position and geometry.

Named locus objects can be applied to overlays to control the visibility of the items in that overlay: only items which pass are drawn, all other items become temporarily invisible. In addition, many Cadcorp SIS methods such as `Scan` accept a named locus as an argument.

Named locus objects can be saved in, and loaded from, named object libraries, and they can be created, edited, and set using the methods below.

`ChangeLocusTestMode`	modify the testing mode of a named locus
`CombineLocus`	create a named locus by combining two named loci using a Boolean operation
`CreateBufferFromItems`	create an area or QZone item surrounding the items in named list
`CreateBufferLocusFromItems`	create a named buffer locus around items in a named list
`CreateCircleLocus`	create a named circular locus
`CreateItemFromLocus`	create an item from a named locus
`CreateLocusFromItem`	create a named locus from the current open item
`CreateRectLocus`	create a named rectangular locus
`GetOverlayLocus`	get a copy of an overlay drawing locus
`LocusIntersect`	create a named locus by intersecting two existing loci
`SetOverlayLocus`	apply a copy of a named locus to an overlay in the current SWD

■ Named list methods

➲Chapter 8: **"Examples", Named lists**, page 257

Named Lists are used to both pass and return lists of items to Cadcorp SIS methods and exist only for the duration of a Cadcorp SIS session. Named lists are referred to by an application/customisation supplied textual name.

It is possible for the contents of a named list to become out of date, such as if an Item in a named list is deleted or if its dataset is removed. Any attempt to access an out of date Item will result in an error. It is therefore best to use named lists as temporary storage, and it is good practice to empty them using `EmptyList` after they have been used.

`AddToList`	add the current open item to a named list
`CombineLists`	combine two named lists using a Boolean operation
`CopyListItems`	copy the items in a named list to the default overlay
`CreateListFromOverlay`	create a named list of all of the items on an overlay
`CreateListFromSelection`	create a named list of the currently selected items
`Delete`	delete all of the items in a named list
`DrawList`	draw items in a named list with overridden styles
`EmptyList`	empty all of the items from a named list, and deletes the named list
`GetListExtent`	get the extents of all of the items in a named list
`GetListItemFlt`	get the value of a floating point property on an item in a named list
`GetListItemInt`	get the value of a long integer property on an item in a named list
`GetListItemStr`	get the value of a string property on an item in a named list
`GetListSize`	get the number of items in a named list
`MoveList`	move, rotate, and scale editable items in a named list
`OpenList`	opens an item from a named list
`Scan`	scan for items, storing any found in a named list
`ScanGeometry`	find items which satisfy a geometrical test with the current open item
`ScanList`	scan a named list for items matching a named filter and/or named locus
`ScanOverlay`	scan an overlay for items, storing any found in a named list
`SelectList`	toggle the selection status of items in a named list
`SetListFlt`	set the value of a floating point property on the items in a named list
`SetListInt`	set the value of an integer property on the items in a named list
`SetListStr`	set the value of a string property on the items in a named list

■ Named object library methods

➲Chapter 8: **"Examples", Named object libraries**, page 270

These methods offer control over the setup of named object libraries (NOLs) within a Cadcorp SIS application, Cadcorp SIS Control or Cadcorp SIS Active Server Compo-

nent. Methods are also provided for querying the contents of NOLs and creating new NOL objects.

There are three special NOLS:

- (standard), which contains all of the built-in NOL objects, and is read-only
- (temporary), which is empty at the start of a session
- (workspace)

The (workspace) NOL is available only in Cadcorp SIS applications using a project workspace (*.sis) file and will, by default, contain any new objects. In the Cadcorp SIS Control and Cadcorp SIS Active Server Component, and in Cadcorp SIS applications not using a project workspace file, the (temporary) NOL will, by default, contain any new objects.

NOL objects can be created in other NOLs either by making the target NOL the default NOL, using the workspace window's library view tab, or the DefaultNol system variable, or by specifying a 'pathname' for the NOL object, eg C:\SisNOLs\Brushes.nolRed Cross Hatch or (temporary)Red Cross Hatch. The default NOL will always be searched first when a NOL object is requested, unless a pathname is used.

NOL object names cannot contain the Tab or Escape character.

NOL object names can be made hierarchical, like a disk directory structure of folders and files, by using the . (full stop or period) character as a separator. For example, the NOL print templates A4.Landscape, A4.Portrait, A3.Landscape, and A3.Portrait will appear in two folders, A4 and A3, as Landscape and Portrait. Named objects without a full stop will appear at the root of the directory tree.

In Cadcorp SIS applications, all modified NOLs, other than (temporary), are saved at the end of the session, or when a Project Workspace file, if any, is closed. NOLs can be removed without saving using the `NolClose` method. In the Cadcorp SIS Control and Cadcorp SIS Active Server Component, NOLs are not saved, except by the `NolSave` method.

`ChangePrjUnits`	copy a named Transverse Mercator projection, changing the units
`CopyFeatureCode`	copy an existing feature code into the currently loaded feature table
`DefineNolDatum`	define a named geodetic datum, using the standard seven Bursa-Wolf parameters to modify WGS 84
`DefineNolItem`	store the current open item in a named object library
`DefineNolItemFromLocus`	store a named locus in a named object library as a named item
`DefineNolObject`	create a named object from an implicit string
`DefineNolPrintTemplate`	define a named print template from the current window contents
`DefineNolPrjLatLon`	define a named (Latitude,Longitude) projection
`DefineNolShape`	define a named shape from the items in a named list
`DefineNolView`	define a named view from the view in the current window
`DeleteNolObject`	delete a named object from a named object library
`ExportFeatureTable`	export a named feature table to a comma-separated file
`GetFeatureTableBranches`	get the feature codes branching from a parent feature code in the currently loaded feature table

`GetImplicitNolObject`	get the implicit equivalent of an object in a named object library
`GetNumNol`	get the number of named object libraries in use
`ImportFeatureTable`	import a named feature table from a comma-separated file
`InsertFeatureCode`	insert a new feature code into the currently loaded feature table
`LoadFeatureTable`	load a named feature table for editing
`NolCatalog`	list objects of a given class in all of the named object libraries
`NolClose`	close a named object library, optionally saving any changes
`NolCompact`	discard all old named object library (NOL) objects and defragments the memory used by a NOL
`NolCreate`	create an empty named object library file
`NolInsert`	insert a named object library file
`NolOwn`	set the ownership of a named object library
`NolSave`	save a named object library file
`RecallNolItem`	create an item from a named object library item
`RecallNolView`	recall a named view from a named object library
`RemoveFeatureCode`	remove an existing feature code from the currently loaded feature table
`StoreFeatureTable`	store the currently open feature table

■ Overlay methods

These methods offer control over overlays.

Many of the methods in this section use a *`pos`* argument to refer to the position of an overlay in the overlays list. The overlays list is a zero-based list, so the first overlay is at position 0, the second at position 1, and so on. Remember this when you specify the *`pos`* argument.

`CreateBackdropOverlay`	create a new overlay, which uses a named item as a backdrop
`CreateDataSourceOverlay`	insert a dataset into the current SWD, which will fetch data from a non-file data source
`CreateDbBlobOverlay`	create an overlay which views Blobs stored in a database
`CreateDbOverlay`	create an overlay, which stores editable Blobs in a database
`CreateDbPointOverlay`	create an overlay, which views points stored in a database
`CreateIndexCoverage`	create tile items covering extents, using a standard naming convention
`CreateIndexOverlay`	create an index overlay
`CreateInternalOverlay`	create an internal overlay
`CreateOpenGisSqlOverlay`	create an overlay using an OpenGIS conformant database
`CreatePhaseOverlay`	create a new phase of an existing overlay
`DefineRecordset`	define a named recordset, for use with databases
`EnsureOpenWithin`	force datasets in the current SWD to open any items within the given extents, at the given scale
`ExplodeOverlayTheme`	explode an overlay theme into a new overlay

`FindDatasetOverlay`	find an overlay, which contains the given dataset
`GetOverlayFilter`	get a copy of an overlay drawing filter
`GetOverlayLocus`	get a copy of an overlay drawing locus
`GetOverlaySchema`	get a copy of an overlay schema
`GetOverlayTheme`	get a copy of an overlay theme
`GetOverlayThemeLegend`	get an overlay theme legend as a Blob string within a projection
`ImportDataset`	import a dataset into the current SWD
`InsertDataset`	insert a dataset into the current SWD
`InsertOverlayTheme`	insert a copy of a named theme into an overlay in the current SWD
`RemoveOverlay`	remove an overlay from the current SWD, deleting it if it is an internal overlay
`RemoveOverlayTheme`	remove a theme from an overlay in the current SWD
`ReorderOverlay`	change the order of overlays
`SetOverlayFilter`	apply a copy of a named filter to an overlay in the current SWD
`SetOverlayLocus`	apply a copy of a named locus to an overlay in the current SWD
`SetOverlaySchema`	apply a copy of a schema to an overlay in the current SWD

■ Print methods

➲Chapter 8: **"Examples", Printing**, page 295

The following methods offer control over composing and printing the contents of Cadcorp SIS application and Cadcorp SIS Control windows, and Cadcorp SIS Active Server Component set-ups.

`Compose`	compose the current window, in preparation for using `PlacePrintTemplate` or `CreatePhoto` on another window
`CreateGraticule`	create a graticule item using the current open photo item
`CreateKeyMap`	create a key map item
`CreateNorthPoint`	create a north point item using the current open photo item
`CreatePhoto`	create a photo item in the current window, filling it with the composed window
`CreateScaleBar`	create a scale bar item using the current open photo item
`DefineNolPrintTemplate`	define a named print template from the current window contents
`GetOverlayThemeLegend`	get an overlay theme legend as a Blob string within a projection
`PhotoGrid`	set the default grid on the current open photo item
`PlacePrintTemplate`	place a print template in the current SWD, filling it with the composed window
`SaveBitmap`	save the current open bitmap item to a file
`SendPrint`	print the current window
`SetPhotoWorldCentre`	set the centre of the view within the current open photo item

■ Property methods

➲Chapter 8: **"Examples", Object properties**, page 253

These methods control the setting and getting of properties on various object types within Cadcorp SIS. Properties fall into two categories: members and user-defined attributes. Members are the internal, system properties of an object, eg Item ID and Pen on an item, or Number of overlays on a window. User-defined attributes are the properties added programmatically by an application or interactively by the user, eg *CostPerSqM#*. Some members can only be queried and not set, eg Item ID.

Some members have special meanings (each is returned as a space-separated list):

`_properties$`	all the properties of an object (members and attributes)
`_properties_edit$`	all the editable members of an object
`_members$`	all the members of an object
`_attributes$`	all the user-defined attributes of an object

The following object types can have their properties queried and set:

- items
- datasets
- overlays
- windows
- named object libraries (NOLs)
- feature tables
- schemas
- themes
- printing
- system variables and options

`DescribeProperty`	set the description of a property
`EvaluateFlt`	evaluate a formula, which has a floating point result
`EvaluateInt`	evaluate a formula, which has an integer result
`EvaluateStr`	evaluate a formula, which has a string result
`GetFlt`	get the value of a floating point property
`GetInt`	get the value of an integer property
`GetListItemFlt`	get the value of a floating point property on an Item in a named list
`GetListItemInt`	get the value of a long integer property on an Item in a named list
`GetListItemStr`	get the value of a string property on an Item in a named list
`GetPropertyDescription`	get the description of a property
`GetStr`	get the value of a string property
`GetStrW`	get the Unicode value of a string property
`RemoveAtt`	remove an attribute from the current open item
`RemoveProperty`	remove a property from an object
`SetFlt`	set the value of a floating point property
`SetInt`	set the value of an integer property
`SetListFlt`	set the value of a floating point property on the items

	in a named list
`SetListInt`	set the value of an integer property on the items in a named list
`SetListStr`	set the value of a string property on the items in a named list
`SetStr`	set the value of a string property
`SetStrW`	set the Unicode value of a string property

◆ Object types

The *`objectType`* argument can take the following values:

`SIS_OT_CURITEM`	the current open item
`SIS_OT_DEFITEM`	the default item
`SIS_OT_DATASET`	datasets
`SIS_OT_OVERLAY`	overlays
`SIS_OT_WINDOW`	the window
`SIS_OT_NOL`	named object libraries
`SIS_OT_FTABLE`	the current feature table
`SIS_OT_SCHEMA`	the current schema
`SIS_OT_SCHEMACOLUMN`	a column in the current schema
`SIS_OT_THEME`	the current theme
`SIS_OT_THEMECOMPONENT`	a component in the current theme
`SIS_OT_PRINTER`	printer
`SIS_OT_SYSTEM`	system variables
`SIS_OT_OPTION`	system-wide Boolean options

■ Querying methods

These methods allow the application programmer or customiser to query the geometry of items.

`GetExtent`	get the extents of the current open item
`GetGeomAngleFromLength`	get the tangent angle a specified length along geometry from the current open item
`GetGeomDim`	get dimension of geometry from the current open item
`GetGeomLength`	get length of geometry from the current open item
`GetGeomLengthUpto`	get the length along the current open item's geometry up to a position
`GetGeomNumPt`	get the number of vertices in geometry from the current open item
`GetGeomNumSeg`	get the number of segments in geometry from the current open item
`GetGeomPosFromLength`	get the position a specified length along geometry from the current open item
`GetGeomPt`	get the position of a vertex from geometry in the current open item

`GetGeomSegAxis`	get the axis of a bulged segment within geometry in the current open item
`GetGeomSegBulge`	get the bulge of a segment within geometry in the current open item
`GetGeomSegShape`	get the shape of a segment within geometry of the current open item
`GetGeomSelfIntersection`	look for a self-intersection within a single piece of geometry from the current open item
`GetGeomTgtFromLength`	get the tangent vector a specified length along the current open item's geometry
`GetHook`	get the hook point of the current open item
`GetNumGeom`	get the number of geometry pieces in the current open item

■ Rubber sheeting methods

These methods control Cadcorp SIS rubber sheeting functionality. Cadcorp SIS can create rubber sheet items from both vector and raster graphics.

`CreateBitmap`	create a bitmap item
`CreateDisplacement`	create a displacement item, prior to doing a rubber sheet operation
`CreateRubberSheet`	create a rubber sheet item from the displacement items in a named list
`RubberSheetRaster`	apply the current rubber sheet transformation to the current open bitmap item
`SetRubberTransform`	set the current rubber sheet transformation from the current open rubber sheet item

■ Selection methods

The following methods offer control over the current selection list. Care must be taken when using these methods, because the selection list is strictly outside the control of an application or customisation; the user may change the selection at any time. Use named lists to step through items without interfering with the user selection.

`DeselectAll`	clear the current selection list
`GetNumSel`	get the number of items selected in the current SWD
`OpenSel`	open an item in the current selection list
`SelectAll`	select all hittable and editable items
`SelectItem`	toggle the selection status of the current open item
`SelectList`	toggle the selection status of items in a named list

■ Spatial searching methods

➲Chapter 8: **"Examples", Spatial searches**, page 275

These methods let you perform spatial queries on Area, Polygon, and QZone items.

`Scan`	scan for items, storing any found in a named list
`ScanDataset`	scan a dataset for items, storing any found in a named list
`ScanGeometry`	find items which satisfy a geometrical test with the current open item
`ScanList`	scan a named list for items matching a named filter and/or named locus

`ScanOverlay`	scan an overlay for items, storing any found in a named list
`ScanPointContainers`	find area items which contain a point

■ Table and schema methods

These methods control named tables and allow creation and editing of named schema objects.

Cadcorp SIS uses named tables to view data from a database table, with rows in the table being linked to graphical items by item properties.

Cadcorp SIS uses named schema objects to control the display of data-oriented parts of the user interface, eg the table window.

`CreateDbTable`	create a named table, which views data from a database
`GetOverlaySchema`	get a copy of an overlay schema
`InsertSchemaColumn`	insert a new column into the currently loaded schema
`LoadSchema`	load a named schema for editing
`RefreshDbTable`	refresh a named table from its database
`RemoveSchemaColumn`	remove an existing column from the currently loaded schema
`SetOverlaySchema`	apply a copy of a schema to an overlay in the current SWD
`StoreSchema`	store the currently open schema
`SwdNewWindowTable`	create a new table window onto an existing SWD
`TableNewWindow`	create or activates a view onto a named table

■ Theme methods

➲Chapter 8: **"Examples", Themes**, page 289

These methods allow creation and editing of named theme objects.

Cadcorp SIS uses named theme objects to control the display of items (eg brush, pen, and shape), depending on item properties, and also to annotate items, eg with labels, bar charts, or pie charts.

`CopyThemeComponent`	copy an existing theme component into the currently loaded theme
`CreateBarTheme`	create a new bar chart theme
`CreateContourTheme`	create a new contour theme
`CreateDotTheme`	create a new dot density theme
`CreateExtrudeTheme`	create a new extrude theme
`CreateFlowTheme`	create a new flow theme
`CreateGraduatedTheme`	create a new graduated theme
`CreateIndividualTheme`	create a new individual theme
`CreateLabelTheme`	create a new label theme
`CreatePieTheme`	create a new pie chart theme
`CreateRangeTheme`	create a new range theme
`CreateReliefTheme`	create a new relief theme
`CreateTopoTheme`	create a new topology theme
`ExplodeOverlayTheme`	explode an overlay theme into a new overlay
`GetOverlayTheme`	get a copy of an overlay theme
`GetOverlayThemeLegend`	get an overlay theme legend as a Blob string within a

	projection
`InsertOverlayTheme`	insert a copy of a named theme into an overlay in the current SWD
`LoadTheme`	load a named theme for editing
`RemoveOverlayTheme`	remove a theme from an overlay in the current SWD
`StoreTheme`	store the currently open theme

■ Topology methods

These methods let you create and manipulate topological items.

Many of the methods use a named seed. Named seeds are transient polygon or chain items which can be used to perform detailed querying of existing polygon or chain items, or to create new polygon or chain items. Named seeds are referenced by a textual name supplied by application/customisation, in the same way as named lists. Named seeds are not visible on screen, and are not saved in the dataset they belong to. They can however be copies of visible, saved polygon or chain items.

`CreateIsoRoute`	create a multi-line item, which can be reached from a position, within a given cost
`IsoRoute`	find link and node items, which can be reached from a position, within a given cost
`MeasureRoute`	measure the best route between two positions
`MultiRoute`	measure routes between items in a named list
`TopoClean`	clean up topological link items
`TopoCombineNamedSeeds`	create a named seed object by doing a Boolean operation on existing named seed objects
`TopoConvertToArea`	convert the current open polygon item into an area item
`TopoConvertToChain`	convert the current open line item into a topological chain item
`TopoConvertToLine`	convert the current open chain item into a line item
`TopoConvertToPolygon`	convert the current open area item into a topological polygon item
`TopoCreateArea`	create an area item from the current open polygon item
`TopoCreateBoolean`	create a named seed object by doing a Boolean operation on existing polygon items
`TopoCreateChain`	create a chain item from a named seed object
`TopoCreateEmptyNamedSeed`	create a new, empty transient named seed object
`TopoCreateLine`	create a line item from the current open chain item
`TopoCreateLink`	create a topological link item, copying the geometry from the current open line item
`TopoCreateNamedSeed`	create a transient named seed object from the current open seed item
`TopoCreateNode`	create a node item, merging it in to any existing topology
`TopoCreatePolygon`	create a polygon item from a named seed object
`TopoDeleteLink`	delete the current open link item
`TopoDeleteNamedSeed`	delete a transient named seed object
`TopoDeleteNode`	delete or simplifies the current open node item
`TopoDeleteSeed`	delete the current open topological chain or polygon item
`TopoEdgeFill`	create a named seed object by following the current

	open link item to make a closed loop
`TopoFindRoute`	create a named seed object by route-finding between two node items within a dataset
`TopoFloodFill`	create a named seed object by flood-filling links within a dataset
`TopoGetLinkNode`	get the ID of a node item from the current open link item
`TopoGetLinkNumSeed`	get the number of seed items attached to the current open link item
`TopoGetLinkSeed`	get the signed ID of a polygon or chain item from the current open link item
`TopoGetNamedSeedDataset`	get the dataset with which a named seed is compatible
`TopoGetNamedSeedLoopLink`	get the ID of a link item from a named seed object
`TopoGetNamedSeedLoopSize`	get the number of links referred to by a loop in a named seed object
`TopoGetNamedSeedNumLoop`	get the number of loops in a named seed object
`TopoGetNodeLink`	get the signed ID of a link item from the current open node item
`TopoGetNodeNumLink`	get the number of link items attached to the current open node item
`TopoGrowNamedSeed`	add a link ID into a named seed object
`TopoIsChain`	test if a named seed object is a topological chain
`TopoIsPolygon`	test if a named seed object is a topological polygon
`TopoMoveNode`	move the current open node item, dragging any connected link items
`TopoReverseSeed`	reverse the current open chain or polygon item
`TopoShrinkNamedSeed`	remove a link item from a named seed object

■ Viewing methods

➲Chapter 8: **"Examples", Windows**, page 239

These methods offer control of the view in a map window or 3D window.

`CreateDrapeBitmap`	create a named bitmap item from the current view, which is suitable for draping in the 3D window
`DrapeBitmap`	drape a bitmap item, stored in a named object library, in the 3D window
`DefineNolView`	define a named view from the view in the current window
`Get3DEye`	get the position of the eye in a 3D window
`Get3DLook`	get the position looked towards in a 3D window
`GetDatasetExtent`	get the extents of all of the items in a dataset
`GetDisplayExtent`	get the padded visible extents of the current window
`GetViewExtent`	get the visible extents of the current window
`GetViewPrj`	get a copy of the view projection
`RecallNolView`	recall a named view from a named object library
`Redraw`	redraw a window or windows
`RedrawExtent`	redraw windows or part of a window
`ScrollView`	scroll the current window by a number of pixels
`Set3DView`	set the eye and look position in a 3D view
`SetDefaultPrj`	set the default viewing and co-ordinate system projections
`SetGazetteerView`	find and zoom to a location using a plug-in gazetteer

`SetViewExtent`	set the visible extents of the current window
`SetViewPrj`	set the projection of the current window's view
`ZoomExtent`	zoom the view to the extents of all of the items in all of the visible, hittable, and editable overlays
`ZoomView`	zoom the current window by a scale factor

Appendix 3

ACOM commands

- Introduction 333
- File menu 333
- Edit menu 334
- Map menu 334
- Table menu 335
- 3D menu 336
- Construct menu 336
- Item menu 337
- Alter menu 339
- Measure menu 340
- Tools menu 341
- Window menu 341
- Help menu 341
- Miscellaneous commands 341

■ Introduction

This appendix lists the ACOM commands in Cadcorp SIS. They are grouped according to their menu, apart from miscellaneous commands, which are gathered at the end of the chapter.

The hash symbol (#) indicates that the command is new.

■ File menu

	MM	ME	MD	OV	OM	OD	ASC
AComExit	✓	✓	✓				
AComExport	✓	✓	✓		✓	✓	
AComExportServer		✓	✓			✓	
AComExportTiles		✓	✓			✓	
AComFileClose	✓	✓	✓				
AcomFileNew	✓	✓	✓				
AComFileOpen	✓	✓	✓				
AComFileSave	✓	✓	✓				
AComFileSaveAll	✓	✓	✓				
AComFileSaveAs	✓	✓	✓				
AComFileSend	✓	✓	✓				
AComPrint	✓	✓	✓				
AComPrintPreview	✓	✓	✓				
AComPrintSetup	✓	✓	✓				

	MM	ME	MD	OV	OM	OD	ASC
AComPrintTemplate	✓	✓	✓				
AComPrintTemplateQuick	✓	✓	✓				
AComPrintTemplateRecall	✓	✓	✓				
AComPrintTemplateStore	✓	✓	✓		✓	✓	
AComWorkspaceClose	✓	✓	✓				
AComWorkspaceOpen	✓	✓	✓				
AComWorkspaceSave	✓	✓	✓				

■ Edit menu

AComBufferFence		✓	✓			✓	
AComCopy	✓	✓	✓	✓	✓	✓	
AComCut	✓	✓	✓	✓	✓	✓	
AComDelete	✓	✓	✓	✓	✓	✓	
AComDeselectAll	✓	✓	✓	✓	✓	✓	
AComFence	✓	✓	✓		✓	✓	
AComFenceCircle	✓	✓	✓		✓	✓	
AComFenceOffset		✓	✓			✓	
AComItemProperties	✓	✓	✓	✓	✓	✓	
AComPaste	✓	✓	✓	✓	✓	✓	
AComPasteFrom	✓	✓	✓	✓	✓	✓	
AComPasteSpecial	✓	✓	✓	✓	✓	✓	
AComRecallNamedItem		✓	✓			✓	
AComRedo	✓	✓	✓	✓	✓	✓	
AComReplicate	✓	✓	✓		✓	✓	
AComSelect	✓	✓	✓		✓	✓	
AComSelectAll	✓	✓	✓	✓	✓	✓	
AComSelectSlide	✓	✓	✓	✓	✓	✓	
AComStoreNamedItem		✓	✓			✓	
AComTableQuery	✓	✓	✓				
AComUndo	✓	✓	✓	✓	✓	✓	

■ Map menu

AComAddOverlay	✓	✓	✓	✓	✓	✓	
AComAddTheme	✓	✓	✓	✓	✓	✓	
AComAxesAngle		✓	✓			✓	
AComAxesMove		✓	✓			✓	
AComAxesNormal		✓	✓			✓	
AComAxesShow		✓	✓			✓	
AComAxesSpinX		✓	✓			✓	
AComAxesSpinY		✓	✓			✓	
AComAxesSpinZ		✓	✓			✓	
AComGazetteer	✓	✓	✓	✓	✓	✓	

	MM	ME	MD	OV	OM	OD	ASC
AComLayers	✓	✓	✓		✓	✓	
AComPan	✓	✓	✓	✓	✓	✓	
AComPanContinuous (#)	✓	✓	✓	✓	✓	✓	
AComPanDrag	✓	✓	✓	✓	✓	✓	
AComPluginGazetteer (#)	✓	✓	✓	✓	✓	✓	
AComPrj	✓	✓	✓		✓	✓	
AComRasterZoom	✓	✓	✓	✓	✓	✓	
AComRecallView	✓	✓	✓		✓	✓	
AComRecentre (#)	✓	✓	✓	✓			
AComRedraw	✓	✓	✓	✓	✓	✓	
AComRegenView	✓	✓	✓		✓	✓	
AComRoamerMode	✓	✓	✓	✓	✓	✓	
AComRotateView	✓	✓	✓		✓	✓	
AComSetCurrentOverlay	✓	✓	✓	✓	✓	✓	
AComSnapGrid		✓	✓			✓	
AComStoreView	✓	✓	✓		✓	✓	
AComViewBack (#)	✓	✓	✓	✓	✓	✓	
AComViewForward (#)	✓	✓	✓	✓	✓	✓	
AComViewHome (#)	✓	✓	✓	✓	✓	✓	
AComViewScale	✓	✓	✓		✓	✓	
AComZoomAll	✓	✓	✓	✓	✓	✓	
AComZoomExtent	✓	✓	✓	✓	✓	✓	
AComZoomIn2	✓	✓	✓	✓	✓	✓	
AComZoomModeIn	✓	✓	✓	✓	✓	✓	
AComZoomModeOut	✓	✓	✓	✓	✓	✓	
AComZoomOut	✓	✓	✓	✓	✓	✓	
AComZoomSelect	✓	✓	✓	✓	✓	✓	
AComZoomToScale	✓	✓	✓	✓	✓	✓	

■ Table menu

AComTableFillColumn	✓	✓	✓
AComTableFitToHeader	✓	✓	✓
AComTableFitToWindow	✓	✓	✓
AComTableJoinOverlay	✓	✓	✓
AComTableQuery	✓	✓	✓
AComTableRefill	✓	✓	✓
AComTableScroll	✓	✓	✓
AComTableSort	✓	✓	✓
AComTableSortAscending	✓	✓	✓
AComTableSortDescending	✓	✓	✓
AComTableStatistics	✓	✓	✓
AComTableZoom	✓	✓	✓

■ 3D menu

	MM	ME	MD	OV	OM	OD	ASC
AComCreateDrapeBitmap			✓			✓	
AComOglDetails	✓	✓	✓		✓	✓	
AComOglDrape			✓			✓	
AComOglDrawX	✓	✓	✓		✓	✓	
AComOglExaggerateZ	✓	✓	✓		✓	✓	
AComOglModeCruise	✓	✓	✓		✓	✓	
AComOglModeEye	✓	✓	✓		✓	✓	
AComOglModeModel	✓	✓	✓		✓	✓	
AComOglModePan	✓	✓	✓		✓	✓	
AComOglModeZoom	✓	✓	✓		✓	✓	
AComOglReset	✓	✓	✓		✓	✓	

■ Construct menu

AComArc3P		✓	✓			✓	
AComArcACP		✓	✓			✓	
AComArcAPP		✓	✓			✓	
AComArcCPP		✓	✓			✓	
AComArcRCPP		✓	✓			✓	
AComAreaEx	✓	✓	✓	✓	✓	✓	
AComAssembly		✓	✓			✓	
AComBezier		✓	✓			✓	
AComBlock		✓	✓			✓	
AComBoxLabel	✓	✓	✓		✓	✓	
AComBoxText	✓	✓	✓		✓	✓	
AComBufferZone		✓	✓			✓	
AComCircleP2		✓	✓			✓	
AComCircleP3		✓	✓			✓	
AComCirclePC		✓	✓			✓	
AComCircleRC		✓	✓			✓	
AComCircleRPP		✓	✓			✓	
AComCookieCut		✓	✓			✓	
AComCreateChain		✓	✓			✓	
AComCreateFormulaGrid			✓			✓	
AComCreateLink		✓	✓			✓	
AComCreateNode		✓	✓			✓	
AComCreateScatterGrid			✓			✓	
AComDefineShape		✓	✓			✓	
AComDimChain		✓	✓			✓	
AComDimDatum		✓	✓			✓	
AComDimDistance		✓	✓			✓	
AComDimRun		✓	✓			✓	
AComDisplace		✓	✓			✓	

	MM	ME	MD	OV	OM	OD	ASC
AComDivide		✓	✓			✓	
AComDividePath		✓	✓			✓	
AComEllipse	✓	✓	✓	✓	✓	✓	
AComExtrude			✓			✓	
AComFreeHand	✓	✓	✓		✓	✓	
AComInsert		✓	✓			✓	
AComIsoRoute		✓	✓			✓	
AComLineEx	✓	✓	✓	✓	✓	✓	
AComLineText		✓	✓			✓	
AComLocusFromItem		✓	✓			✓	
AComMakePolygon		✓	✓			✓	
AComPathArray		✓	✓			✓	
AComPhoto		✓	✓				
AComPoint	✓	✓	✓	✓	✓	✓	
AComPolarArray		✓	✓			✓	
AComQZone		✓	✓			✓	
AComRect	✓	✓	✓	✓	✓	✓	
AComRectArray		✓	✓			✓	
AComRuleAngle		✓	✓			✓	
AComSpagArea		✓	✓			✓	
AComSpagLine		✓	✓			✓	
AComSpagPoint		✓	✓			✓	
AComText	✓	✓	✓	✓	✓	✓	
AComThiessen			✓			✓	
AComTin			✓			✓	
AComTinFromPoints			✓			✓	

■ Item menu

AComAdjacentSeed		✓	✓			✓	
AComAlignText		✓	✓			✓	
AComAreaSetSeed		✓	✓			✓	
AComAreaToTopology		✓	✓			✓	
AComAssemblyEdit		✓	✓			✓	
AComBitmapCompress		✓	✓			✓	
AComBoxToText		✓	✓			✓	
AComConnectLink		✓	✓			✓	
AComConvertToGray		✓	✓			✓	
AComConvertTopology		✓	✓			✓	
AComCreateGridShadow			✓			✓	
AComCreateQZoneGrid			✓			✓	
AComDividePolygon		✓	✓			✓	
AComExplode		✓	✓			✓	
AComExplodeGroup		✓	✓			✓	
AComExplodeShape		✓	✓			✓	

	MM	ME	MD	OV	OM	OD	ASC
AComFillPhoto		✓	✓			✓	
AComGraticule		✓	✓			✓	
AComGraticuleStyleRecall		✓	✓			✓	
AComGraticuleStyleStore		✓	✓			✓	
AComGridZoneRange			✓			✓	
AComInsertStar		✓	✓			✓	
AComJoinLines		✓	✓			✓	
AComJunction		✓	✓			✓	
AComKeyMap		✓	✓			✓	
AComLegend		✓	✓			✓	
AComLineAppend		✓	✓			✓	
AComLineToLink		✓	✓			✓	
AComLinkAdjacent		✓	✓			✓	
AComMakeArea		✓	✓			✓	
AComMakeRubber		✓	✓			✓	
AComMeasVolume			✓			✓	
AComMultiAreaToTopology		✓	✓			✓	
AComNodeAdjacent		✓	✓			✓	
AComNorthPoint		✓	✓			✓	
AComPanPhoto		✓	✓			✓	
AComPhotoGridOverlay		✓	✓			✓	
AComRotatePhoto		✓	✓			✓	
AComRubberExplode		✓	✓			✓	
AComRubberSetCurrent		✓	✓			✓	
AComSaveBitmap		✓	✓			✓	
AComScaleBar		✓	✓			✓	
AComSeedToArea		✓	✓			✓	
AComSeedToLine		✓	✓			✓	
AComSelectAssembly		✓	✓			✓	
AComSelectClass (Area)		✓	✓			✓	
AComSelectClass (Assembly)		✓	✓			✓	
AComSelectClass (Bitmap)		✓	✓			✓	
AComSelectClass (Box Text)		✓	✓			✓	
AComSelectClass (Chain)		✓	✓			✓	
AComSelectClass (Dimension)		✓	✓			✓	
AComSelectClass (Displacement)		✓	✓			✓	
AComSelectClass (Graticule)		✓	✓			✓	
AComSelectClass (Grid)		✓	✓			✓	
AComSelectClass (Group)		✓	✓			✓	
AComSelectClass (Insert)		✓	✓			✓	
AComSelectClass (KeyMap)		✓	✓			✓	
AComSelectClass (Label)		✓	✓			✓	
AComSelectClass (Line Text)		✓	✓			✓	
AComSelectClass (Line)		✓	✓			✓	

	MM	ME	MD	OV	OM	OD	ASC
`AComSelectClass` (Link)		✓	✓			✓	
`AComSelectClass` (Meta-File)		✓	✓			✓	
`AComSelectClass` (Multi-Area)		✓	✓			✓	
`AComSelectClass` (Multi-Geom)		✓	✓			✓	
`AComSelectClass` (Multi-Line)		✓	✓			✓	
`AComSelectClass` (Multi-Point)		✓	✓			✓	
`AComSelectClass` (Node)		✓	✓			✓	
`AComSelectClass` (NorthPoint)		✓	✓			✓	
`AComSelectClass` (Photo)		✓	✓			✓	
`AComSelectClass` (Plug-In)		✓	✓			✓	
`AComSelectClass` (Point)		✓	✓			✓	
`AComSelectClass` (Polygon)		✓	✓			✓	
`AComSelectClass` (QZone)			✓			✓	
`AComSelectClass` (RubberSheet)		✓	✓			✓	
`AComSelectClass` (Scalebar)		✓	✓			✓	
`AComSelectClass` (Solid)		✓	✓			✓	
`AComSelectClass` (Surface)		✓	✓			✓	
`AComSelectClass` (Text)		✓	✓			✓	
`AComSelectClass` (TIN)		✓	✓			✓	
`AComTextExplode`		✓	✓			✓	
`AComTextToBox`		✓	✓			✓	
`AComTinDrapeLines`			✓			✓	
`AComTinFromGrid`			✓			✓	
`AComTinMerge`			✓			✓	
`AComTinPerimeter`			✓			✓	
`AComTinSnip`			✓			✓	
`AComTinSubdivide`			✓			✓	
`AComTriangulate`			✓			✓	
`AComZoomPhoto`		✓	✓				

■ Alter menu

`AComAddGeometry`	✓	✓	✓		✓	✓	
`AComAlignLineToAxes`		✓	✓			✓	
`AComAlignLineToSelf`		✓	✓			✓	
`AComAlignPixels`		✓	✓			✓	
`AComBitmapCompress`		✓	✓			✓	
`AComBoundary`		✓	✓			✓	
`AComBreakLine`		✓	✓			✓	
`AComChamfer`		✓	✓			✓	
`AComCleanLine`		✓	✓			✓	
`AComCleanTopo`		✓	✓			✓	
`AComConvertToGray`		✓	✓			✓	
`AComConvexHull`		✓	✓			✓	

	MM	ME	MD	OV	OM	OD	ASC
AComDecompose	✓	✓	✓		✓	✓	
AComExclusiveOr		✓	✓			✓	
AComFacetLine		✓	✓			✓	
AComFillet		✓	✓			✓	
AComFillGeometry		✓	✓			✓	
AComIntersect		✓	✓			✓	
AComIntersectLine		✓	✓			✓	
AComJoin		✓	✓			✓	
AComLinkDeleteBad		✓	✓			✓	
AComLinkExplode		✓	✓			✓	
AComMergeNode		✓	✓			✓	
AComMove	✓	✓	✓		✓	✓	
AComMoveDts	✓	✓	✓		✓	✓	
AComNodeDeleteBad		✓	✓			✓	
AComPathDel		✓	✓			✓	
AComPathMove		✓	✓			✓	
AComRemoveGeometry	✓	✓	✓		✓	✓	
AComReprofileLine		✓	✓			✓	
AComResample		✓	✓			✓	
AComReverse	✓	✓	✓		✓	✓	
AComRotate	✓	✓	✓		✓	✓	
AComRubberSheet		✓	✓			✓	
AComSegDelete		✓	✓			✓	
AComSegMove		✓	✓			✓	
AComSetResampleMethod		✓	✓			✓	
AComSimplify		✓	✓			✓	
AComSmoothMidpoint		✓	✓			✓	
AComSmoothVertex		✓	✓			✓	
AComSnipDeleteInside		✓	✓			✓	
AComSnipDeleteOutside		✓	✓			✓	
AComStretch	✓	✓	✓		✓	✓	
AComSubtract		✓	✓			✓	
AComTrim		✓	✓			✓	
AComUnion		✓	✓			✓	

■ Measure menu

AComGridBearing	✓	✓	✓		✓	✓	
AComGridPath	✓	✓	✓		✓	✓	
AComMeasArea	✓	✓	✓		✓	✓	
AComMeasRadius		✓	✓			✓	
AComMeasureAngle	✓	✓	✓		✓	✓	
AComMeasureDist	✓	✓	✓		✓	✓	
AComMeasureFence (#)	✓	✓	✓		✓	✓	

	MM	ME	MD	OV	OM	OD	ASC
AComMeasureLen	✓	✓	✓		✓	✓	
AComMeasurePos	✓	✓	✓		✓	✓	
AComMeasureRoute		✓	✓			✓	

■ Tools menu

AComCalibrateDigitiser		✓	✓				
AComPositionBar	✓	✓	✓				
AComPreferences	✓	✓	✓	✓	✓	✓	
AComShowProgramWindow	✓	✓	✓	✓	✓	✓	
AComToolBarPref	✓	✓	✓				
AComWorkspaceWnd	✓	✓	✓				

■ Window menu

AComBottom	✓	✓	✓				
AComClose	✓	✓	✓				
AComCloseAll	✓	✓	✓				
AComOglWindow	✓	✓	✓				
AComRedrawAll	✓	✓	✓				
AComSplitHorizontal	✓	✓	✓				
AComSplitVertical	✓	✓	✓				
AComTableWindow	✓	✓	✓				
AComWindowArrangeIcons	✓	✓	✓				
AComWindowCascade	✓	✓	✓				
AComWindowNew	✓	✓	✓				
AComWindowNext	✓	✓	✓				
AComWindowTile	✓	✓	✓				
AComWindowTileHorizontal	✓	✓	✓				
AComZoomNew	✓	✓	✓				

■ Help menu

AComAbout	✓	✓	✓	✓	✓	✓	
AComHelp	✓	✓	✓				
AComWebCadcorp	✓	✓	✓				
AComWebDownload	✓	✓	✓				

■ Miscellaneous commands

AComDrawX	✓	✓	✓	✓	✓	✓	
AComFileNewSwd	✓	✓	✓				
AComOglDown	✓	✓	✓		✓	✓	
AComOglIn	✓	✓	✓		✓	✓	
AComOglLeft	✓	✓	✓		✓	✓	
AComOglOut	✓	✓	✓		✓	✓	
AComOglRight	✓	✓	✓		✓	✓	

	MM	ME	MD	OV	OM	OD	ASC
AComOglUp	✓	✓	✓		✓	✓	
AComOleExit	✓	✓	✓				
AComOleSaveAs	✓	✓	✓				
AComOleUpdate	✓	✓	✓				
AComSnapX	✓	✓	✓				
AComSnapY	✓	✓	✓				
AComSnapZ	✓	✓	✓				
AComTableNew	✓	✓	✓				
AComTablePaste	✓	✓	✓				
AComTiFilterCurrent	✓	✓	✓				
AComTiFilterHide	✓	✓	✓				
AComTiFilterShow	✓	✓	✓				
AComTiLibraryChapterCopy	✓	✓	✓				
AComTiLibraryChapterCut	✓	✓	✓				
AComTiLibraryChapterDelete	✓	✓	✓				
AComTiLibraryChapterNewFolder	✓	✓	✓				
AComTiLibraryChapterPaste	✓	✓	✓				
AComTiLibraryCurrent	✓	✓	✓				
AComTiLibraryDemote	✓	✓	✓				
AComTiLibraryDisown	✓	✓	✓				
AComTiLibraryImportFtable (#)		✓	✓			✓	
AComTiLibraryNew	✓	✓	✓				
AComTiLibraryNewObject	✓	✓	✓				
AComTiLibraryObjectProperties	✓	✓	✓				
AComTiLibraryObjectRename	✓	✓	✓				
AComTiLibraryOpen	✓	✓	✓				
AComTiLibraryOwn	✓	✓	✓				
AComTiLibraryPaste	✓	✓	✓				
AComTiLibraryPromote	✓	✓	✓				
AComTiLibraryRemove	✓	✓	✓				
AComTiLibrarySave	✓	✓	✓				
AComTiLibraryToggle	✓	✓	✓				
AComTiLibraryToggleStar (#)	✓	✓	✓			✓	
AComTiOverlayAddSchemaColumn	✓	✓	✓				
AComTiOverlayAddTheme	✓	✓	✓				
AComTiOverlayApplyFilter	✓	✓	✓				
AComTiOverlayApplyFocus	✓	✓	✓				
AComTiOverlayCopy	✓	✓	✓				
AComTiOverlayCurrent	✓	✓	✓				
AComTiOverlayCut	✓	✓	✓				
AComTiOverlayDatasetDetails	✓	✓	✓				
AComTiOverlayDelete	✓	✓	✓				
AComTiOverlayDemote	✓	✓	✓				
AComTiOverlayEditable	✓	✓	✓				

	MM	ME	MD	OV	OM	OD	ASC
AComTiOverlayHittable	✓	✓	✓				
AComTiOverlayInvisible	✓	✓	✓				
AComTiOverlayJoinTable	✓	✓	✓				
AComTiOverlayNotes	✓	✓	✓				
AComTiOverlayPasteTheme	✓	✓	✓				
AComTiOverlayPromote	✓	✓	✓				
AComTiOverlayProperties	✓	✓	✓				
AComTiOverlayQuery	✓	✓	✓				
AComTiOverlayRename	✓	✓	✓				
AComTiOverlayResetFilter	✓	✓	✓				
AComTiOverlayResetLocus	✓	✓	✓				
AComTiOverlaySelectItems	✓	✓	✓				
AComTiOverlayThemeCopy	✓	✓	✓				
AComTiOverlayThemeCut	✓	✓	✓				
AComTiOverlayThemeDelete	✓	✓	✓				
AComTiOverlayThemeDemote	✓	✓	✓				
AComTiOverlayThemeExplode	✓	✓	✓				
AComTiOverlayThemePromote	✓	✓	✓				
AComTiOverlayThemeProperties	✓	✓	✓				
AComTiOverlayThemeRename	✓	✓	✓				
AComTiOverlayThemeSave	✓	✓	✓				
AComTiOverlayThemeSaveColourset	✓	✓	✓				
AComTiOverlayThemeToggle	✓	✓	✓				
AComTiOverlayVisible	✓	✓	✓				
AComTiOverlayZoom	✓	✓	✓				
AComTiSchemaColumnDelete	✓	✓	✓				
AComTiSchemaColumnDemote	✓	✓	✓				
AComTiSchemaColumnPromote	✓	✓	✓				
AComTiSchemaColumnProperties	✓	✓	✓				
AComTiSchemaColumnRename	✓	✓	✓				
AComTiSchemaColumnToggle	✓	✓	✓				
AComTiSwdAddOverlay	✓	✓	✓				
AComTiSwdAddTheme	✓	✓	✓				
AComTiSwdClose	✓	✓	✓				
AComTiSwdLayers	✓	✓	✓				
AComTiSwdPasteOverlay	✓	✓	✓				
AComTiSwdProperties	✓	✓	✓				
AComTiSwdSave	✓	✓	✓				
AComTiSwdSaveAs	✓	✓	✓				
AComTiTableColumnToggle	✓	✓	✓				
AComTiTableCopy	✓	✓	✓				
AComTiTableCut	✓	✓	✓				
AComTiTableDelete	✓	✓	✓				
AComTiTableProperties	✓	✓	✓				

	MM	ME	MD	OV	OM	OD	ASC
`AComTiTableRename`	✓	✓	✓				
`AComTiTableView`	✓	✓	✓				
`AComZoomIn`	✓	✓	✓	✓	✓	✓	

Appendix 4

Cadcorp SIS properties

- Introduction 345
- Item properties (SIS_OT_CURITEM) 345
- Dataset properties (SIS_OT_DATASET). 350
- Default item properties (SIS_OT_DEFITEM) 366
- Feature table properties (SIS_OT_FTABLE) 367
- Named object library properties (SIS_OT_NOL). 368
- System options (SIS_OT_OPTION) 370
- Overlay properties (SIS_OT_OVERLAY) 373
- Printer properties (SIS_OT_PRINTER) 375
- Schema properties (SIS_OT_SCHEMA) 376
- Schema column properties (SIS_OT_SCHEMACOLUMN). 377
- System properties (SIS_OT_SYSTEM) 377
- Theme properties (SIS_OT_THEME) 383
- Theme component properties 405
- Window properties (SIS_OT_WINDOW) 405

■ Introduction

This appendix describes the properties available in Cadcorp SIS.

■ Item properties (SIS_OT_CURITEM)

Item is the root or base class for all other item classes. Item properties are valid for all item sub-classes.

Alignment `_text_align&`

the combined horizontal and vertical alignment of text about its digitised position. The values have the same meaning as the Win32 function `SetTextAlign()`, with the addition of Middle alignment.

Valid values

```
SIS_TOP_LEFT
SIS_TOP_RIGHT
SIS_TOP_CENTRE
SIS_BOTTOM_LEFT
SIS_BOTTOM_RIGHT
SIS_BOTTOM_CENTRE
SIS_BASE_LEFT
SIS_BASE_RIGHT
```

```
SIS_BASE_CENTRE
SIS_MIDDLE_LEFT
SIS_MIDDLE_RIGHT
SIS_MIDDLE_CENTRE
```

Bold `_text_bold&`

Is the text bold?

Valid values

True or False

Box `_text_box&`

Should a box be drawn around box text characters?

Valid values

True or False

Brush `_brush$`

the brush used to cover any area covered by the item

Class name `_class$`

the class of the item. Use this value in API methods which require an item class, eg the `formula$` argument in `CreateClassTreeFilter`.

Dataset `_dataset$`

the name of the item's dataset

Description `_DESC$`

a textual description of the item. MapTips often displays the description.

Dimension `_dimension&`

the dimension of an item. A point is 0-dimensional, a line is 1-dimensional, an area is 2-dimensional, a surface that is solid is 3-dimensional.

Feature code `_FC&`

the feature code for the item. If an item has a feature code, its pen, brush, shape, etc are taken from a feature table. You will not be able to change many of the item's properties until you remove the feature code. You can remove a feature code by clearing the item's Feature table property.

Feature table `_featureTable$`

the feature table which an item uses to get information about a feature code

Font `_font$`

the font used to draw the text. Cadcorp SIS will look for a TrueType font, which has the same name.

Horizontal `_bHorizontal&`

Should point geometry within this item force any shape to be drawn horizontally?

Valid values

True or False

Horizontal Alignment: `_text_alignH&`

the horizontal alignment of text about its digitised position

Valid values

`SIS_LEFT SIS_CENTRE SIS_RIGHT`

Italic `_text_italic&`

Is the text drawn italic?

Valid values

True or False

Item class `_classLocal$`

the item's class. Every item has a class, which determines its appearance.

Item ID `_id&`

the ID of the item. Every item has an ID number, which is unique within its dataset. The ID for an item will stay the same for the lifetime of the item. Item ID values are allocated automatically.

Layer `_layer$`

the layer attribute of an item. You can enter any layer name on editable items. Layers can then be turned on and off using the session window. The concept of layers is used in many CAD systems, and in the AutoCAD DXF and DWG formats.

Length `_length#`

the length of the geometry in metres

Level `_level&`

the level within an overlay that an item is drawn on. If lots of items are in one overlay, you can control the order in which they are drawn by setting their levels.

Maximum scale `_scalemax#`

the highest reproduction scale at which an item will be drawn. If you set this value to 1000, and then zoom out to 1:2000, the item will become invisible.

This value is stored internally using a low precision byte. The value you enter will be rounded to the nearest valid value automatically.

Minimum scale `_scalemin#`

the lowest reproduction scale at which an item will be drawn. If you set this value to 100, and then zoom in to 1:50 the item will become invisible.

This value is stored internally using a low precision byte. The value you enter will be rounded to the nearest valid value automatically.

Opaque `_text_opaque&`

Should text be drawn with an opaque background? If you make text opaque, the text item's brush is used to colour in the space around the letters. This is useful to ensure that text is legible when it is drawn on top of complicated graphics.

Origin Latitude `_oLat#`

the latitude origin, in degrees

Origin Longitude `_oLon#`

the longitude origin, in degrees

Origin X `_ox#`

the X co-ordinate of the item's origin in metres from the dataset origin

Origin Y `_oy#`

the Y co-ordinate of the item's origin in metres from the dataset origin

Origin Z `_oz#`

the Z co-ordinate of the item's origin in metres from the dataset origin

Parent feature code `_parent_FC&`

the parent feature code

Pen `_pen$`

the pen used to draw the item

Point height `_point_height&`

the height of the text in points. Since Cadcorp SIS mapping data can be printed at any scale, the item's dataset scale is used to determine how big a point is in the real world. (You can use negative point sizes to fix the size of the text relative to the screen instead of relative to the real world.)

Scale `_scale#`

the scale of the shape. When shape objects are defined in a named object library they are measured in paper units. You can use this scale property to make the shape larger or smaller than its designed size.

Since Cadcorp SIS mapping data can be printed at any scale, the item's dataset scale is also used to scale the shape. (You can use negative point sizes to fix the size of the shape objects relative to the screen instead of relative to the real world.)

Shape `_shape$`

the shape used to draw the point item

Simple `_bSimple&`

Is the item simple?

Valid values

True or False

Size in X `_sx#`

the width of the item in metres

Size in Y `_sy#`

the height of the item in metres

Size in Z `_sz#`

the depth of the item in metres

Straight `_straight&`

Is the geometry completely made up of straight line segments? In mapping datasets, most line items will be straight, but many CAD-style drawings use curved geometry like circles, arcs, and Bezier curves.

Thin in X `_thinX&`

Does all geometry in the item have the same X co-ordinate? This can be used in the initial stages of a data cleaning operation to filter out lines which are horizontal (and are therefore probably Y grid lines).

Valid values

True or False

Thin in Y `_thinY&`

Does all geometry in the item have the same Y co-ordinate? This can be used in the initial stages of a data cleaning operation to filter out lines which are horizontal (and are therefore probably X grid lines).

Valid values

True or False

Thin in Z `_thinZ&`

Does all geometry in the item have the same Z co-ordinate? This can be used to filter items which are completely on the XY plane (or on a parallel plane.) This is most likely to be used in the initial stages of a data cleaning operation.

Valid values

True or False

Underlined `_text_underlined&`

Is the text underlined?

Valid values

True or False

Upright `_text_upright&`

Are the text letters always drawn upright? You can use this property to ensure that the text letters are drawn and printed upright, even if the text or the view is rotated.

Valid values

True or False

URI `_URI$`

the Uniform Resource Identifier (URI) associated with an item, which allows you to link to an object. This object could be a web page, an image file, or an application.

Vertical alignment `_text_alignV&`

the vertical alignment of the text about its digitised position

Valid values

`SIS_TOP SIS_MIDDLE SIS_BASELINE SIS_BOTTOM`

■ Dataset properties (SIS_OT_DATASET)

Cadcorp SIS understands many different datasets, eg AutoCAD DXF or Ordnance Survey NTF, and each of these datasets has its own properties.

◆ Base Dataset (*.bds)

Attributes `_attributes$`

a list of attributes held by items in the dataset

Class name `_class$`

the class name of the dataset

Configuration `_configuration$`

a string used to represent the state of a dataset. Some datasets store state information in the SWD. The format of the configuration string varies from dataset to dataset. It is often ASCII encoded binary information.

Editable `_bEditable&`

Is the dataset editable?

Valid values

True or False

Feature table `_featureTable$`

the feature table which dataset items with no feature table of their own use when their feature table is set. This feature table is also used to get information about available feature codes, and to create a default feature table.

This does not override the feature table on an item.

Layers `_layers$`

the set of all values of the layer property for all items in the dataset

Maximum number of open gateways `_nMaxGatewayOpen&`

the maximum number of gateway sub-datasets a dataset should attempt to open simultaneously

Modified `_bModified&`

Has the dataset been modified?

Valid values

True or False

Name `_name$`

the dataset name. For a file-based dataset, this will be the filename.

Next item ID `_idNextItem&`

the item ID, which will be used for the next item created in this dataset

Notes `_notes$`

the user's notes on the dataset

Number of items `_nItems&`

the number of items in the dataset

Owned `_bOwned&`

Is the dataset owned? An owned dataset may be edited by the owner. Other users on a network will be able to see the dataset, but not gain ownership to it. A dataset may be disowned to allow editable access to another user.

Valid values

True or False

Precision `_precision&`

the precision of the items in the dataset

Valid values

16 16-bit integers

32 32-bit integers

64 64-bit double precision floating point numbers

Projection `_projection$`

the dataset projection in OpenGIS Well-Known-Text format. (Some Cadcorp SIS Projection objects are not supported in OpenGIS.)

Scale `_scale#`

the dataset scale

Size `_size&`

the amount of memory, in bytes, that the dataset uses

◆ Editable Blobs

Attributes `_attributes$`

a list of attributes held by items in the dataset

Blob format `_blobFormat$`

a description of the format of Blobs used to represent items

Class name `_class$`

the class name of the dataset

Database connection `_connect$`

the connection string of the dataset's recordset

Editable `_bEditable&`

Is the dataset editable?

Valid values

True or False

Feature table `_featureTable$`

the feature table which dataset items with no feature table of their own use when their feature code is set. This feature table is also used to get information about available feature codes, and to create a default feature filter.

This does not override the feature table on an item.

Fetch size `_fetchSize&`

the size of a SQL fetch used when retrieving a Blob string from the database table, in the range 1KB (1024) to 1MB (1024 × 1024). This value is used when the `_MaxBlobSize&` property is set to 0, and the database cannot tell Cadcorp SIS how long the Blob string is. In this case, Cadcorp SIS will get chunks of bytes, where the starting chunk size is this value, until the whole Blob string has been read. Each subsequent chunk size is twice the previous chunk size. For example, if this value is set to 4KB (4096), the chunk sizes will be 4KB, 8KB, 16KB, 32KB, etc, until the whole Blob string has been read.

Be careful when setting this value. The wrong value, either too large or too small, could have a significant effect on Blob loading times.

If this value is 0, the default fetch size will be used (currently 64KB).

Item table `_itemTable$`

the name of the database table containing item blobs

Layers `_layers$`

the set of all values of the Layer property for all items in the dataset

Maximum Blob size `_MaxBlobSize&`

the maximum length of a Blob string in the database table, in the range 1KB (1024) to 4MB (4 × 1024 × 1024). If this value is set to 0, Blob strings of any size can be read and written, with no limit other than available memory.

Maximum number of open gateways `_nMaxGatewayOpen&`

the maximum number of gateway sub-datasets a dataset should attempt to open simultaneously

Modified `_bModified&`

Has the dataset been modified?

Valid values

True or False

Name `_name$`

the dataset name. For a file-based dataset this will be the filename.

Next item ID `_idNextItem&`

the item ID which will be used for the next item created in this dataset

Notes `_notes$`

the user's notes on the dataset

Number of failed edits `_nEditBad&`

the number of failed edits that have occurred in an editable Blobs or OpenGIS SQL92 Database dataset

Number of items `_nItems&`

the number of items in the dataset

Number of successful edits `_nEditGood&`

the number of successful edits that have occurred in an editable Blobs or OpenGIS SQL92 Database dataset

Precision `_precision&`

the precision of the items in the dataset

Valid values

16 16-bit integers

32 32-bit integers

64 64-bit double precision floating point numbers

Projection `_projection$`

the dataset Projection in OpenGIS Well-Known-Text format. (Some Cadcorp SIS Projection objects are not supported in OpenGIS.)

Scale `_scale#`

the dataset scale

Size `_size&`

the amount of memory, in bytes, that the dataset uses

Textual Blobs `_bTextBlob&`

Are item Blobs in ASCII text format, or binary? This affects how Cadcorp SIS communicates with the dataset's database.

Valid values

True or False

Transactions `_bTransact&`

Does the database-based dataset use transactions?

Valid values

True or False

◆ Index dataset

Attributes `_attributes$`

a list of attributes held by items in the dataset

Class name `_class$`

the class name of the dataset

Editable `_bEditable&`

Is the dataset editable?

Valid values

True or False

Feature table `_featureTable$`

the feature table which dataset items with no feature table of their own use when their feature code is set. This feature table is also used to get information about available feature codes, and to create a default feature filter. This does not override the feature table on an item.

Gateways `_bGateway&`

Does an index dataset display tile outlines (using gateway items) for all of the tiles found?

Valid values

True or False

Label `_bLabel&`

Does an index dataset display box text labels for all of the tiles found?

Valid values

True or False

Layers `_layers$`

the set of all values of the layer property for all items in the dataset

Maximum number of open gateways `_nMaxGatewayOpen&`

the maximum number of gateway sub-datasets a dataset should attempt to open simultaneously

Maximum number of open tiles `_MaxOpen&`

the maximum number of tiles that an index dataset will attempt to open at once, in the range 0 to 1024. Setting the maximum to 0 will make the index dataset display tile

outlines (using gateway items and box text labels for each tile found, but no data in the tiles themselves, thus automatically creating a key map).

Modified `_bModified&`

Has the dataset been modified?

Valid values

True or False

Name `_name$`

the dataset name. For a file-based dataset, this will be the filename.

Next item ID `_idNextItem&`

the item ID, which will be used for the next item, created in this dataset

Notes `_notes$`

the user's notes on the dataset.

Number of items `_nItems&`

the number of items in the dataset

Overflow for tile items `_fractionOverflow#`

the fractional overflow of an index dataset, in the range 0.0 to 1.0. An index dataset uses this value when the map view changes and they attempt to open any tiles which fall within the new view. A value of 0.0 will do no padding, and a value of 1.0 will force two extra rows and columns of tiles to be opened (if the tiles exist). Values between 0.0 and 1.0 may open extra rows or columns of tiles depending on the value and how close the new view extents are to the tile edges.

This value is useful for viewing an index dataset which contains tiles whose data overflows the tile edges.

Precision `_precision&`

the precision of the items in the dataset

Valid values

16 16-bit integers

32 32-bit integers

64 64-bit double precision floating point numbers

Projection `_projection$`

the dataset projection in OpenGIS Well-Known-Text format. (Some Cadcorp SIS Projection objects are not supported in OpenGIS.)

Pyramid `_bPyramid&`

Should an index dataset attempt to open dataset tiles with different sizes?

Valid values

True or False

Scale `_scale#`

the dataset scale

Size `_size&`

the amount of memory, in bytes, that the dataset uses

Size for tile labels `_fractionLabel#`

the fraction of the height of an index dataset tile used to calculate the tile label text height, in the range 0.001 to 2.0. The tile label text height is calculated as follows

`TextHeight=(TileHeight)*(_fractionLabel#)/ (NumberCharactersInLabel)`

Tile path `_tilePath$`

a list of directories and file extensions used by an index dataset, separated by semi-colons, eg `C:Local*.bds;C:Local*.bmp;D:Network*.bds`

◆ Internal dataset

Attributes `_attributes$`

a list of attributes held by items in the dataset

Class name `_class$`

the class name of the dataset

Editable `_bEditable&`

Is the dataset editable?

Valid values

True or False

Feature table `_featureTable$`

the feature table which dataset items with no feature table of their own use when their feature table is set. This feature table is also used to get information about available feature codes, and to create a default feature filter. This does not override the feature table on an item.

Layers `_layers$`

the set of all values of the layer property for all items in the dataset

Maximum item status `_maxItemStatus&`

the maximum edit status for items in the dataset

Valid values

`SIS_INVISIBLE` items are invisible

`SIS_VISIBLE` items are visible

`SIS_HITTABLE` items are visible and can be selected

`SIS_EDITABLE` items are selectable and can be edited

Maximum number of open gateways `_nMaxGatewayOpen&`

the maximum number of gateway sub-datasets a dataset should attempt to open simultaneously

Modified `_bModified&`

Has the dataset been modified?

Valid values

True or False

Name `_name$`

the dataset name. For a file-based dataset, this will be the filename.

Next item ID `_idNextItem&`

the item ID, which will be used for the next item created in this dataset

Notes `_notes$`

the user's notes on the dataset

Number of items `_nItems&`

the number of items in the dataset

Pending `_bPending&`

Is the dataset waiting for an operation to complete?

Valid values

True or False

Precision `_precision&`

the precision of the items in the dataset

16 16-bit integers

32 32-bit integers

64 64-bit double precision floating point numbers

Projection `_projection$`

the dataset projection in OpenGIS Well-Known-Text format. (Some Cadcorp SIS Projection objects are not supported in OpenGIS.)

Scale `_scale#`

the dataset scale

Size `_size&`

the amount of memory, in bytes, that the dataset uses

◆ OpenGIS SQL92 Database

Attributes `_attributes$`

a list of attributes held by items in the dataset

Class name `_class$`

the class name of the dataset

DB feature table `_F_TABLE_NAME$`

the name of the OpenGIS table containing feature information

DB geometry table `_G_TABLE_NAME$`

the name of the OpenGIS table containing geometry information

Database connection `_connect$`

the connection string of the dataset's recordset

Editable `_bEditable&`

Is the dataset editable?

Valid values

True or False

Feature table `_featureTable$`

the feature table which dataset items with no feature table of their own use when their feature code is set. This feature table is also used to get information about available feature codes, and to create a default feature filter. This does not override the feature table on an item.

Fetch size `_fetchSize&`

the size of a SQL fetch used when retrieving a Blob string from the database table, in the range 1KB (1024) to 1MB (1024 × 1024). ➲page 352, **Fetch size**

Layers `_layers$`

the set of all values of the Layer property for all items in the dataset

Maximum Blob size `_MaxBlobSize&`

the maximum length of a Blob string in the database table, in the range 1KB (1024) to 4MB (4 × 1024 × 1024). If this value is set to 0, Blob strings of any size can be read and written, with no limit other than available memory.

Maximum number of open gateways `_nMaxGatewayOpen&`

the maximum number of gateway sub-datasets a dataset should attempt to open simultaneously

Modified `_bModified&`

Has the dataset been modified?

Valid values

True or False

Name `_name$`

the dataset name. For a file-based dataset this will be the filename.

Next item ID `_idNextItem&`

the item ID, which will be used for the next item created in this dataset

Notes `_notes$`

the user's notes on the dataset

Number of failed edits `_nEditBad&`

the number of failed edits that have occurred in an Editable Blobs or OpenGIS SQL92 dataset

Number of items `_nItems&`

the number of items in the dataset

Number of successful edits `_nEditGood&`

the number of successful edits that have occurred in an Editable Blobs or OpenGIS SQL92 dataset

Precision `_precision&`

the precision of the items in the dataset

Valid values

16 16-bit integers

32 32-bit integers

64 64-bit double precision floating point numbers

Projection `_projection$`

the dataset Projection in OpenGIS Well-Known-Text format. (Some Cadcorp SIS Projection objects are not supported in OpenGIS.)

Scale `_scale#`

the dataset scale

Size `_size&`

the amount of memory, in bytes, that the dataset uses

Transactions `_bTransact&`

Does the database-based dataset use transactions?

Valid values

True or False

◆ Raster File (*.bmp, *.gif, *.jpg, *.jpeg, *.png, *.rlc, *.tif, *.tiff)

Attributes `_attributes$`

a list of attributes held by items in the dataset

Class name `_class$`

the class name of the dataset

Configuration `_configuration$`

a string used to represent the state of a dataset. Some datasets store state information in the SWD. The format of the configuration string varies from dataset to dataset. It is often ASCII encoded binary information.

Editable `_bEditable&`

Is the dataset editable?

Valid values

True or False

Feature table `_featureTable$`

the feature table which dataset items with no feature table of their own use when their feature code is set. This feature table is also used to get information about available feature codes, and to create a default feature filter.

This does not override the feature table on an item.

Layers `_layers$`

the set of all values of the Layer property for all items in the dataset

Maximum number of open gateways `_nMaxGatewayOpen&`

the maximum number of gateway sub-datasets a dataset should attempt to open simultaneously

Modified `_bModified&`

Has the dataset been modified?

Valid values

True or False

Name `_name$`

the dataset name. For a file-based dataset this will be the filename.

Next item ID `_idNextItem&`

the item ID which will be used for the next item created in this dataset

Notes `_notes$`
the user's notes on the dataset

Number of items `_nItems&`
the number of items in the dataset

Precision `_precision&`
the precision of the items in the dataset

Valid values

16 16-bit integers

32 32-bit integers

64 64-bit double precision floating point numbers

Projection `_projection$`
The dataset Projection in OpenGIS Well-Known-Text format. (Some Cadcorp SIS Projection objects are not supported in OpenGIS.)

Scale `_scale#`
the dataset scale

Size `_size&`
the amount of memory, in bytes, that the dataset uses

◆ View Blobs

Attributes `_attributes$`
a list of attributes held by items in the dataset

Blob format `_blobFormat$`
a description of the format of blobs used to represent items

Class name `_class$`
the class name of the dataset

Column aliases `_aliases$`
a comma-separated list of column aliases in the dataset's recordset

Column names `_columns$`
a comma-separated list of column names in the dataset's recordset

Database connection `_connect$`
the connection string of the dataset's recordset

Editable `_bEditable&`
Is the dataset editable?

Valid values

True or False

Feature table `_featureTable$`

the feature table which dataset items with no feature table of their own use when their feature code is set. This feature table is also used to get information about available feature codes, and to create a default feature filter. This does not override the feature table on an item.

Fetch size `_fetchSize&`

the size of a SQL Fetch used when retrieving a Blob string from the database table, in the range 1KB (1024) to 1MB (1024 × 1024). ➲page 352, **Fetch size**

Layers `_layers$`

the set of all values of the Layer property for all items in the dataset

Maximum Blob size `_MaxBlobSize&`

the maximum length of a Blob string in the database table, in the range 1KB (1024) to 4MB (4 × 1024 × 1024). If this value is set to 0, Blob strings of any size can be read and written, with no limit other than available memory.

Maximum number of open gateways `_nMaxGatewayOpen&`

the maximum number of gateway sub-datasets a dataset should attempt to open simultaneously

Maximum range per SQL SELECT `_MaxRangePerSql&`

This controls the number of SQL SELECT statements used when a database-based dataset which uses a spatial reference asks the dataset for items within a view.

Modified `_bModified&`

Has the dataset been modified?

Valid values

True or False

Name `_name$`

the dataset name. For a file-based dataset this will be the filename.

Next item ID `_idNextItem&`

the item ID which will be used for the next item created in this dataset

Notes `_notes$`

the user's notes on the dataset

Number of items `_nItems&`

the number of items in the dataset

Precision `_precision&`

the precision of the items in the dataset

16 16-bit integers

32 32-bit integers

64 64-bit double precision floating point numbers

Projection `_projection$`

the dataset projection in OpenGIS Well-Known-Text format. (Some Cadcorp SIS Projection objects are not supported in OpenGIS.)

Scale `_scale#`

the dataset scale

Size `_size&`

the amount of memory, in bytes, that the dataset uses

SQL WHERE expression `_where$`

the SQL WHERE expression in a dataset's recordset

Table names `_tables$`

a comma-separated list of table names in a database dataset's recordset

Textual Blobs `_bTextBlob&`

Are item blobs in ASCII text format, or binary? This affects how Cadcorp SIS communicates with the dataset's database.

◆ View points

Attributes `_attributes$`

a list of attributes held by items in the dataset

Class name `_class$`

the class name of the dataset

Column aliases `_aliases$`

a comma-separated list of column aliases in the dataset's recordset

Column names `_columns$`

a comma-separated list of column names in the dataset's recordset

Column number for X co-ordinates `_nFieldX&`

the table column number of X co-ordinates

Column number for Y co-ordinates `_nFieldY&`

the table column number of Y co-ordinates

Column number for Z co-ordinates `_nFieldZ&`

the table column number of Z co-ordinates

Database connection `_connect$`

the connection string of the dataset's recordset

Editable `_bEditable&`

Is the dataset editable?

Valid values

True or False

Feature table `_featureTable$`

the feature table which dataset items with no feature table of their own use when their feature code is set. This feature table is also used to get information about available feature codes, and to create a default feature filter. This does not override the feature table on an item.

Layers `_layers$`

the set of all values of the Layer property for all items in the dataset

Maximum number of open gateways `_nMaxGatewayOpen&`

the maximum number of gateway sub-datasets a dataset should attempt to open simultaneously

Maximum range per SQL SELECT `_MaxRangePerSql&`

This controls the number of SQL SELECT statements used when a database-based dataset which uses a spatial reference asks the dataset for items within a view.

Modified `_bModified&`

Has the dataset been modified?

Valid values

True or False

Name `_name$`

the dataset name. For a file-based dataset this will be the filename.

Next item ID `_idNextItem&`

the item ID, which will be used for the next item created in this dataset

Notes `_notes$`

the user's notes on the dataset

Number of items `_nItems&`

the number of items in the dataset

Precision `_precision&`

The precision of the items in the dataset.

16 16-bit integers

32 32-bit integers

64 64-bit double precision floating point numbers

Projection `_projection$`

the dataset Projection in OpenGIS Well-Known-Text format. (Some Cadcorp SIS Projection objects are not supported in OpenGIS.)

Scale `_scale#`

the dataset scale

Size `_size&`

the amount of memory, in bytes, that the dataset uses

SQL WHERE expression `_where$`

the SQL WHERE expression in a dataset's recordset

Table names `_tables$`

a comma-separated list of table names in a database dataset's recordset

◆ Windows MetaFile (*.wmf)

Attributes `_attributes$`

a list of attributes held by items in the dataset

Class name `_class$`

the class name of the dataset

Configuration `_configuration$`

a string used to represent the state of a dataset. Some datasets store state information in the SWD. The format of the configuration string varies from dataset to dataset. It is often ASCII encoded binary information.

Editable `_bEditable&`

Is the dataset editable?

Valid values

True or False

Feature table `_featureTable$`

the feature table which dataset items with no feature table of their own use when their feature code is set. This feature table is also used to get information about available feature codes, and to create a default feature filter. This does not override the feature table on an item.

Layers `_layers$`

the set of all values of the Layer property for all items in the dataset

Maximum number of open gateways `_nMaxGatewayOpen&`

the maximum number of gateway sub-datasets a dataset should attempt to open simultaneously

Modified `_bModified&`

Has the dataset been modified?

Valid values

True or False

Name `_name$`

the dataset name. For a file-based dataset this will be the filename.

Next item ID `_idNextItem&`

the item ID, which will be used for the next item created in this dataset

Notes `_notes$`

the user's notes on the dataset

Number of items `_nItems&`

the number of items in the dataset

Precision `_precision&`

the precision of the items in the dataset

16 16-bit integers

32 32-bit integers

64 64-bit double precision floating point numbers

Projection `_projection$`

the dataset Projection in OpenGIS Well-Known-Text format. (Some Cadcorp SIS Projection objects are not supported in OpenGIS.)

Scale `_scale#`

the dataset scale

Size `_size&`

the amount of memory, in bytes, that the dataset uses

■ Default item properties (SIS_OT_DEFITEM)

Whenever a new item is created by a Cadcorp SIS command it takes its default properties from those of the default item. ➲page 345, **Item properties (SIS_OT_CURITEM)**

■ Feature table properties (SIS_OT_FTABLE)

Cadcorp SIS uses named feature table objects to control the display (such as the pen, brush, shape, and so on) of feature coded items.

Brush `_brush$`

the brush used to fill any area covered by an item with this feature code

Description `_DESC$`

a textual description of the feature code. MapTips often displays the description.

Feature code `_FC&`

the feature code itself

Font `_font$`

the font used to draw text in any items with this feature code. Cadcorp SIS will look for a TrueType font, which has the same name.

Layer `_layer$`

the layer name of this feature code

Level `_level&`

the level within an overlay on which to draw an item with this feature code

Maximum scale `_scalemax#`

the highest reproduction scale at which items with this feature code will be drawn. If you set this value to 1000, and then zoom out to 1:2000, the item will become invisible. This value is stored internally using a low precision byte. The value you enter will be rounded to the nearest valid value automatically.

Minimum scale `_scalemin#`

the lowest reproduction scale at which items with this feature code will be drawn. If you set this value to 100, and then zoom out to 1:50, the item will become invisible. This value is stored internally using a low precision byte. The value you enter will be rounded to the nearest valid value automatically.

Parent feature code `_ParentFC&`

the parent feature code of this feature code, allowing hierarchical trees of feature codes to be constructed

Pen `_pen$`

the pen used to draw any item with this feature code

Point height `_point_height&`

the height in points at which to draw text items with this feature code

Shape `_shape$`

the shape used to draw any point item with this feature code

■ Named object library properties (SIS_OT_NOL)

Cadcorp SIS has a set of named object libraries which contain named object library classes, eg pen, brush, filter objects, etc. These named object library classes are used throughout Cadcorp SIS.

For many of the named properties, the returned string is tab-separated because object names can contain spaces.

Disabled `_bDisabled&`

Has the named object library been disabled? Named object libraries which are disabled are not searched for named object library classes.

Valid values

True or False

Modified `_bModified&`

Has the named object library been modified?

Valid values

True or False

Name `_name$`

the named object library name. The named object library names (standard), (workspace) and (temporary) are special. All other named object library names are filenames.

Named blocks `_listBlock$`

a tab-separated list of named block objects in the named object library

Named brushes `_listBrush$`

a tab-separated list of named brush objects in the named object library

Named colour-sets `_listColourset$`

a tab-separated list of named colour-set objects in the named object library

Named datums `_listDatum$`

a tab-separated list of named Geoid Datum objects in the named object library

Named feature tables `_listFtable$`

a tab-separated list of named feature table objects in the named object library

Named filters `_listFilter$`

a tab-separated list of named filter objects in the named object library

Named graticule styles `_listGraticuleStyle$`

a tab-separated list of named graticule style objects in the named object library

Named items `_listLibItem$`

a tab-separated list of named item objects in the named object library

Named loci `_listLocus$`

a tab-separated list of named locus objects in the named object library

Named pens `_listPen$`

a tab-separated list of named pen objects in the named object library

Named print templates `_listPrintTemplate$`

a tab-separated list of named print template objects in the named object library

Named projections `_listPrj$`

a tab-separated list of named projection objects in the named object library

Named schemas `_listSchema$`

a tab-separated list of named schema objects in the named object library

Named shapes `_listShape$`

a tab-separated list of named shape objects in the named object library

Named themes `_listTheme$`

a tab-separated list of named theme objects in the named object library

Named toolbar definitions `_listToolBarDefn$`

a tab-separated list of named toolbar definition objects in the named object library

Named views `_listView$`

a tab-separated list of named view objects in the named object library

Owned `_bOwned&`

Is the named object library owned?

Valid values

True or False

An owned named object library can be edited by the owner. Other users on a network will be able to see the named object library, but not gain ownership to it. A named object library can be disowned to allow editable access to another user. You cannot disown a named object library if it is the default named object library, so you may have to change the default named object library first. If you disown a modified named object library without saving the changes, the named object library will be re-read from the original file.

Size `_size&`

the amount of memory, in bytes, that the named object library uses

Type `_type&`

the type of the named object library

Valid values

`SIS_NOL_FILE`	the named object library is file-based
`SIS_NOL_STANDARD`	the named object library is the (standard) built-in named object library
`SIS_NOL_TEMPORARY`	the named object library is the (temporary) built-in named object library
`SIS_NOL_WORKSPACE`	the named object library is the (workspace) named object library that is part of the current project workspace file

■ System options (SIS_OT_OPTION)

Cadcorp SIS has several Boolean (True or False) system options, which are global to all windows.

Auto create chain items? `_bAutoCreateChains&`

Should chain items be automatically created when link items are created?

Valid values

True or False

Check attributes when merging? `_bCheckMergedAttribs&`

Should Cadcorp SIS check attributes when merging items, and fail if they are different?

Check feature codes when merging? `_bCheckMergedFcodes&`

Should Cadcorp SIS check feature codes when merging items, and fail if they are different?

Check for network dongle? `_bCheckNetworkDongle&`

Should Cadcorp SIS check for a network hardware lock (dongle) if no local hardware lock is found?

Valid values

True or False

Confirm no undo delete? `_bAutoCreateChains&`

Should Cadcorp SIS seek confirmation before deleting items with no support for undo?

Valid values

True or False

Draw rotated bitmaps? `_bDrawRotatedBitmaps&`

Should rotated bitmap items be drawn? Drawing rotated bitmap items is several times slower than drawing unrotated bitmap items, so you can turn this option off for faster redraw times.

Valid values

True or False

Favour old links? `_bFavourOldLinks&`

Should old geometry be favoured in splinter removal? When you convert line and area items to topology, there may be small splinters between the line/area and existing link items. If these splinters are automatically removed, Cadcorp SIS needs to know whether the old link geometry is more reliable than the newer line/area geometry.

Valid values

True or False

Fill areas in 3D? `_bFillAreaIn3d&`

Should 2D items (eg area, polygon) be filled in 3D windows? To fill 2D items in OpenGL, Cadcorp SIS must split the item into triangles. Splitting large concave polygons (typical in GIS data) could slow down drawing 3D views. Also, if the 2D item is not planar, the triangulation chosen may not give you the 3D graphical results you wanted.

Valid values

True or False

Fill surfaces in 2D? `_bFillSurfaceIn2d&`

Should surface items be filled in 2D windows? In a complex 3D model, you may find it easier to work with items in 2D views when they are not filled.

Valid values

True or False

Flicker display? `_bFlickerDisplay&`

Should the selected items flicker?

Valid values

True or False

Highlight selected shapes? `_bHighlightSelectedShapes&`

Should selected shapes and text be highlighted. With this option, when you select a point or text item, the whole shape or text, respectively, is highlighted. Without this option, only a small diamond is drawn over the item origin.

If this option is set, point and text items will draw their shape and text, respectively, when dragging.

Valid values

True or False

Print rotated bitmaps? `_bPrintRotatedBitmaps&`

Should rotated bitmap items be printed. You can choose to print rotated bitmap items independently of whether they are drawn to screen.

Valid values

True or False

Refresh Base Datasets? `_bRefreshBds&`

Should Base Dataset datasets be automatically refreshed? Cadcorp SIS can look on the network disk at intervals, set using `_BdsRefreshInterval&`, to see if another user has modified and saved the Base Dataset files which you are working with. If the Base Dataset file has changed, Cadcorp SIS will automatically reload it. This option increases network traffic, so you should not use it if your network is slow. Also, Cadcorp SIS cannot detect changed files with some combinations of server and client operating systems.

Valid values

True or False

Select overlap automatic? `_bSelectOverlapAutomatic&`

Should the Select Item dialog be shown whenever you snap onto overlapping items?

Valid values

True or False

Show MapTips? `_bShowMapTips&`

Should MapTips be shown when the cursor hovers over an item in a map window?

Valid values

True or False

Simulate GDI styles? `_bSimulateGdiStyles&`

Should GDI pen styles be simulated? Some older printer drivers cannot draw dotted and dashed pen styles. If this is the case, you can force Cadcorp SIS to calculate the start and end positions of the dots and dashes within styled pens. This is a reliable way to get reproducible styles on all output devices. The drawback is that it is slower than using Windows GDI (Graphics Device Interface).

Valid values

True or False

Smooth scrolling? `_bSmoothScrolling&`

Should Cadcorp SIS, when scrolling, draw the exposed areas immediately, avoiding white space appearing in the map window?

Suppress backups? `_bSuppressBackups&`

Should Cadcorp SIS automatically make backup files? Whenever Cadcorp SIS writes a Base Dataset, named object library, project workspace or SWD file it can rename the old file with the name 'Backup of *filename*'. This can be useful to recover the situation when you accidentally save changes which you intended to discard. However, if you always have this option turned on, Cadcorp SIS will use up more of your disk space.

Valid values

True or False

Suppress scale warning? `_bSuppressScaleWarning&`

Should Cadcorp SIS, whenever you are creating new point, text and dimension items, warn that the current dataset scale is unsuitable, eg the text will be very small or very large on screen. The Scale Warning dialog is displayed which allows a more suitable scale to be chosen. This option prevents the Scale Warning dialog from being displayed.

Valid values

True or False

Suppress undo? `_bSuppressUndo&`

Should Cadcorp SIS use the undo buffer? Whenever Cadcorp SIS items are added, edited, or deleted, the change is remembered in a memory buffer. You can use this option to prevent the changes being remembered, and thus reduce memory requirements.

Valid values

True or False

Transparent zooming? `_bTransparentZoom&`

Should the zoom commands (eg **Map>Zoom>In**, **Map>Zoom>Out**, **Map>Pan>Snap**, and so on) be 'transparent'? When the zoom commands are transparent, you can pan and zoom around the map base while running other commands, eg when capturing graphics.

■ Overlay properties (SIS_OT_OVERLAY)

Attributes `_attributes$`

the user defined attributes stored with the overlay

Brush `_brush$`

the brush to use for items if their brush is By Overlay, ie if they have no brush of their own, or if the overlay brush override is set

Brush override `_bBrushOverride&`

Should the overlay brush override any item brush?

Valid values

True or False

Colour `_colour$`

the overlay colour, a space-separated RGB triple. For example, "0 0 255" is blue and "255 255 255" is white. When the overlay colour override is True, items will use this colour instead of their pen colour.

Colour override `_bColourOverride&`

Should the overlay colour override any item pen colour? If this is False, the overlay colour is ignored.

Valid values

True or False

Dataset `_nDataset&`

the serial number of the overlay's dataset

External `_bExternal&`

Is the overlay's dataset a file?

Valid values

True or False

Font `_font$`

the font to use for items if their font is By Overlay, ie if they have no font of their own or if the overlay font override is set

Font override `_bFontOverride&`

Should the overlay font override any item font?

Valid values

True or False

Icon `_nIcon&`

the overlay icon index

Maximum scale `_scalemax#`

the highest reproduction scale at which an item on an overlay will be drawn. If you set this value to 1000 and then zoom out to 1:2000, the item will become invisible. This value is stored internally using a low precision byte. The value you enter will be rounded to the nearest valid value automatically.

Minimum scale `_scalemin#`

the lowest reproduction scale at which an item on an overlay will be drawn. If you set this value to 100 and then zoom out to 1:50 the item will become invisible. This value is stored internally using a low precision byte. The value you enter will be rounded to the nearest valid value automatically.

Name `_name$`

the overlay name. If the overlay's dataset is a file, this name is an alias.

Notes `_notes$`

the overlay notes. These notes can be used to store free-form textual information about the overlay.

Number of themes `_nTheme&`

the number of theme objects on the overlay

Pen `_pen$`

the pen to use for items if their pen is By Overlay, ie if they have no pen of their own or if the overlay pen override is set

Pen override `_bPenOverride&`

Should the overlay pen override any item pen?

Valid values

True or False

Scale `_scale#`

the overlay scale, which optionally overrides the overlay's dataset scale

Scale override `_bScaleOverride&`

Should the overlay scale override the overlay's dataset scale?

Valid values

True or False

Shape `_shape$`

the shape to use for items if their shape is By Overlay, ie if they have no shape of their own or if the overlay shape override is set

Shape override `_bShapeOverride&`

Should the overlay shape override any item shape?

Valid values

True or False

Status `_status&`

Valid values

`SIS_INVISIBLE`	items are invisible
`SIS_VISIBLE`	items are visible
`SIS_HITTABLE`	items are visible and can be selected
`SIS_EDITABLE`	items are selectable and can be edited

■ Printer properties (SIS_OT_PRINTER)

Printer properties are used to control printer settings used by the method `SendPrint`.

Device name `_device$`

the printer device name. In Microsoft Visual Basic, this is the DeviceName property of a Printer object.

Driver name `_driver$`

the printer driver. This will typically be `winspool`. In Microsoft Visual Basic this is the DriverName property of a Printer object.

Number of copies `_copies&`

the number of copies to print, in the range 1 to 100

Orientation `_orientation&`

the printer page orientation, ie portrait or landscape. In Microsoft Visual Basic this is the Orientation property of a Printer object. The Microsoft Visual Basic constants for orientation, `vbPRORLandscape` and `vbPRORPortrait` should be used.

Output port `_output$`

the printer output port. In Microsoft Visual Basic, this is the Port property of a Printer object.

Paper length `_paperLength&`

the length of the printer page (in 1/10ths of millimetre), in the range 100 to 32000 (10mm to 3.2m). This property is used only if Paper size is set to 0 or 256.

In Microsoft Visual Basic, this can be calculated from the Height property of a Printer object. The Height property of a Microsoft Visual Basic Printer object is specified in *twips*, and should be converted to 1/10ths of a millimetre. A twip is a unit of screen measurement equal to 1/20 of a printer's point. There are approximately 1440 twips to a logical inch, 567 twips to a logical centimetre, and 5.67 twips to 1/10 of a millimetre (the length of a screen item measuring one inch or one centimetre or 1/10 of a millimetre when printed).

Paper size `_paperSize&`

the printer page size. If this is set to 0 or 256, the Paper length and Paper width properties will be used instead. In Microsoft Visual Basic, this is the PaperSize property of a Printer object. The Microsoft Visual Basic constants for the recognised paper sizes, eg use `vbPRPSA4` for A4.

Paper width `_paperWidth&`

the width of the printer page (in 1/10ths of millimetre), in the range 100 to 32000 (10mm to 3.2m). This property is only used if Paper size is set to 0 or 256.

In Microsoft Visual Basic this can be calculated from the Width property of a Printer object. The Width property of a Microsoft Visual Basic Printer object is specified in twips, and should be converted to 1/10ths of a millimetre. ➲page 376, **Paper length**

■ Schema properties (SIS_OT_SCHEMA)

Cadcorp SIS uses named schema objects to control the display of data-oriented parts of the user interface, eg the table window, etc.

Schema objects consist of a number of columns, each of which has a formula, which is used to evaluate values on items. For example, in the table window when viewing an overlay each row is equivalent to an item and each column comes from the overlay schema. The value in a cell is therefore calculated by evaluating the column formula, on the row item.

MapTip column `_nMapTipColumn&`

the index of the schema column which will appear in MapTips. This property has been superseded by the `_bMapTip&` schema column property.

Number of columns `_nColumns&`

the number of columns in the schema

■ Schema column properties (SIS_OT_SCHEMACOLUMN)

Description `_description$`

a description of the schema column. The description is used in several places in the user interface, eg the column name in a table window.

Formula `_formula$`

the formula used to calculate the value of a schema column

Hidden `_bHidden&`

Is the schema column hidden?

Valid values

True or False

Horizontal alignment `_hAlign&`

the horizontal alignment of a schema column. By default, numerical columns are right justified, and textual columns are left justified.

Valid values

```
SIS_LEFT
SIS_CENTRE
SIS_RIGHT
```

MapTip `_bMapTip&`

Does the schema column appear in MapTips?

Valid values

True or False

Width `_width&`

the initial width, in pixels, of a schema column when a table window is created

■ System properties (SIS_OT_SYSTEM)

Cadcorp SIS has several system variables, which are global to all windows.

Position argument `_ArgPos$`

the position argument entered by the user. This system variable is used by the GisLink methods `GetPos` and `GetPosEx`.

Prompt `_ArgPrompt$`

the prompt to display (in the Status Bar) while waiting for a user argument. This system variable is used by the GisLink methods `GetPos` and `GetPosEx`.

Asys command `_AsyncCommand$`

the callback command to run when the current operation is completed. This is used only by GisLink, which waits until `Release` or `ReleaseNA` is called before starting a command which requires user input.

Backup directory `_BakDir$`

the directory where backup files are created. By default this string is empty, which means that backups of Base Dataset, named object library, Project Workspace and SWD files are created adjacent to the originals. But, to provide some protection in the event of a disk crash, or to stop a disk filling up, you can store backups on a different disk.

BDS refresh interval `_BdsRefreshInterval&:`

the time between Base Dataset auto-refreshes, in seconds

Build number `_BuildNumber&`

the Cadcorp SIS build number. The complete Cadcorp SIS version number can be generated by concatenating `_VersionMajor&`, `_VersionMinor&` and `_BuildNumber&`.

Selection colour `_ColSelection&`

the colour with which to draw selected items, expressed as a long integer comprising Red, Green, and Blue (RGB) components.

The valid range for an RGB colour is 0 to 16 777 215 (&HFFFFFF& in hexadecimal). The high byte of a number in this range equals 0. The lower 3 bytes, from least to most significant byte, determine the amount of red, green, and blue, respectively. The red, green, and blue components are each represented by a number between 0 and 255 (&HFF).

Coordinate classes `_CoordClasses$:`

a space-separated list of available co-ordinate classes. See Co-ordinate Grids for details.

Current document name `_CurSwdName$`

the file name of the current document (SWD)

Current window handle `_CurWndHandle&`

the window handle of the current MDI child window

Default nol `_DefaultNol$`

the name of the default named object library

Execution error `_ExecError&`

the error code returned from the last API method used. See the automatically generated programming file (GisLink.bas or SisConst.h) for a list. Cadcorp SIS Control applications can call `GetErrorString` to get a string equivalent of the error code.

Greeking height `_GreekHeight#`

the greeking height, in pixels, below which text will not be drawn

Hot snapping `_HotSnap&`

Should mouse movements detect underlying graphics (snapcodes and MapTips)? This is known as 'hot snapping'.

Valid values

0 hot snapping off

1 hot snapping on (not including cursor-based datasets)

2 no hot snapping options

3 hot snapping on (including cursor-based datasets)

Licence number `_LicenceNumber$`

the Cadcorp SIS licence number. If this system variable is set (using the `SetStr` method, or the `-LN:` command-line option), the value will be used, irrespective of its validity, and any other licence numbers in the system registry will be ignored.

Main window title `_MainWndTitle$`

the window title of the main window. This string can be used for setting focus from GisLink customisations.

Netwrk dongle server `_NetDongleServer$`

the network dongle server name or IP address

Network dongle timeout `_NetDongleTimeout&`

the network dongle timeout period (in minutes)

Number of arguments `_NumArg&`

the total number of arguments entered by the user. This system variable is used by the GisLink methods `GetPos` and `GetPosEx`.

Maximum texture size `_OpenGLMaxTextureSize&`

the maximum texture size used when draping in the 3D Window

Position bar co-ordinate class `_PosBarCoordClass$`

the co-ordinate class of the position bar. See Co-ordinate Grids for details.

Tracking `_PosBarTracking&`

Is the position bar tracking mouse movements?

Valid values

True or False

Printer colour capabilities `_PrintColour&`

the printer colour capabilities. This controls the colours that Cadcorp SIS generates when composing printer output.

By default, Cadcorp SIS asks the installed printer driver what the colour capabilities of the printer are. Cadcorp SIS will then generate graphics, which use these colours.

However, some printer drivers do not know, or mis-report, the colour capabilities of some printers. When this happens, the output can be black and white on a colour printer, or colours can be invisible on a black and white printer.

Valid values

`SIS_PRINTCAPS_QUERY`	query the printer driver to get the colour capabilities of the printer
`SIS_PRINTCAPS_MONO`	force the output to monochrome (pens are black or white, brushes and bitmaps are grey)
`SIS_PRINTCAPS_COLOUR`	assume the printer can handle 24-bit colours

Roamer shape `_RoamerShape&`

selects the shape of the magnifying glass displayed by the **Map>Zoom>Roamer** command

Valid values

0 circular

1 square

Rubber band anchor `_RubberBandAnchor$`

the anchor position for a rubber-band operation. The anchor position depends on the rubber-band mode as follows:

`SIS_RUBBERBAND_NONE`	the anchor position is ignored
`SIS_RUBBERBAND_LINE`	the anchor position is the start of the line
`SIS_RUBBERBAND_CIRCLE`	the anchor position is the centre of the circle
`SIS_RUBBERBAND_RECT`	the anchor position is one corner of the rectangle
`SIS_RUBBERBAND_ITEMS`	the anchor position is the origin for the item drag

This system variable is used by the GisLink methods `GetPos` and `GetPosEx`.

Rubber band fix height `_RubberBandFixHeight#`

the height of a fixed size rubber-band rectangle to draw when rubber-banding with `SIS_RUBBERBAND_RECT` mode

Rubber band fix radius `_RubberBandFixRadius#`

the radius of a fixed size rubber-band circle to draw when rubber-banding with `SIS_RUBBERBAND_CIRCLE` mode

Rubber band fix width `_RubberBandFixWidth#`

the width of a fixed size rubber-band rectangle to draw when rubber-banding with `SIS_RUBBERBAND_RECT` mode

Rubber band list `_RubberBandList$`

the named list containing items to draw when rubber-banding with `SIS_RUBBERBAND_ITEMS` mode. This system variable is used by the GisLink methods `GetPos` and `GetPosEx`.

Rubber band mode `_RubberBandMode&`

the current rubber-banding mode

Valid values

`SIS_RUBBERBAND_NONE`	do not rubber-band
`SIS_RUBBERBAND_LINE`	rubber-band a straight line
`SIS_RUBBERBAND_CIRCLE`	rubber-band a circle
`SIS_RUBBERBAND_RECT`	rubber-band a rectangle
`SIS_RUBBERBAND_ITEMS`	drag items from a named list
`SIS_RUBBERBAND_FIX_CIRCLE`	rubber-band a fixed size circle
`SIS_RUBBERBAND_FIX_RECT`	rubber-band a fixed size rectangle

This system variable is used by the GisLink methods `GetPos` and `GetPosEx`.

Select read-only `_SelectRO&`

Is the default selection command read-only? This controls whether the **Edit>Select** command allows items to be edited.

By default, the **Edit>Select** command shows grab handles on editable items. Some applications/customisations of Cadcorp SIS may wish to suppress these features.

Valid values

True	the Edit/Select command will be read-only
False	the Edit/Select command will allow editable items to be edited

Show menu `_ShowMenu&`

Should the main menu be shown?

Valid values

True or False

Snap code `_SnapCode&`

the type of geometry snapped to by the last snap in the current map window

Snap key `_SnapKey&`

the keyboard key or mouse button used for the last position entered by the user. The keyboard key values are upper case ASCII values. The mouse button values are:

1 Left

4 Middle

2 Right

Snapped position `_SnapPos$`

the last position entered by the user. This could be by a mouse snap, or a position entered by typing in the position bar. This system variable is used by the GisLink methods `GetPos` and `GetPosEx`.

Snap tolerance `_SnapTolerance&`

the tolerance to use for future snapping, measured in screen pixels

Splinter value `_Splinter#`

the default topological splinter value. When creating topology, topological items with a higher splinter value will be merged.

Total dataset size `_TotalDtsSize&`

the total amount of memory, in bytes, that all the datasets use

Type of argument `_TypeArg&`

the type of the last user argument

Valid values

`SIS_ARG_ESCAPE`	the escape key was pressed or another command was started
`SIS_ARG_ENTER`	the enter key was pressed
`SIS_ARG_BACKSPACE`	the backspace key was pressed
`SIS_ARG_POSITION`	a position was entered

This system variable is used by the GisLink methods `GetPos` and `GetPosEx`.

Major version `_VersionMajor&`

the Cadcorp SIS major version number. The complete Cadcorp SIS version number can be generated by concatenating `_VersionMajor&`, `_VersionMinor&`, and `_BuildNumber&`.

Minor version `_VersionMinor&`

the Cadcorp SIS minor version number. The complete Cadcorp SIS version number can be generated by concatenating `_VersionMajor&`, `_VersionMinor&`, and `_BuildNumber&`.

Wait bar `_WaitBar&`

Should the wait bar and cancel button be shown during long operations?

Valid values

True the wait bar and cancel buttons will be shown

False the wait bar and cancel buttons will not be shown

■ Theme properties (SIS_OT_THEME)

Cadcorp SIS uses named theme objects to control the display of items (such as their brush, pen, and shape) depending on item properties and also to annotate items, eg with bar charts or pie charts.

◆ Bar charts

The bar chart theme annotates themed items by drawing a bar chart at the item origin.

Alignment `_align&`

the alignment of annotation created by bar charts or pie charts theme objects around the item origin

Valid values

```
SIS_ALIGN_BOTTOM_LEFT
SIS_ALIGN_BOTTOM_CENTRE
SIS_ALIGN_BOTTOM_RIGHT
SIS_ALIGN_MIDDLE_LEFT
SIS_ALIGN_MIDDLE_CENTRE
SIS_ALIGN_MIDDLE_RIGHT
SIS_ALIGN_TOP_LEFT
SIS_ALIGN_TOP_CENTRE
SIS_ALIGN_TOP_RIGHT
```

These constants are not the same as the text alignment constants, because the vertical text alignment option `SIS_BASELINE` has no equivalent for theme alignment.

Bold `_text_bold&`

Should the legend/label text be drawn bold?

Valid values

True or False

Brush `_brush$`

the brush used for the legend/label text backdrop, if the legend/label text is opaque

Class `_classTheme$`

the class of the theme

Disabled `_bDisabled&`

Should theme graphics be hidden?

Valid values

True or False

Font `_font$`

the font used for the legend/label text

Height `_height&`

the height in mm of annotation items at the theme 'sum' value: Value for Bar Charts.

Horizontal `_bHorizontal&`

Should the Bar Charts theme annotation be drawn horizontally, instead of vertically?

Valid values

True or False

Italic `_text_italic&`

Should the legend/label text be drawn italicised?

Valid values

True or False

Maximum scale `_scalemax#`

the highest reproduction scale at which theme annotation graphics will be drawn. If you set this value to 1000 and then zoom out to 1:2000, the theme annotation will become invisible.

Minimum scale `_scalemin#`

the lowest reproduction scale at which theme annotation graphics will be drawn. If you set this value to 100 and then zoom in to 1:50, the theme annotation will become invisible.

Number of Bar blocks `_nBlocks&`

the number of blocks in a Bar Charts theme

Opaque `_text_opaque&`

Should the legend/label text be drawn opaque?

If this value is True for legend text, the legend will draw with a surrounding box drawn with the legend pen and filled with the legend brush.

If this value is True for label text, the labels will appear with a filled background, drawn using the label brush.

Valid values

True or False

Pen `_pen$`

the pen used to draw the legend/label text

Point height `_point_height&`

the point height of the legend/label text

Scaling function `_function&`

the mathematical function used to calculate the height of annotation items, eg a Pie Charts theme

Valid values

`SIS_FUNCTION_CONSTANT`	all annotation items are the same size
`SIS_FUNCTION_SQUAREROOT`	annotation items are scaled by the square root of their value
`SIS_FUNCTION_LINEAR`	annotation items are scaled linearly by their value
`SIS_FUNCTION_LOG10`	annotation items are scaled logarithmically (base-10) by their value, if the value is above the theme sum, and linearly, if the value is below the theme sum

Sub-title `_subtitle$`

the legend sub-title

Title `_title$`

the theme title

Underlined `_text_underlined&`

Should the legend/label text be drawn underlined?

Valid values

True or False

Value `_value#`

the value at which a Bar Charts block will be drawn at the theme height value: Value for Bar Charts

Width `_width&`

the width, in mm, of a block on a Bar Charts theme

◆ Contour

a theme which annotates TIN items with contour lines

Bold `_text_bold&`

Should the legend/label text be drawn bold?

Valid values

True or False

Brush `_brush$`

the brush used for the legend/label text backdrop, if the legend/label text is opaque

Class `_classTheme$`

the class of the theme

Disabled `_bDisabled&`

Should theme graphics be hidden?

Valid values

True or False

Font `_font$`

the font used for the legend/label text

Italic `_text_italic&`

Should the legend/label text be drawn italicised?

Major height `_height_major#`

the height, in metres, of the major contour lines

Maximum scale `_scalemax#`

the highest reproduction scale at which theme annotation graphics will be drawn. If you set this value to 1000 and then zoom out to 1:2000, the theme annotation will become invisible.

Minimum scale `_scalemin#`

the lowest reproduction scale at which theme annotation graphics will be drawn. If you set this value to 100 and then zoom in to 1:50, the theme annotation will become invisible.

Minor height `_height_minor#`

the height, in metres, of the minor contour lines

Opaque `_text_opaque&`

Should the legend/label text be drawn opaque?

If this value is True for legend text, the legend will draw with a surrounding box drawn with the legend pen and filled with the legend brush.

If this value is True for label text, the labels will appear with a filled background, drawn using the label brush.

Valid values

True or False

Pen `_pen$`

the pen used to draw the legend/label text

Pen for Major lines `_penMajor$`

the pen used to draw the major contour lines

Pen for Minor lines `_penMinor$`

the pen used to draw the minor contour lines

Places `_places&`

the number of decimal places which a dimension is reported in on a legend

Point height `_point_height&`
the point height of the legend/label text

Sub-title `_subtitle$`
the legend sub-title

Title `_title$`
the theme title

Underlined `_text_underlined&`
Should the legend/label text be drawn underlined?

Valid values

True or False

Units `_units&`
the units that the dimension should be displayed in on a legend

◆ Dot Density

The Dot Density theme annotates 2-dimensional items such as areas and polygons with a number of dots. Each dot can be given a shape, or be represented by a single pixel.

Bold `_text_bold&`
Should the legend/label text be drawn bold?

Valid values

True or False

Brush `_brush$`
the brush used for the legend/label text backdrop, if the legend/label text is opaque

Class `_classTheme$`
the class of the theme

Disabled `_bDisabled&`
Should theme graphics be hidden?

Valid values

True or False

Dot brush `_dot_brush$`
the brush used to draw the theme shape

Dot pen `_dot_pen$`
the pen used to draw the theme shape

Dot shape `_dot_shape$`
the shape used to annotate the filled area

Font `_font$`

the font used for the legend/label text

Formula `_formula$`

the value used to calculate the number of dots

Italic `_text_italic&`

Should the legend/label text be drawn italicised?

Maximum scale `_scalemax#`

the highest reproduction scale at which theme annotation graphics will be drawn. If you set this value to 1000 and then zoom out to 1:2000, the theme annotation will become invisible.

Minimum scale `_scalemin#`

the lowest reproduction scale at which theme annotation graphics will be drawn. If you set this value to 100 and then zoom out to 1:50, the theme annotation will become invisible.

Opaque `_text_opaque&`

Should the legend/label text be drawn opaque?

If this value is True for legend text, the legend will draw with a surrounding box drawn with the legend pen, and filled with the legend brush.

If this value is True for label text, the labels will appear with a filled background, drawn using the label brush.

Valid values

True or False

Pen `_pen$`

the pen used to draw the legend/label text

Point height `_point_height&`

the point height of the legend/label text

Sub-title `_subtitle$`

the legend sub-title

Title `_title$`

the theme title

Underlined `_text_underlined&`

Should the legend/label text be drawn underlined?

Valid values

True or False

Value `_value#`

the value at which a Bar Charts block, or a Graduated shape, will be drawn at the theme height value: value for Bar Charts or point height for Graduated

◆ Extrude 2D items in 3D views

This theme extrudes 2D items in a 3D window, using a formula which is evaluated for each themed item.

Class `_classTheme$`

the class of the theme

Disabled `_bDisabled&`

Should theme graphics be hidden?

Valid values

True or False

Formula `_formula$`

the extrusion height

Title `_title$`

the theme title

◆ Flow

This theme annotates TIN items with flow direction arrows.

Bold `_text_bold&`

Should the legend/label text be drawn bold?

Valid values

True or False

Brush `_brush$`

the brush used for the legend/label text backdrop, if the legend/label text is opaque

Class `_classTheme$`

the class of the theme

Disabled `_bDisabled&`

Should theme graphics be hidden?

Valid values

True or False

Flow brush `_flow_brush$`

the brush used to draw the theme shape

Flow pen `_flow_pen$`

the pen used to draw the theme shape

Flow shape `_flow_shape$`

the shape used to annotate flow direction

Font `_font$`

the font used for the legend/label text

Italic `_text_italic&`

Should the legend/label text be drawn italicised?

Valid values

True or False

Maximum gradient `_gradientmax#`

the maximum gradient which the Flow theme should annotate

Maximum scale `_scalemax#`

the highest reproduction scale at which theme annotation graphics will be drawn. If you set this value to 1000 and then zoom out to 1:2000, the theme annotation will become invisible.

Minimum gradient `_gradientmin#`

the minimum gradient which the Flow theme should annotate

Minimum scale `_scalemin#`

the lowest reproduction scale at which theme annotation graphics will be drawn. If you set this value to 100 and then zoom in to 1:50, the theme annotation will become invisible.

Opaque `_text_opaque&`

Should the legend/label text be drawn opaque?

If this value is True for legend text, the legend will draw with a surrounding box drawn with the legend pen and filled with the legend brush.

If this value is True for label text, the labels will appear with a filled background, drawn using the label brush.

Valid values

True or False

Pen `_pen$`

the pen used to draw the legend/label text

Point height `_point_height&`

the point height of the legend/label text

Sub-title `_subtitle$`

the legend sub-title

Title `_title$`

the theme title

Underlined `_text_underlined&`

Should the legend/label text be drawn underlined?

Valid values

True or False

◆ Graduated

This theme annotates themed items by drawing a shape, scaled around a value, at the item's origin.

Bold `_text_bold&`

Should the legend/label text be drawn bold?

Valid values

True or False

Brush `_brush$`

the brush used for the legend/label text backdrop if the legend/label text is opaque

Class `_classTheme$`

the class of the theme

Disabled `_bDisabled&`

Should theme graphics be hidden?

Valid values

True or False

Font `_font$`

the font used for the legend/label text

Formula `_formula$`

the value used to display size of graduation

Graduated brush `_graduated_brush$`

the brush used to fill any areas in the annotation shape. Use "" to get the default brush of the shape

Graduated pen `_graduated_pen$`

the pen used to draw the annotation shape. Use "" to get the default pen of the shape

Graduated shape `_graduated_shape$`

the shape used to annotate the overlay items

Italic `_text_italic&`

Should the legend/label text be drawn italicised?

Valid values

True or False

Maximum scale `_scalemax#`

the highest reproduction scale at which theme annotation graphics will be drawn. If you set this value to 1000 and then zoom out to 1:2000, the theme annotation will become invisible.

Minimum scale `_scalemin #`

the lowest reproduction scale at which theme annotation graphics will be drawn. If you set this value to 100 and then zoom in to 1:50 the theme annotation will become invisible.

Opaque `_text_opaque&`

Should the legend/label text be drawn opaque?

If this value is True for legend text, the legend will draw with a surrounding box drawn with the legend pen, and filled with the legend brush.

If this value is True for label text, the labels will appear with a filled background, drawn using the label brush.

Valid values

True or False

Pen `_pen$`

the pen used to draw the legend/label text

Point height `_point_height&`

the point height of the legend/label text

Scaling function `_function&`

the mathematical function used to calculate the height of annotation items, eg a Pie Charts theme

Valid values

`SIS_FUNCTION_CONSTANT`	all annotation items are the same size
`SIS_FUNCTION_SQUAREROOT`	annotation items are scaled by the square-root of their value
`SIS_FUNCTION_LINEAR`	annotation items are scaled linearly by their value
`SIS_FUNCTION_LOG10`	annotation items are scaled logarithmically (base -10) by their value, if the value is above the theme sum, and linearly, if the value is below the theme sum

Sub-title `_subtitle$`

the legend sub-title

Title `_title$`

the theme title

Underlined `_ text_underlined&`

Should the legend/label text be drawn underlined?

Valid values

True or False

Value `_value#`

the value at which Graduated shape will be drawn at the theme height value: point Height for Graduated

◆ Individual Values

This theme styles items (changes their brush, pen, and shape, and so on) by matching the result of a formula against a list of known values.

Bold `_text_bold&`

Should the legend/label text be drawn bold?

Valid values

True or False

Brush `_brush$`

the brush used for the legend/label text backdrop if the legend/label text is opaque

Class `_classTheme$`

the class of the theme

Disabled `_bDisabled&`

Should theme graphics be hidden?

Valid values

True or False

Font `_font$`

the font used for the legend/label text

Formula `_formula$`

the value used to match against known values

Italic `_text_italic&`

Should the legend/label text be drawn italicised?

Valid values

True or False

Number of values `_nValues&`

the number of values in an Individual Values theme

Opaque `_text_opaque&`

Should the legend/label text be drawn opaque?

If this value is True for legend text, the legend will draw with a surrounding box drawn with the legend pen and filled with the legend brush.

If this value is True for label text, the labels will appear with a filled background, drawn using the label brush.

Valid values

True or False

Pen `_pen$`

the pen used to draw the legend/label text

Point height `_point_height&`

the point height of the legend/label text

Sub-title `_subtitle$`

the legend sub-title

Title `_title$`

the theme title

Underlined `_text_underlined&`

Should the legend/label text be drawn underlined?

Valid values

True or False

◆ Labels

This theme annotates themed items by drawing a text label at the item origin. If you want to move the labels, you can explode the theme which will make the labels editable.

Bold `_text_bold&`

Should the legend/label text be drawn bold?

Valid values

True or False

Box `_text_box&`

Should the label text be drawn with a surrounding box?

Valid values

True or False

Brush `_brush$`

the brush used for the legend/label text backdrop, if the legend/label text is opaque

Class `_classTheme$`

the class of the theme

Disabled `_bDisabled&`

Should theme graphics be hidden?

Valid values

True or False

Fixed height `_bFixed&`

Is the text a fixed height on screen?

Valid values

True or False

Font `_font$`

the font used for the legend/label text

Formula `_formula$`

the label text

Hide overlapping `_bHideOverlap&`

Should overlapping labels be skipped?

Valid values

True or False

Horizontal alignment `_text_alignH&`

the horizontal alignment of the label text about the origin of the labelled item

Valid values

```
SIS_LEFT
SIS_CENTRE
SIS_RIGHT
```

Italic `_text_italic&`

Should the legend/label text be drawn italicised?

Valid values

True or False

Label offset `_offset&`

offset used by Labels theme, in text points

Label placement `_placement&`

placement option for labelling items with line geometry

Valid values

`SIS_LABEL_START` Place labels at the start.

`SIS_LABEL_MIDDLE` Place labels at the middle.

`SIS_LABEL_END` Place labels at the end.

`SIS_LABEL_ALONG` Place labels along the line geometry, ie like line text.

When the `SIS_LABEL_ALONG` is selected, and all of the label text fits onto the line, the label text will be justified according to the label text horizontal justification, eg at the start for `SIS_ALIGN_LEFT` and the end for `SIS_ALIGN_RIGHT`. If the text does not fit, the label text will be placed according to the label text's horizontal alignment.

Maximum scale `_scalemax#`

the highest reproduction scale at which theme annotation graphics will be drawn. If you set this value to 1000 and then zoom out to 1:2000, the theme annotation will become invisible.

Minimum scale `_scalemin#`

the lowest reproduction scale at which theme annotation graphics will be drawn. If you set this value to 100 and then zoom in to 1:50, the theme annotation will become invisible.

Opaque `_text_opaque&`

Should the legend/label text be drawn opaque?

If this value is True for legend text, the legend will draw with a surrounding box drawn with the legend pen and filled with the legend brush.

If this value is True for label text, the labels will appear with a filled background, drawn using the label brush.

Valid values

True or False

Pen `_pen$`

the pen used to draw the legend/label text

Point height `_point_height&`

the point height of the legend/label text

Rotate labels `_bRotate&`

Should labels be rotated to match items?

Valid values

True or False

Sub-title `_subtitle$`

the legend sub-title

Title `_title$`

the theme title

Underlined `_text_underlined&`

Should the legend/label text be drawn underlined?

Valid values

True or False

Vertical alignment `_text_alignV&`

the vertical alignment of the label text about the origin of the labelled item

Valid values

```
SIS_TOP
SIS_MIDDLE
SIS_BASELINE
SIS_BOTTOM
```

◆ Pie Charts

This theme which annotates themed items by drawing a Pie Chart at the item origin.

Alignment: `_align&`

the alignment of annotation created by Pie Charts theme objects around the item origin

Valid values

```
SIS_ALIGN_BOTTOM_LEFT
SIS_ALIGN_BOTTOM_CENTRE
SIS_ALIGN_BOTTOM_RIGHT
SIS_ALIGN_MIDDLE_LEFT
SIS_ALIGN_MIDDLE_CENTRE
SIS_ALIGN_MIDDLE_RIGHT
SIS_ALIGN_TOP_LEFT
SIS_ALIGN_TOP_CENTRE
SIS_ALIGN_TOP_RIGHT
```

These constants are not the same as the text alignment constants, because the vertical text alignment option `SIS_BASELINE` has no equivalent for theme alignment.

Bold `_text_bold&`

Should the legend/label text be drawn bold?

Valid values

True or False

Brush `_brush$`

the brush used for the legend/label text backdrop, if the legend/label text is opaque

Class `_classTheme$`

the class of the theme

Disabled `_bDisabled&`

Should theme graphics be hidden?

Valid values

True or False

Font `_font$`

the font used for the legend/label text

Height `_height&`

the height in mm of annotation items at the theme "sum" value: Sum for Pie Charts

Italic `_text_italic&`

Should the legend/label text be drawn italicised?

Valid values

True or False

Maximum scale `_scalemax#`

the highest reproduction scale at which theme annotation graphics will be drawn. If you set this value to 1000 and then zoom out to 1:2000, the theme annotation will become invisible.

Minimum scale `_scalemin#`

the lowest reproduction scale at which theme annotation graphics will be drawn. If you set this value to 100 and then zoom in to 1:50, the theme annotation will become invisible.

Number of Pie slices `_nSlices&`

the number of slices

Opaque `_text_opaque&`

Should the legend/label text be drawn opaque?

If this value is True for legend text, the legend will draw with a surrounding box drawn with the legend pen and filled with the legend brush.

If this value is True for label text, the labels will appear with a filled background, drawn using the label brush.

Valid values

True or False

Pen `_pen$`

the pen used to draw the legend/label text

Point height `_point_height&`

the point height of the legend/label text

Scaling function `_function&`

the mathematical function used to calculate the height of annotation items

Valid values

`SIS_FUNCTION_CONSTANT`	all annotation items are the same size
`SIS_FUNCTION_SQUAREROOT`	annotation items are scaled by the square-root of their value
`SIS_FUNCTION_LINEAR`	annotation items are scaled linearly by their value
`SIS_FUNCTION_LOG10`	annotation items are scaled logarithmically (base-10) by their value, if the value is above the theme sum, and linearly, if the value is below the theme sum.

Sub-title `_subtitle$`

the legend sub-title

Sum `_sum#`

the value at which Pie Charts will be drawn at the Height value

Title `_title$`

the theme title

Underlined `_text_underlined&`

Should the legend/label text be drawn underlined?

Valid values

True or False

◆ Ranges

This theme styles items (changes their brush, pen, and shape, for example) by matching the result of a formula against a range of known values.

Bold `_text_bold&`

Should the legend/label text be drawn bold?

Valid values

True or False

Brush `_brush$`

the brush used for the legend/label text backdrop, if the legend/label text is opaque

Valid values

True or False

Class `_classTheme$`

the class of the theme

Disabled `_bDisabled&`

Should theme graphics be hidden?

Valid values

True or False

Font `_font$`

the font used for the legend/label text

Formula `_formula$`

the value used to match against known ranges

Italic `_text_italic&`

Should the legend/label text be drawn italicised?

Valid values

True or False

Number of ranges `_nRanges&`

the number of ranges

Opaque `_text_opaque&`

Should the legend/label text be drawn opaque?

If this value is True for legend text, the legend will draw with a surrounding box drawn with the legend pen and filled with the legend brush.

If this value is True for label text, the labels will appear with a filled background, drawn using the label brush.

Valid values

True or False

Pen `_pen$`

the pen used to draw the legend/label text

Point height `_point_height&`

the point height of the legend/label text

Sub-title `_subtitle$`

the legend sub-title

Title `_title$`

the theme title

Underlined `_text_underlined&`

Should the legend/label text be drawn underlined?

Valid values

True or False

◆ Relief

This theme styles grid items, assigning different colours to different grid values. When the grid is a DTM the colours can represent ranges of heights.

Bold `_text_bold&`

Should the legend/label text be drawn bold?

Valid values

True or False

Brush `_brush$`

the brush used for the legend/label text backdrop, if the legend/label text is opaque

Valid values

True or False

Class `_classTheme$`

the class of the theme

Disabled `_bDisabled&`

Should theme graphics be hidden?

Valid values

True or False

Font `_font$`

the font used for the legend/label text

Italic `_text_italic&`

Should the legend/label text be drawn italicised?

Opaque `_text_opaque&`

Should the legend/label text be drawn opaque?

If this value is True for legend text, the legend will draw with a surrounding box drawn with the legend pen and filled with the legend brush.

If this value if True for label text, the labels will appear with a filled background, drawn using the label brush.

Valid values

True or False

Pen `_pen$`

the pen used to draw the legend/label text

Point height `_point_height&`

the point height of the legend/label text

Sub-title `_subtitle$`

the legend sub-title

Title `_title$`

the theme title.

Underlined `_text_underlined&`

Should the legend/label text be drawn underlined?

Valid values

True or False

◆ Static theme

Static themes are created by exploding an existing annotation theme, such as a Bar Charts theme. Exploding an annotation theme creates an internal overlay containing all of the annotation graphics. Exploding a theme is useful because it lets you move and edit the annotation graphics. However, once a theme is exploded, it is no longer associative or dynamic, so editing the data on the underlying items will not change the annotation graphics. The static theme remembers the Legend at the time of the explosion.

Class `_classTheme$`

the class of the theme

Exploded class `_classExploded$`

the class of theme which was exploded to make a Static theme

◆ Topology

This theme annotates link/node topology which has certain characteristics. The Topology theme is useful for checking many aspects of link/node networks.

Annotate automatic nodes `_bUseShapeAuto&`

Should automatic node items be annotated?

Valid values

True or False

Annotate blocked links `_bUseShapeBlocked&`

Should blocked link items be annotated?

Valid values

True or False

Annotate edge nodes `_bUseShapeEdge&`

Should edge node items be annotated?

Valid values

True or False

Annotate junctions `_bUseShapeJunction&`

Should node items with turning rules be annotated?

Valid values

True or False

Annotate non-automatic nodes `_bUseShapeNonAuto&`

Should non-automatic node items be annotated?

Valid values

True or False

Annotate one-way links `_bUseShapeDirection&`

Should one-way link items be annotated?

Valid values

True or False

Annotate trailing links `_bUseShapeSingle&`

Should dangling node items be annotated?

Valid values

True or False

Automatic node shape `_shapeAuto$`

the shape to use for automatic node items

Blocked link shape `_shapeBlocked$`

the shape to use for blocked link items

Bold `_text_bold&`

Should the legend/label text be drawn bold?

Valid values

True or False

Brush `_brush$`

the brush used for the legend/label text backdrop, if the legend/label text is opaque

Class `_classTheme$`

the class of the theme

Disabled `_bDisabled&`

Should theme graphics be hidden?

Valid values

True or False

Edge node shape `_shapeEdge$`

the shape to use for edge node items

Fixed height `_bFixed&`

Are the shape objects a fixed height on screen?

Valid values

True or False

Font `_font$`

the font used for the legend/label text

Italic `_text_italic&`

Should the legend/label text be drawn italicised?

Valid values

True or False

Junction shape `_shapeJunction$`

the shape to use for node items with turning rules

Maximum scale `_scalemax#`

the highest reproduction scale at which theme annotation graphics will be drawn. If you set this value to 1000 and then zoom out to 1:2000, the theme annotation will become invisible.

Minimum scale `_scalemin#`

the lowest reproduction scale at which theme annotation graphics will be drawn. If you set this value to 100 and then zoom in to 1:50 the theme annotation will become invisible.

Non-automatic node shape `_shapeNonAuto$`

the shape to use for non-automatic node items

One-way link shape `_shapeDirection$`

the shape to use for one-way link items

Opaque `_text_opaque&`

Should the legend/label text be drawn opaque?

If this value is True for legend text, the legend will draw with a surrounding box drawn with the legend pen and filled with the legend brush.

If this value is True for label text, the labels will appear with a filled background, drawn using the label brush.

Valid values

True or False

Pen `_pen$`

the pen used to draw the legend/label text

Point height `_point_height&`

the point height of the legend/label text

Sub-title `_subtitle$`

the legend sub-title

Title `_title$`

the theme title

Trailing link shape `_shapeSingle$`

the shape to use for dangling node items

Underlined `_text_underlined&`

Should the legend/label text be drawn underlined?

Valid values

True or False

■ Theme component properties

◆ SIS_OT_THEMECOMPONENT

Brush `_brush$`

the brush used to fill the theme component, eg a slice in a Pie Charts theme, or items which match the theme component. Use "" to avoid overriding the item brush.

Class `_classTheme$`

the class of the theme

Disabled `_bDisabled&`

Should theme graphics be hidden?

Valid values

True or False

Formula `_ formula$`

The formula used to calculate a value for the theme component, eg a block in a Bar Charts theme, or a slice in a Pie Charts theme. The formula must evaluate to a number.

Pen `_pen$`

the pen used to draw the theme component, eg a slice in a Pie Charts theme, or items which match the theme component. Use "" to avoid overriding the item pen.

Shape `_shape$`

the shape used for point items which match the theme component. Use "" to not override the point shape.

Title `_title$`

the title of a theme component for display in the theme legend

Value `_value#`

a range value in Ranges and Contour themes

■ Window properties (SIS_OT_WINDOW)

Cadcorp SIS windows have properties which are a combination of the properties of a Cadcorp SIS window and the SWD of the window. In a Cadcorp SIS application, the window is equivalent to a single child window of the main frame window. In the Cadcorp SIS Control the window is equivalent to the control itself.

◆ Map window

Allow keyboard moves `_bKeyboardMove&`

Should keyboard keys (eg +, *, arrow keys) change the view?

Valid values

True or False

Allow keyboard snaps `_bKeyboardSnap&`

Should keyboard snaps be allowed?

Valid values

True or False

Class name `_class$`

the class of window

Valid values

`"SisWndDisplay"`

Display angle `_displayAngle#`

the angle of the view, in radians. A normal, unrotated view will have a display angle of 0.0.

Display scale `_displayScale#`

the display scale. Setting the display scale will zoom the view to the given scale.

Redraw `_bRedraw&`

Should modified items and overlays be redrawn immediately? By default, as soon as an item is created, deleted, or modified, the window graphics are updated. But by setting this value to False you can make Cadcorp SIS skip the redraw operation. You can still use the `Redraw` API method to force Cadcorp SIS to redraw the window.

Valid values

True or False

Scroll bars `_bScrollBars&`

Should the window have horizontal and vertical scrollbars for fast map scrolling?

Valid values

True or False

Style padding `_style_pad&`

the style padding, specified in style points. When any portion, or all, of a window is being painted, the style padding will be added to the area being painted. By setting the screen padding you can make point, text items and so on redraw, even if their origin does not appear in the paint region. This should lead to cleaner painting while panning.

View Scale `_viewScale#`

the view scale. Changing the view scale has no effects on the graphics you can see on the screen. However, when you use the **File>Print**... and **File>Print Template>Wizard...** commands, the view scale is used to reduce the graphics down to fit onto the paper used for printing.

◆ 3D window

Allow keyboard moves `_bKeyboardMove&`

Should keyboard keys (eg +, *, arrow keys) change the view?

Valid values

True or False

Allow mouse moves `_bMouseMove&`

Should mouse interaction change the view?

Valid values

True or False

Angle `_angleDeg#`

the 3D view angle, in degrees

Class name `_class$`

the class of window

Valid values

`SisWndOgl`

Cone angle `_coneDeg#`

the 3D-view cone angle, in degrees

Draw smooth surfaces `_bQualitySmooth&`

Should the window ‘smooth’ surface items?

Valid values

True or False

Draw textured surfaces `_bQualityTexture&`

Should the window draw textures, eg the drape bitmap?

Valid values

True or False

Draw wire-line `_bQualityWireLine&`

Should the window draw in wireline instead of solid?

Valid values

True or False

Elevation angle `_elevDeg#`

the 3D view elevation angle, in degrees

Exaggeration factor `_exaggerateZ#`

the exaggeration factor to apply to 3D geometry. A factor of greater than 1.0 will make 3D geometry stretch in Z and a factor of less than 1.0 will make 3D geometry shrink in Z.

Generalisation `_generalise&`

the 3D window generalisation factor for facetting curved lines, with a range 0 (rough) to 255 (smooth)

Mode `_mode&`

the interaction mode of a 3D window

Valid values

`SIS_3DMODE_CRUISE`	cruise mode (combination of zoom and rotation of the look point around the eye point)
`SIS_3DMODE_EYE`	eye mode (rotations move the look point around the eye point)
`SIS_3DMODE_MODEL`	model mode (rotations move the eye point around the look point)
`SIS_3DMODE_PAN`	pan mode (steps move both the eye and the look point)
`SIS_3DMODE_ZOOM`	zoom mode (steps move the eye point only)

Scroll bars `_bScrollBars&`

Should the window have horizontal and vertical scrollbars for fast map scrolling?

Valid values

True or False

◆ Table window

Class name `_class$`

the class of window

Valid values

`SisWndTable`

Overlay `_nOverlayShow&`

the index in the list of SWD overlays of the overlay being viewed in a table window.

◆ Common window properties

Allow local menu `_bPopupMenu&`

Can the local menu be used?

Valid values

True or False

Attributes `_attributes$`

the user defined attributes stored with the SWD

Default overlay `_nDefaultOverlay&`

the current default overlay in the SWD of the window. The default overlay is the overlay on which items are created, provided the overlay is editable.

Modified `_bModified&`

Has the SWD been modified?

This flag tracks changes only to the SWD itself, eg adding and removing of overlays or setting overlay properties. It does not track changes to the datasets of the overlays it contains, each of which has its own flag, eg Modified.

Saving an SWD will cause any file-based datasets to be saved first.

Valid values

True or False

Number of overlays `_nOverlay&`

the number of overlays in the SWD of the window

Size `_size&`

the total amount of memory, in bytes, that the datasets in the SWD use

Template `_bTemplate#`

Is this a saved window template?

Appendix 5

Global constants

■ Introduction 411
■ Global constants 411
■ Old global constants 417

■ Introduction

This appendix lists the global constants in Cadcorp SIS.

■ Global constants

Horizontal text alignment

SIS_LEFT	0
SIS_RIGHT	2
SIS_CENTRE	6

Vertical text alignment

SIS_TOP	0
SIS_BOTTOM	8
SIS_BASELINE	24
SIS_MIDDLE	72

Horizontal and vertical text alignment

SIS_TOP_LEFT	0
SIS_TOP_RIGHT	2
SIS_TOP_CENTRE	6
SIS_BOTTOM_LEFT	8
SIS_BOTTOM_RIGHT	10
SIS_BOTTOM_CENTRE	14
SIS_BASE_LEFT	24
SIS_BASE_RIGHT	26
SIS_BASE_CENTRE	30
SIS_MIDDLE_LEFT	72
SIS_MIDDLE_RIGHT	74
SIS_MIDDLE_CENTRE	78

Menu commands

SIS_COM_ALL	0
SIS_COM_ADD	1
SIS_COM_REMOVE	2
SIS_COM_NONE	3

Overlay status

SIS_INVISIBLE	0
SIS_VISIBLE	1

SIS_HITTABLE	2
SIS_EDITABLE	3

Redraw

SIS_CURRENTWINDOW	0
SIS_CURRENTSWD	1
SIS_ALLWINDOWS	2
SIS_QUICK_REDRAW	32

Attribute filters

SIS_FILTERRESET	0
SIS_FILTERADD	1
SIS_FILTERREMOVE	2

Argument entry

SIS_ARG_ESCAPE	0
SIS_ARG_ENTER	1
SIS_ARG_BACKSPACE	2
SIS_ARG_POSITION	3

Save and Exit

SIS_NOSAVE	0
SIS_SAVE	1
SIS_PROMPTSAVE	2

Save bitmap formats

SIS_SAVEBMP_BMP	0
SIS_SAVEBMP_JPG	1
SIS_SAVEBMP_DITHERBMP	2
SIS_SAVEBMP_PNG	3
SIS_SAVEBMP_TIFF	4
SIS_SAVEBMP_TIFF_PACKBITS	5
SIS_SAVEBMP_TIFF_GP4	6

Class filters

SIS_CLASSEXCLUDE	0
SIS_CLASSINCLUDE	1
SIS_OPENBRANCH	2

Units

SIS_LENGTHDIM	1
SIS_AREADIM	2
SIS_VOLUMEDIM	3

Boolean operations

SIS_BOOLEAN_AND	1
SIS_BOOLEAN_OR	2
SIS_BOOLEAN_XOR	3
SIS_BOOLEAN_DIFF	4
SIS_REVERSE	64
SIS_FORCE_POSITIVE	65

Geometry tests

SIS_GT_EQUAL	0
SIS_GT_DISJOINT	1
SIS_GT_INTERSECT	2
SIS_GT_TOUCH	3
SIS_GT_CROSS	4
SIS_GT_CROSSBY	5

SIS_GT_WITHIN	6
SIS_GT_CONTAIN	7
SIS_GT_OVERLAP	8

Property object types

SIS_OT_CURITEM	0
SIS_OT_DEFITEM	1
SIS_OT_SYSTEM	2
SIS_OT_WINDOW	4
SIS_OT_OVERLAY	5
SIS_OT_DATASET	6
SIS_OT_OPTION	7
SIS_OT_NOL	8
SIS_OT_PRINTER	9
SIS_OT_FTABLE	10
SIS_OT_SCHEMA	11
SIS_OT_SCHEMACOLUMN	12
SIS_OT_THEME	13
SIS_OT_THEMECOMPONENT	14

Geometry testing modes

SIS_GM_ORIGIN	0
SIS_GM_EXTENTS	1
SIS_GM_GEOMETRY	2

Cadcorp SIS Control licence levels

SIS_LEVEL_UNLICENSED	0
SIS_LEVEL_MANAGER	1
SIS_LEVEL_MODELLER	2
SIS_LEVEL_VIEWER	3

General constants

SIS_OVERRIDE	1
SIS_WRITEABLE	0
SIS_READONLY	-1

Rubber sheet methods

SIS_RUBBER_BEST_FIT	0
SIS_RUBBER_LINEAR_PATCH	1
SIS_RUBBER_INVERSE_SQUARE	2

3D manipulation modes

SIS_3DMODE_CRUISE	0
SIS_3DMODE_EYE	1
SIS_3DMODE_MODEL	2
SIS_3DMODE_PAN	3
SIS_3DMODE_ZOOM	4

Line segment shapes

SIS_LINE_STRAIGHT	0
SIS_LINE_BULGE	1
SIS_LINE_BEZIER	2

Feature filters

SIS_FEATUREEXCLUDE	0
SIS_FEATUREINCLUDE	1
SIS_FEATURECASCADE	2

Blob string formats

SIS_BLOB_SIS	0
SIS_BLOB_OGIS_WKB	1
SIS_BLOB_OGIS_WKT	2
SIS_BLOB_OGIS_GML	3

Creating area items

SIS_AREA_ONE_TO_ONE	0
SIS_AREA_MANY_TO_ONE	1
SIS_AREA_DISJOINT	2

Rubber-banding

SIS_RUBBERBAND_NONE	0
SIS_RUBBERBAND_LINE	1
SIS_RUBBERBAND_CIRCLE	2
SIS_RUBBERBAND_RECT	3
SIS_RUBBERBAND_ITEMS	4
SIS_RUBBERBAND_FIX_CIRCLE	5
SIS_RUBBERBAND_FIX_RECT	6

Scatter Grid creation modes

SIS_SCATTER_GRID_INTERPOLATE	0
SIS_SCATTER_GRID_CLOSEST	1
SIS_SCATTER_GRID_SUM	2
SIS_SCATTER_GRID_COUNT	3

Cleaning line items

SIS_CLEAN_LINE_NONE	0
SIS_CLEAN_LINE_REMOVE_0	1
SIS_CLEAN_LINE_REMOVE_180	2
SIS_CLEAN_LINE_REMOVE_SELF	4

Cleaning topological items

SIS_CLEAN_TOPO_NONE	0
SIS_CLEAN_TOPO_REMOVE_DANGLING	1
SIS_CLEAN_TOPO_FIX_UNDER_OVER	2
SIS_CLEAN_TOPO_REMOVE_SEEDS	4

Formula calculations

SIS_CALCULATE_COUNT	0
SIS_CALCULATE_SUM	1
SIS_CALCULATE_AVERAGE	2

Axes types

SIS_AXES_CARTESIAN	0
SIS_AXES_SPHERICAL	1

Type units

SIS_UNIT_ANGLE	0
SIS_UNIT_LINEAR	1
SIS_UNIT_AREA	2
SIS_UNIT_VOLUME	3

Angle units

SIS_UNITA_DEGREES	0
SIS_UNITA_RADIANS	1
SIS_UNITA_DMS	2
SIS_UNITA_GRADIANS	3

Linear units

SIS_UNIT1_M	0
SIS_UNIT1_MM	1
SIS_UNIT1_CM	2
SIS_UNIT1_KM	3
SIS_UNIT1_FEET	4
SIS_UNIT1_INCHES	5
SIS_UNIT1_IMPERIAL	6
SIS_UNIT1_YARD	7
SIS_UNIT1_FATHOM	8
SIS_UNIT1_MILE	9
SIS_UNIT1_NAUTMILE	10

Area units

SIS_UNIT2_M	0
SIS_UNIT2_MM	1
SIS_UNIT2_CM	2
SIS_UNIT2_KM	3
SIS_UNIT2_FEET	4
SIS_UNIT2_INCHES	5
SIS_UNIT2_YARDS	6
SIS_UNIT2_ACRE	7
SIS_UNIT2_HECTARE	8
SIS_UNIT2_TUBO	9
SIS_UNIT2_MILE	10
SIS_UNIT2_NAUTMILE	11

Volume units

SIS_UNIT3_M	0
SIS_UNIT3_MM	1
SIS_UNIT3_CM	2
SIS_UNIT3_LITRE	3
SIS_UNIT3_FEET	4
SIS_UNIT3_INCHES	5
SIS_UNIT3_YARDS	6
SIS_UNIT3_GALLON_IMP	7
SIS_UNIT3_GALLON_US	8

Theme scaling functions

SIS_FUNCTION_CONSTANT	0
SIS_FUNCTION_SQUAREROOT	1
SIS_FUNCTION_LINEAR	2
SIS_FUNCTION_LOG10	3

Printer colour capabilities

SIS_PRINTCAPS_QUERY	0
SIS_PRINTCAPS_MONO	1
SIS_PRINTCAPS_COLOUR	2

Theme annotation alignment

SIS_ALIGN_BOTTOM_LEFT	0
SIS_ALIGN_BOTTOM_CENTRE	1
SIS_ALIGN_BOTTOM_RIGHT	2
SIS_ALIGN_MIDDLE_LEFT	3
SIS_ALIGN_MIDDLE_CENTRE	4
SIS_ALIGN_MIDDLE_RIGHT	5
SIS_ALIGN_TOP_LEFT	6
SIS_ALIGN_TOP_CENTRE	7
SIS_ALIGN_TOP_RIGHT	8

NOL types

SIS_NOL_FILE	0
SIS_NOL_STANDARD	16
SIS_NOL_TEMPORARY	32
SIS_NOL_WORKSPACE	64

Graticule span types

SIS_GRATICULE_NONE	0
SIS_GRATICULE_GRID	1
SIS_GRATICULE_CROSSHAIR	2
SIS_GRATICULE_CROSSHAIR_TEXT	3

Deprecated constants

SIS_ERROR_NO_CURRENT_LAYER	5
SIS_LEVEL_MAPPER	1
SIS_LEVEL_EDITOR	2

Label Theme placement options

SIS_LABEL_START	0
SIS_LABEL_MIDDLE	1
SIS_LABEL_END	2
SIS_LABEL_ALONG	3

Index Dataset flags

SIS_INDEX_OUTLINES	1
SIS_INDEX_LABELS	2
SIS_INDEX_PYRAMID	4

Error codes

SIS_ERROR_OK	0
SIS_ERROR_SYNTAX	1
SIS_ERROR_ITEM_NOT_FOUND	2
SIS_ERROR_ITEM_GEOM_NOT_OWNED	3
SIS_ERROR_NO_ITEM_OPEN	4
SIS_ERROR_NO_CURRENT_DATASET	5
SIS_ERROR_NO_CURRENT_WINDOW	6
SIS_ERROR_NO_POSITION	7
SIS_ERROR_INVALID_HWND	8
SIS_ERROR_UNSOLICITED	9
SIS_ERROR_NO_GROUP_OPEN	10
SIS_ERROR_BAD_INDEX	11
SIS_ERROR_INVALID_ANGLE	12
SIS_ERROR_BAD_ITEMSCAN_STATUS	13
SIS_ERROR_NO_ITEMSCAN_RUNNING	14
SIS_ERROR_FILTER_NOT_FOUND	15
SIS_ERROR_CLASS_NOT_FOUND	16
SIS_ERROR_INVALID_PROPERTY	17
SIS_ERROR_USER_CANCEL	18
SIS_ERROR_FAILED	19
SIS_ERROR_INVALID_DATASET	20
SIS_ERROR_INVALID_OVERLAY	21
SIS_ERROR_UNDEFINED_OVERRIDE	22
SIS_ERROR_INVALID_NAME	23
SIS_ERROR_INVALID_STATUS	24
SIS_ERROR_FILE_ACCESS	25
SIS_ERROR_BAD_FILTER_TYPE	26
SIS_ERROR_INVALID_POSITION	27
SIS_ERROR_INVALID_EXTENT	28
SIS_ERROR_INVALID_PARAM	29

```
SIS_ERROR_INVALID_LIST                30
SIS_ERROR_COMMAND_FAILED              31
SIS_ERROR_CORRUPT_MESSAGE             32
SIS_ERROR_LOCUS_NOT_FOUND             33
SIS_ERROR_BUSY                        34
SIS_ERROR_LIBRARY                     35
SIS_ERROR_DATASET_MODIFIED            36
SIS_ERROR_BAD_DATASET_TYPE            37
SIS_ERROR_ITEM_READONLY               38
SIS_ERROR_GROUP_ALREADY_PLACED        39
SIS_ERROR_GROUP_EMPTY                 40
SIS_ERROR_NO_EDITABLE_ITEMS           41
SIS_ERROR_NO_SELECTION                42
SIS_ERROR_LIST_NOT_FOUND              43
SIS_ERROR_BAD_ITEM_TYPE               44
SIS_ERROR_EXTREME_PARAMETER           45
SIS_ERROR_NOL_NOT_FOUND               46
SIS_ERROR_EMPTY_RESULT                47
SIS_ERROR_ITEM_NOT_IN_SWD             48
SIS_ERROR_LIST_EMPTY                  49
SIS_ERROR_INSUFFICIENT_ITEM_ACCESS    50
SIS_ERROR_PARTIAL_SUCCESS             51
SIS_ERROR_OUT_OF_MEMORY               52
SIS_ERROR_INCONSISTENT_PARAMS         53
SIS_ERROR_REMOVE_DEFAULT_NOL          54
SIS_ERROR_NO_COMPOSED_WINDOW          55
SIS_ERROR_NOL_NOT_SUITABLE            56
SIS_ERROR_METHOD_NOT_IN_LEVEL         57
SIS_ERROR_COMMAND_NOT_IN_LEVEL        58
SIS_ERROR_DIFFERENT_DATASETS          59
SIS_ERROR_NOT_DATA_ENABLED            60
SIS_ERROR_TEMPLATE_NO_DATASETS        61
SIS_ERROR_TEMPLATE_NO_PHOTO           62
SIS_ERROR_TEMPLATE_PHOTO_NOT_EMPTY    63
SIS_ERROR_BAD_RESAMPLE_METHOD         64
SIS_ERROR_INVALID_FORMULA             65
SIS_ERROR_SCHEMA_NOT_FOUND            66
SIS_ERROR_THEME_NOT_FOUND             67
SIS_ERROR_SEED_NOT_FOUND              68
SIS_ERROR_NAMER_NOT_FOUND             69
SIS_ERROR_COMMAND_NOT_FOUND           70
SIS_ERROR_PROJECTION_NOT_FOUND        71
SIS_ERROR_DATUM_NOT_FOUND             72
SIS_ERROR_FTABLE_NOT_FOUND            73
SIS_ERROR_PTEMP_NOT_FOUND             74
SIS_ERROR_TABLE_NOT_FOUND             75
SIS_ERROR_NO_FTABLE_LOADED            76
SIS_ERROR_NOL_MODIFIED                77
SIS_ERROR_BRUSH_NOT_FOUND             78
SIS_ERROR_COLOURSET_NOT_FOUND         79
SIS_ERROR_PEN_NOT_FOUND               80
SIS_ERROR_NO_SCHEMA_LOADED            81
SIS_ERROR_VIEW_NOT_FOUND              82
SIS_ERROR_NO_THEME_LOADED             83
SIS_ERROR_NO_LONGER_SUPPORTED         84
SIS_ERROR_NOL_OBJECT_NOT_FOUND        85
```

■ Old global constants

These are obsolete. New applications should use the constants with the `SIS_` prefix.

```
GIS_TOP_LEFT                    0
GIS_TOP_RIGHT                   2
GIS_TOP_CENTRE                  6
GIS_BOTTOM_LEFT                 8
GIS_BOTTOM_RIGHT                10
GIS_BOTTOM_CENTRE               14
GIS_BASE_LEFT                   24
GIS_BASE_RIGHT                  26
GIS_BASE_CENTRE                 30
GIS_MIDDLE_LEFT                 72
GIS_MIDDLE_RIGHT                74
GIS_MIDDLE_CENTRE               78
GIS_COM_ALL                     0
GIS_COM_ADD                     1
GIS_COM_REMOVE                  2
GIS_COM_NONE                    3
GIS_INVISIBLE                   0
GIS_VISIBLE                     1
GIS_HITTABLE                    2
GIS_EDITABLE                    3
GIS_CURRENTWINDOW               0
GIS_CURRENTSWD                  1
GIS_ALLWINDOWS                  2
GIS_FILTERRESET                 0
GIS_FILTERADD                   1
GIS_FILTERREMOVE                2
GIS_ARG_ESCAPE                  0
GIS_ARG_ENTER                   1
GIS_ARG_BACKSPACE               2
GIS_ARG_POSITION                3
GIS_NOSAVE                      0
GIS_SAVE                        1
GIS_PROMPTSAVE                  2
GIS_CLASSEXCLUDE                0
GIS_CLASSINCLUDE                1
GIS_OPENBRANCH                  2
GIS_LENGTHDIM                   1
GIS_AREADIM                     2
GIS_VOLUMEDIM                   3
GIS_SHOWUNITS                   -1
GIS_NOSHOWUNITS                 0
GIS_DISOWNDATASET               0
GIS_OWNDATASET                  -1
GIS_BOOLEAN_AND                 1
GIS_BOOLEAN_OR                  2
GIS_OVERRIDE                    1
GIS_WRITEABLE                   0
GIS_READONLY                    -1
```

Appendix 6

Index dataset naming conventions

- Introduction 419
- Bartholomew 50km 419
- Count Rectangles 420
- Digital 420
- LandLine 10 000 420
- LandLine 1:1250 421
- LandLine 1:2500 421
- National Grid 250 metre 421
- Ordnance Survey NTF 421
- OS Asset Manager 5km (1:2500) 421
- OS Basedata.GB 422
- OS Boundary-Line 422
- OS Network Manager 422
- OS Panorama 422
- OS Profile (Contour) 422
- OS Profile (DTM) 423
- OS Route Manager 423
- OS Strategi 423
- OSCAR 423
- OSNI 423
- Pyramid 424
- USGS naming conventions 425

Introduction

An index dataset uses index dataset naming conventions to tile together the map base. This appendix lists all index dataset naming conventions available in Cadcorp SIS.

Bartholomew 50km

(1:250 000)

Namer	`ABartholomewNamer`
Parameters	none
Typical Name	`XXX_YYYY.ext`
Properties	`_projection$` `_scale#`

■ Count Rectangles

Namer `ACountNamer(bCartesian%, width#,height#,projection$)`

Parameters *bCartesian%*

`True` the tiles are in a cartesian projection

`False` the tiles are in a spherical projection

width#, height#
the width and height of each tile, in metres or degrees for Cartesian and spherical tiles respectively

projection$
a named projection stored in a named object library (without quote characters)

Typical Name `0xXXXXYYYY.ext`

Properties `_projection$`
`_scale#`

■ Digital

Namer `ADigitNamer(digit%,step%,width#,bCount%,projection$)`

Parameters *digit%*
the total number of digits in each tile name, in the range 2 to 32 inclusive

step%
the number of tiles across a grid cell, in the range 1 to 9 inclusive

width#
the tile grid cell width

bCount%

`True` the tilenames are sequential within a cell

`False` the tilenames are not sequential within a cell

projection$
a named projection stored in a named object library (without quote characters)

Typical Name `XY.ext`, `XXYY.ext`, `XXXYYY.ext`

Properties `_projection$`
`_scale#`

■ LandLine 10 000

5km (1:10 000)

Namer `ANtfNamer` or `ALandLine10000Namer`

Parameters none

Properties `_projection$`
`_scale#`

Typical Name `SS78NE.ext`

■ LandLine 1:1250

500m (1:1250)

Namer `ANtfNamer` or `ALandLine1250Namer`

Parameters none

Properties `_projection$`
`_scale#`

Typical Name `SS7887SE.ext`

■ LandLine 1:2500

1km (1:2500)

Namer `ANtfNamer` or `ALandLine2500Namer`

Parameters none

Properties `_projection$`
`_scale#`

Typical Name `SS7887.ext`

■ National Grid 250 metre

250m (1:500)

Namer `ANatGrid250Namer`

Parameters none

Properties `_projection$`
`_scale#`

Typical Name `SS788700.ext`

■ Ordnance Survey NTF

Namer `ANtfNamer`

Parameters none

Properties `_projection$`
`_scale`

■ OS Asset Manager 5km (1:2500)

Namer `ANtfNamer` or `AOscarAssetNamer`

Parameters none

Properties `_projection$`
`_scale#`

Typical Name `SS78NE.ext`

■ OS Basedata.GB

100km (1:625 000)

Namer	`ANtfNamer` or `AOsRoutePlannerNamer`
Parameters	none
Properties	`_projection$` `_scale#`
Typical Name	`SS.ext`

■ OS Boundary-Line

25km (1:10 000)

Namer	`ANtfNamer` or `AOsBoundaryNamer`
Parameters	none
Properties	`_projection$` `_scale`
Typical Name	`2144.ext`

■ OS Network Manager

10km (1:50 000)

Namer	`ANtfNamer` or `AOscarNetworkNamer`
Parameters	none
Properties	`_projection$` `_scale`
Typical Name	`SS78.ext`

■ OS Panorama

20km (1:50 000)

Namer	`ANtfNamer` or `AOsLandrangerNamer`
Parameters	none
Properties	`_projection$` `_scale`
Typical Name	`SS78.ext`

■ OS Profile (Contour)

5km (1:10 000)

Namer	`ANtfNamer` or `AOsProfileContourNamer`
Parameters	none
Properties	`_projection$`

	`_scale`
Typical Name	`SS78NE_C.ext`

■ OS Profile (DTM)

5km (1:10 000)

Namer	`ANtfNamer` or `AOsProfileDtmNamer`
Parameters	none
Properties	`_projection$` `_scale`
Typical Name	`SS78NE_D.ext`

■ OS Route Manager

50km (1:25 000)

Namer	`ANtfNamer` or `AOscarRouteNamer`
Parameters	none
Properties	`_projection$` `_scale`
Typical Name	`SS78.ext`

■ OS Strategi

50Km (1:250 000)

Namer	`ANtfNamer` or `AOsStrategiNamer`
Parameters	none
Properties	`_projection$` `_scale`
Typical Name	`SS.ext`

■ OSCAR

(Pre April 94) 5Km (1:10000)

Namer	`ANtfNamer` or `AOscarNamer`
Parameters	none
Properties	`_projection$` `_scale`
Typical Name	`SS78NE.ext`

■ OSNI

600m (1:1250) 1.2km (1:2500) 2.4km (1:5000) 9.6km (1:10 000)

Namer `AOsniNamer`

Parameters none

Properties `_projection$`
`_scale`

Typical Name

1:1250	`00609SE1.ext`
1:2500	`00609SE.ext`
1:50 000	`00609.ext`
1:10 000	`006.ext`

■ Pyramid

Namer `APyramidNamer(bCartesian%, width#, height#, projection$, nDigit%)`

Parameters *`bCartesian%`*

`True` the tiles are in a Cartesian projection

`False` the tiles are in a spherical projection

`width#`, *`height#`*
the width and height of each tile, in metres or degrees for Cartesian and spherical tiles respectively

`projection$`
a named Projection stored in a named object library (without quote characters)

`nDigit%`
the maximum number of digits in a file name, in the range 1 to 24 inclusive. This value also controls the depth of the pyramid.

Properties `_bCartesian&`
Are the tiles in a Cartesian projection?

`_nDigit&`
the number of digits in the tile name

`_projection$`
the tile naming convention projection

`_scale#`
the tile scale

`_degree_height#`
the tile height, in degrees

`_metre_height#`
the tile height, in metres

`_span#`
the tile span, in metres

`_degree_width#`
the tile width, in degrees

`_metre_width#`
the tile width, in metres

Typical Name `0qNNNNNN.ext`

■ USGS naming conventions

◆ USNGNamer_1_000_000

square tiles, at 1:1 000 000

Parameters *`namer$`*
argument to `AGlsCreateIndexOverlay` and `AGlsCreateIndexCoverage`

◆ USNGNamer_1_000_000 (datum$,zone%)

Parameters *`datum$`*
the North American Datum to use, with values `nad83` or `nad27`
`zone%`
the UTM value, which must be a number between 1 and 60 inclusive

◆ USNGNamer_100_000

square tiles, at 1:100 000

Parameters *`namer$`*
argument to `AGlsCreateIndexOverlay` and `AGlsCreateIndexCoverage`

◆ USNGNamer_100_000 (datum$,zone%)

Parameters *`datum$`*
the North American Datum to use, with values `nad83` or `nad27`
`zone%`
the UTM value, which must be a number between 1 and 60 inclusive

◆ USNGNamer_10_000

square tiles, at 1:10 000

Parameters *`namer$`*
argument to `AGlsCreateIndexOverlay` and `AGlsCreateIndexCoverage`

◆ USNG Namer_10_000 (datum$,zone%)

Parameters *`datum$`*
the North American Datum to use, with values `nad83` or `nad27`
`zone%`
the UTM value which must be a number between 1 and 60 inclusive

◆ USNG Namer_1_000

square tiles, at 1:1 000

Parameters `namer$`
argument to `AGlsCreateIndexOverlay` and `AGlsCreateIndexCoverage`

◆ USNG Namer_1_000 (datum$, zone%)

Parameters `datum$`

the North American Datum to use, with values `nad83` or `nad27`

`zone%`

the UTM value which must be a number between 1 and 60 inclusive

◆ GTOPO30 Namer

a global digital elevation model (DEM) with a horizontal grid spacing of 30 arc seconds (approximately 1km), specified by the United States Geological Service

◆ GTOPO30 Antartica Namer

a global digital elevation model (DEM) with a horizontal grid spacing of 45 arc seconds (approximately 1.5km), specified by the United States Geological Service

Appendix 7

Setting up connections using data from OpenGIS Servers

■ OpenGIS Web Map Server 427
■ OpenGIS Web Feature Server. 427

■ OpenGIS Web Map Server

Cadcorp SIS ASC and Cadcorp SIS ISAPI can act as an OpenGIS Web Map server. This uses the OpenGIS specification (WMS) to serve raster grids over the internet. A Cadcorp SIS desktop product (the client) can read the information as an overlay. Any OpenGIS WMS conformant service (Web Address) can be used as the dataset source. This acts as an OpenGIS Web Map Server (WMS) client, downloading raster images over the internet from an OpenGIS WMS.

The plug-in lets you configure the connection to the WMS, and to preview images before continuing.

Use the following arguments for the `CreateDataSourceOverlay` method:

Arguments

`pos` SHORT INTEGER
the position in the overlays list at which to insert the overlay

`clsDataSource` STRING
use `AOgcWmsDataset` for the classname of the data source

`params` STRING
the parameters:
[`URL=`*`url`*`,`] [`dynamic=true|false`].

`URL=`*`url`*	this is the URL of the OpenGIS Web Map Server. The default value is `""`.
`dynamic=true|false`	Should the WMS client be dynamic, getting an image from the selected OpenGIS WMS for each redraw? The default value is true.

■ OpenGIS Web Feature Server

Cadcorp SIS ASC and Cadcorp SIS ISAPI can act as an OpenGIS Web Feature server. This uses the OpenGIS specification (WFS) to serve vector geometry and attributes over the internet. A Cadcorp SIS desktop product (the client) can read the information as an overlay. Any OpenGIS WFS conformant service (Web Address) can be used as the dataset source. This acts as an OpenGIS Web Feature Server (WFS) client, down-

loading vector data in OpenGIS GML format over the internet from an OpenGIS WFS.

The plug-in lets you configure the connection to the WFS.

Use the following arguments for the `CreateDataSourceOverlay` method:

Arguments

`pos` SHORT INTEGER

the position in the overlays list at which to insert the overlay

`clsDataSource` STRING

use `WFSDataSource` for the classname of the data source

`params` STRING

the parameters:

`[GetFeaturesURI=`*`uri`*`, ] [DescribeFeatureTypeURI=`*`uri`*`, ] [PostData=`*`httpdata`*`, ] [ OverlayName=`*`name`*`, ][RequestMethod=post|get, ] [Validate=true|false]`

`GetFeaturesURI=`*`uri`*	the URL of the OpenGIS Web Feature Server `GetFeatures` request. The default value is `""`.
`DescribeFeatureTypeURI=`*`uri`*	the URL of the OpenGIS Web Feature Server `DescribeFeatureType` request. The default value is `""`.
`PostData=`*`httpdata`*	if the `RequestMethod` is `"post"`, this is the HTTP POST data for the request
`OverlayName=`*`name`*	the overlay name to be used in Cadcorp SIS
`RequestMethod=post|get`	specifies which HTTP request method to use, either `"post"` or `"get"`
`Validate=true|false`	specifies whether or not to validate the downloaded GML

ASCII character set

0	1	2	3	4	5	6	7	8 backspace	9 tab
10 linefeed	11	12	13 carriage return	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32 space	33 !	34 "	35 #	36 $	37 %	38 &	39 '
40 (	41)	42 *	43 +	44 ,	45 -	46 .	47 /	48 0	49 1
50 2	51 3	52 4	53 5	54 6	55 7	56 8	57 9	58 :	59 ;
60 <	61 =	62 >	63 ?	64 @	65 A	66 B	67 C	68 D	69 E
70 F	71 G	72 H	73 I	74 J	75 K	76 L	77 M	78 N	79 O
80 P	81 Q	82 R	83 S	84 T	85 U	86 V	87 W	88 X	89 Y
90 Z	91 [	92 \	93]	94 ^	95 _	96 `	97 a	98 b	99 c
100 d	101 e	102 f	103 g	104 h	105 i	106 j	107 k	108 l	109 m
110 n	111 o	112 p	113 q	114 r	115 s	116 t	117 u	118 v	119 w
120 x	121 y	122 z	123 {	124 |	125 }	126 ~	127	128 €	129
130 ‚	131 ƒ	132 „	133 …	134 †	135 ‡	136 ˆ	137 ‰	138 Š	139 ‹
140 Œ	141	142 Ž	143	144	145 ‘	146 ’	147 “	148 ”	149 •
150 –	151 —	152 ˜	153 ™	154 š	155 ›	156 œ	157	158 ž	159 Ÿ
160 space	161 ¡	162 ¢	163 £	164 €	165 ¥	166 |	167 §	168 ¨	169 ©
170 ª	171 «	172 ¬	173 –	174 ®	175 ¯	176 °	177 ±	178 ²	179 ³
180 ´	181 µ	182 ¶	183 ·	184 ¸	185 ¹	186 º	187 »	188 ¼	189 ½
190 ¾	191 ¿	192 À	193 Á	194 Â	195 Ã	196 Ä	197 Å	198 Æ	199 Ç
200 È	201 É	202 Ê	203 Ë	204 Ì	205 Í	206 Î	207 Ï	208 Ð	209 Ñ
210 Ò	211 Ó	212 Ô	213 Õ	214 Ö	215 ×	216 Ø	217 Ù	218 Ú	219 Û
220 Ü	221 Ý	222 Þ	223 ß	224 à	225 á	226 â	227 ã	228 ä	229 å
230 æ	231 ą	232 è	233 é	234 ê	235 ë	236 ì	237 í	238 î	239 ï
240 ð	241 ñ	242 ò	243 ó	244 ô	245 õ	246 ö	247 ÷	248 ø	249 ú
250 ú	251 û	252 ü	253 ý	254 þ	255 ÿ				

Index

A

ACOM commands 333
Activate **76**
ActivateWnd **76**
AddCommand 24
AddCommand 5, 24
AddCommand (Cadcorp SIS Control) **77**
AddCommand (GisLink) **77**
AddToList **78**
AllowCommands (Cadcorp SIS Control) **79**
AllowCommands (GisLink) **79**
AppCommand
  event 25
ASCII character set 429

B

BezierTo **80**
BorderBevel (Control property) 22
BulgeTo **80**

C

Cadcorp SIS Control
  adding to a Visual Basic project 17
  adding to a Visual C++ Project Workspace 18
  commands 24
  events 25
  licensing 34
  methods 15
  properties 21
callback commands 9
CallCommand 9, **80**
CanDoCommand **81**
ChangeFeatureFilter **81**
ChangeLocusTestMode **81**
ChangePrjUnits **82**
ChangeValueListFilter **82**
CheckNetworkDongle 33
CheckNetworkDongle (Control property) 22
CleanLines **83**
CloseDataset **83**
CloseIndexDatasetTile **84**
CloseItem **84**
CombineFilter **84**
CombineLists **85**
CombineLocus **85**
CommandAction
  event 26
CommandMessage (Control property) 22
CommandPrompt (Control property) 23
CompactDataset **86**
Compose **86**
constants
  global 2, 411
  superseded 417
Copy **86**
CopyFeatureCode **86**
CopyListItems **87**
CopyThemeComponent **87**
create a bar theme 289
create a graduated theme 292
create a label theme 291
create a pie theme 290
create a range theme 291
create an extruded theme 290
create and apply a schema 287
CreateAreaFromLines **87**
CreateAssembly **88**
CreateBackdropOverlay **88**
CreateBarTheme **89**
CreateBds **89**
CreateBitmap **89**
CreateBlock **90**
CreateBoolean **90**
CreateBoundary **90**
CreateBoxLabel **91**
CreateBoxText **91**
CreateBufferFromItems **92**
CreateBufferLocusFromItems **92**
CreateCircle **92**
CreateCircleLocus **93**
CreateClassTreeFilter **93**

CreateCombinedFilter **93**
CreateContourTheme **94**
CreateConvexHull **94**
CreateDataSourceOverlay **94**
CreateDbBlobOverlay **95**
CreateDbOverlay **96**
CreateDbPointOverlay **97**
CreateDbTable 20**, 98**
CreateDisplacement **98**
CreateDotTheme **99**
CreateDoubleBoolean **99**
CreateDrapeBitmap **100**
CreateEllipse **100**
CreateExtrudeTheme **100**
CreateExtrusion **100**
CreateFeatureFilter **101**
CreateFlowTheme **101**
CreateFormulaGrid **101**
CreateGraduatedTheme **101**
CreateGraticule **102**
CreateGridFromQZone **102**
CreateGroup **102**
CreateGroupFromItems **102**
CreateIndexCoverage **103**
CreateIndexOverlay **103**
CreateIndividualTheme **104**
CreateInsert **104**
CreateInternalOverlay **104**
CreateIsoRoute **105**
CreateItem **106**
CreateItemB **106**
CreateItemFromLocus **107**
CreateKeyMap **107**
CreateLabelTheme **107**
CreateLineText **108**
CreateLinkFilter **108**
CreateListFromOverlay **108**
CreateListFromSelection **108**
CreateLocusFromItem **109**
CreateNorthPoint **109**
CreateOpenGisSqlOverlay **109**
CreatePhaseOverlay **110**
CreatePhoto **110**
CreatePieTheme **110**
CreatePoint **111**
CreatePropertyFilter **111**
CreateQZoneFromGrid **111**
CreateRangeTheme **111**
CreateRectangle **112**
CreateRectLocus **112**
CreateReliefTheme **112**
CreateRubberSheet **112**
CreateScaleBar **113**
CreateScatterGrid **113**
CreateSurface **114**
CreateText **114**
CreateThiessen **114**
CreateTin **114**
CreateTopoTheme **115**
CreateValueListFilter **115**
custom commands
  adding 7

D

data types 76
dataset
  properties 350
DatasetItemEdit
  event 26
datasets 241—252
default
  properties 366
DefineNolDatum **115**
DefineNolItem **116**
DefineNolItemFromLocus **116**
DefineNolObject **116**
DefineNolPrintTemplate **118**
DefineNolPrjLatLon **118**
DefineNolPrjTm **119**
DefineNolShape **119**
DefineNolView **120**
DefineRecordset **120**
Delete **121**
DeleteItem **121**
DeleteNolObject **121**
DeleteRecordset **122**
DescribeProperty **122**
DeselectAll **122**
Digital Namers 419
DigitiserSnap **122**
Display (Control property) 23
DoCommand 25
DoCommand 25**, 122**
DoEvents 30
double 76
DrapeBitmap **123**
DrawList **123**

E

edit a theme at a known position 293
edit an existing theme 293
EmptyGroup **124**
EmptyList **124**
EnsureOpenWithin **124**
EvaluateFlt **124**
EvaluateInt **125**
EvaluateStr **125**
events 25
example code 223—294
Exit **125**
ExplodeOverlayTheme **126**
Export **126**
ExportBds **128**
ExportBmp **128**
ExportECW **129**
ExportFeatureTable **129**
ExportJpeg **129**
ExportPdf **129**
ExportPng **131**
ExportVrml **131**
ExportWmf **132**

F

FacetGeometry **132**
feature
  properties 367
find out the number of themes on an overlay 294
FindDatasetOverlay **132**
FindExternalDataset **133**
Form_Load 6

G

Generate Programming File command 5
Get3DEye **133**
Get3Dlook **133**
GetArg 29
GetAxesAngle **133**
GetAxesFromLatLonHgt **133**
GetAxesPrj **134**
GetAxesType **134**
GetBlob **134**
GetBlobB **135**
GetBlobExtent **136**
GetCommandTick **136**
GetCoordExtent **137**
GetCoordString **137**
GetDataset **138**
GetDatasetContainer **138**
GetDatasetExtent **138**
GetDatasetPrj **138**
GetDisplayExtent **139**
GetErrorString 21**, 139**
GetExtent **139**
GetFeatureFilterBranches **140**
GetFeatureTableBranches **140**
GetFlt **141**
GetGeomAngleFromLength **141**
GetGeomDim **141**
GetGeomLength **142**
GetGeomLengthUpto **142**
GetGeomNumPt **143**
GetGeomNumSeg **143**
GetGeomPosFromLength **143**
GetGeomPt **144**
GetGeomSegAxis **144**
GetGeomSegBulge **144**
GetGeomSegShape **145**
GetGeomSelfIntersection **145**
GetGeomTgtFromLength **146**
GetGridItemValue **146**
GetHook **146**
GetImplicitNolObject **147**
GetInt **147**
GetLatLonHgtFromAxes **147**
GetListExtent **148**
GetListItemFlt **148**
GetListItemInt **148**
GetListItemStr **149**
GetListSize **149**
GetNumAscClients **150**
GetNumGeom **150**
GetNumNol **150**
GetNumSel **150**
GetNumSwd **150**
GetNumWnd **151**
GetOverlayFilter **151**
GetOverlayLocus **151**
GetOverlaySchema **151**
GetOverlayTheme **152**
GetOverlayThemeLegend **152**
GetPhotoWorldPos **153**
GetPos 13**, 153**
GetPosEx 13**,** 29**, 153**
GetPropertyDescription **154**
GetSpatialReference **154**
GetSpatialReferenceFromExtent **155**
GetStr **155**
GetStrW **156**
GetViewExtent **156**
GetViewPos **156**
GetViewPosEx **157**
GetViewPrj **157**
GIS_
  constants 418
GisLink commands 7
GisLink customisation 5

H

hot snapping 379

I

ImportDataset **157**
ImportFeatureTable **158**
InsertDataset **158**
InsertFeatureCode **158**
InsertGeomPt **158**
InsertOverlayTheme **159**
InsertSchemaColumn **159**
IsAscLicensed **159**
IsoRoute **160**
item
  properties 345

J

JoinLines **161**

L

Level (Control property) 23
LicenceError
  event 26
licensing 34
LineTo **161**
ListCaps button 6
LoadFeatureTable **161**
LoadSchema **161**
LoadSwd **162**
LoadTheme **162**
LocusIntersect **162**
long integer 76

M

measure a route 225
MeasureAzimuth **162**
MeasureGreatCircle **163**
MeasureRoute **163**
Message **164**
methods 15
MetreFromStr **164**
Microsoft Visual Basic 16
Microsoft Visual C++ 17
MouseTrack
  event 27
move items around the map base 226
MoveAxes **165**
MoveList **165**
MoveTo **165**
MultiRoute **165**

N

named
  object libraries 270
  objects 270
  properties 368
  shapes 275
  views 274
NolCatalog **166**
NolClose **167**
NolCompact **168**
NolCreate **168**
NolInsert **168**
NolOwn **168**
NolSave **169**

O

one-shot commands 9
OpenClosestItem **169**
OpenDatasetItem **169**
OpenExistingDatasetItem **170**
OpenFormulaItem **170**
OpenGIS
  Web Map Servers 65
OpenItem **170**
OpenList **170**
OpenOverlayItem **171**
OpenSel **171**
overlay
  properties 373
overlays 241—252
OwnDataset **171**

P

pan across the map base 228
pan the map base by a given amount 229
Paste **172**
PasteFrom **172**
PhotoGrid **172**
PlaceGroup **172**
PlacePrintTemplate **173**
position input 28—33
printer
  properties 375
Program window 14
Prompt **173**
PromptChange
  event 27
properties 345

R

RecallNolItem **173**
RecallNolView **173**
Redraw 25, **173**
RedrawExtent **174**
RefreshDataset **174**
RefreshDbTable **174**
RegisterGroupType **175**
RegisterTrigger **175**
Release **175**
ReleaseNA **176**
remove a theme 294
RemoveAtt **176**
RemoveCommand **176**
RemoveFeatureCode **176**
RemoveOverlay **177**
RemoveOverlayTheme **177**
RemoveProperty **177**
RemoveSchemaColumn **177**
Render **178**
ReorderOverlay **178**
RubberSheetRaster **178**

S

SaveBitmap **179**
SaveDataset **179**
SaveSwd **179**
ScaleChange
  event 27
Scan **180**
ScanDataset **180**
ScanGeometry **181**
ScanList **181**
scanning 275
ScanOverlay **182**
ScanPointContainers **182**
schema
  properties 376, 377
schemas
  example code 287
ScrollView **183**
searches 275
SelectAll **183**
SelectionChange
  event 27
SelectItem **183**
SelectList **183**
SendPrint **184**
Set3DView **184**
SetAxesAngle **185**
SetAxesGrid **185**
SetAxesNormal **185**

SetAxesPrj **185**
SetCombinedFilterClause **186**
SetCommandBitmap **186**
SetCoordUnits **187**
SetDatasetPrj **188**
SetDefaultPrj **188**
SetFlt **188**
SetForegroundWindow **189**
SetGazetteerView **189**
SetGeomPt **189**
SetGeomSegBulge **190**
SetGridItemValue **190**
SetInt 33**, 190**
SetListFlt **191**
SetListInt **191**
SetListStr **191**
SetOverlayFilter **192**
SetOverlayLocus **192**
SetOverlaySchema **192**
SetPhotoWorldCentre **193**
SetRubberTransform **193**
SetStr **194**
SetStrW **194**
setting up the environment 224
SetUnits **195**
SetupLink 6
SetupLink **195**
SetViewExtent **196**
SetViewPrj **196**
short integer 76
ShowMenu **196**
ShowWaitCursor (Control property) 23
SimplifyGeom **196**
SIS_
  constants 411
SisConst.bas 16
SisConst.h 18
Snap
  event 27
Snap2D **197, 314**
SnipGeometry **198**
spatial searches 275
SplitExtent **198**
SplitPos **198**
sprites 226
startup form 6
StoreAsArea **199**
StoreAsLine **199**
StoreFeatureTable **199**
StoreSchema **199**
StoreTheme **200**
StrFromMetre **200**
string 76
SWD (Control property) 23
SwdClose **200**
SwdNew **201**
SwdNewFromSwt **201**
SwdNewWindow **201**
SwdNewWindow3D **201**
SwdNewWindowTable **201**
SwdOpen **202**
SwdSave **202**
SwdSaveAs **202**
SwdSaveAsSwt **202**
SwitchCommand 9**, 202**
SwtClose **203**
SwtNew **203**
SwtNewFromSwt **203**
SwtOpen **203**
SwtSave **203**
SwtSaveAs **204**
SwtSaveAsSwt **204**
system
  properties 370**,** 377
system commands
  running 9

T

TableNewWindow **204**
Takeover **204**
theme
  properties 383**,** 405
themes
  example code 289
TickCommand **205**
TopoClean **205**
TopoCombineNamedSeeds **206**
TopoConvertToArea **206**
TopoConvertToChain **206**
TopoConvertToLine **206**
TopoConvertToPolygon **207**
TopoCreateArea **207**
TopoCreateBoolean **207**
TopoCreateChain **207**
TopoCreateEmptyNamedSeed **208**
TopoCreateLine **208**
TopoCreateLink **208**
TopoCreateNamedSeed **208**
TopoCreateNode **209**
TopoCreatePolygon **209**
TopoDeleteLink **209**
TopoDeleteNamedSeed **209**
TopoDeleteNode **209**
TopoDeleteSeed **210**
TopoEdgeFill **210**
TopoFindRoute **210**
TopoFloodFill **211**
TopoGetLinkNode **212**
TopoGetLinkNumSeed **212**
TopoGetLinkSeed **212**
TopoGetNamedSeedDataset **212**
TopoGetNamedSeedLoopLink **213**
TopoGetNamedSeedLoopSize **213**
TopoGetNamedSeedNumLoop **213**
TopoGetNodeLink **214**
TopoGetNodeNumLink **214**
TopoGrowNamedSeed **214**
TopoIsChain **214**
TopoIsPolygon **215**
TopoMoveNode **215**
TopoReverseSeed **215**
TopoShrinkNamedSeed **215**
TraceGeom **216**
TrackMouse (Control property) 24
twips 376

U

`UpdateItem` **216**
`UpdateWorkspaceWindow` **217**
User position input 13
using triggers 223

V

view a schema in a table window 288
Visual Basic 16
  Startup form 6
Visual C++ 17

W

window
  properties 405
windows 239—241
`WndArrangeIcons` **217**
`WndCascade` **217**
`WndTile` **217**
`WndTileHorizontal` **217**
`WorkspaceClose` **217**
`WorkspaceNew` **218**
`WorkspaceOpen` **218**
`WorkspaceSave` **219**

Z

zoom the map base 229
`ZoomExtent` **219**
`ZoomView` **219**