REVMEDIA - “The Developer’s Link” Technical Briefings #1

OpenInsight for Workgroups Window Structures

SPREZZATURA LTD

Printed in the United Kingdom.

Author	Andrew P McAuley
Publisher	Andrew P McAuley
Technical Editor	Cameron Purdy

Contents

Chapter 1: Introduction

Welcome to the REVMEDIA Technical Briefing “OpenInsight Window Structures”.

REVMEDIA Technical Briefings are designed to continue where the documentation provided with OpenInsight leaves off. They are not intended to replace existing documentation, rather to supplement them. As such, they will only infrequently reproduce information published elsewhere. Where such a case exists the sources will be credited.

This document is designed to provide a complete reference work to the structure of an uncompiled OpenInsight Window, that is, a window as it is stored in SYSREPOSWINS. A later reference will deal with the structure of compiled OpenInsight Windows, that is, those stored in SYSRESPOSWINEXES.

It is assumed that the reader is already a competent OpenInsight programmer; if this is not the case, very little of this document will make sense.

The author would like to take this opportunity to thank those who made this work possible. These include (but are not limited to) Cameron Purdy (who, whilst Technical Editor should not be held responsible for errors and omissions - any such are the fault of the author!) , Gene Gleyzer and Aaron Kaplan of Revelation Technologies Inc. for technical assistance, Cameron Christie of Sprezzatura for proof-reading and Reed Technology and Carl Pates of AMSyS for encouraging this research.

Finally my thanks to my wife whose ~~nagging~~ <g> encouragement forced me to at last get my nose back on the grindstone and restart REVMEDIA!

Chapter 2: OpenInsight Window Structure - An Overview

This chapter provides an overview of the issues which will be addressed within the rest of the document.

It deals with the following issues :-

- Location of Windows
- Makeup of window

Location of Windows

Uncompiled OpenInsight Windows are stored in the SYSREPOSWINS table on the REVBOOT directory. Normally they are placed there by a REPOSITORY NEW or REPOSITORY WRITE method. In this way the system ensures that it knows the rights associated with an entity.

If a Window does not have an associated SYSREPOS entity, it will not be accessible from an OpenInsight application. For this reason it is not recommended that you write new window structures directly to the SYSREPOSWINS table. Rather, if the developer creates a new window structure and wants to update the system with it, they should do it via a repository call as follows

```
Declare Function Repository
RetVal = Repository(A,B,C,D,E,F,G,H,I,J,K,L)
```

where

A The method to execute, in this case “NEW”

B The name of the Window to update. Specify the full entity ID, which is the key to the SYSREPOS table. The fully qualified entity identifier is required. This takes the form AppName*OIWIN**WinName.

C The State to set the entity to. This is an incompletely specified repository flag which was intended to show the stage of the development life-cycle that had been reached by the entity. It has not yet been fully implemented. May be left null.

D The Publishable flag. Whether the entity should be deployed when the RDK is used. If you want to deploy the uncompiled version of the window, set this to 1. To supress the deployment of the uncompiled window, set it to 0.

E The Shareable flag. Whether the entity should be made available to any applications which inherit from the current application. It would be usual to set this to true although the circumstances of your application will dictate this.

F A field mark delimited list of the entities that this entity is used by. These should have the structure of valid SYSREPOS keys. Note that the REPOSITORY function will not validate these to actually exist. In fact if the rows do not exist in the SYSREPOS table it will create them complete with links to the entity being created. May be left null.

G A field mark delimited list of the entities that this entity uses. These should have the structure of valid SYSREPOS keys. Note that the REPOSITORY function will not validate these to actually exist. In fact if the rows do not exist in the SYSREPOS table it will create them complete with links to the entity being created. May be left null.

H A field mark delimited list of document entities associated with this entity. These should have the structure of valid SYSREPOS keys. Note that the REPOSITORY function will not validate these to actually exist. In fact if the rows do not exist in the SYSREPOS table it will create them complete with links to the entity being created. May be left null.

I A value mark delimited list of users with access permissions for this entity. Note that the REPOSITORY function will not validate these to actually exist, nor will it create them for you. May be left null.

J A value mark delimited list of users with update permissions for this entity. Note that the REPOSITORY function will not validate these to actually exist, nor will it create them for you. May be left null.

K The SYSREPOS title to give this entity.

L The window structure to write to the SYSREPOSWINS table.

Note that subsequent updates to the window structure can be made directly by writing to the SYSREPOSWINS table. This will update the window itself but will not record the fact that updates have been performed. Alternatively the REPOSITORY WRITE method can be used. This takes the same parameters as the REPOSITORY NEW but parameter A is set to “WRITE not “NEW”. This method can also be used to update any of the details given in the SYSREPOS table. Do not pass a null Window definition however or you will overwrite the existing definition with null!

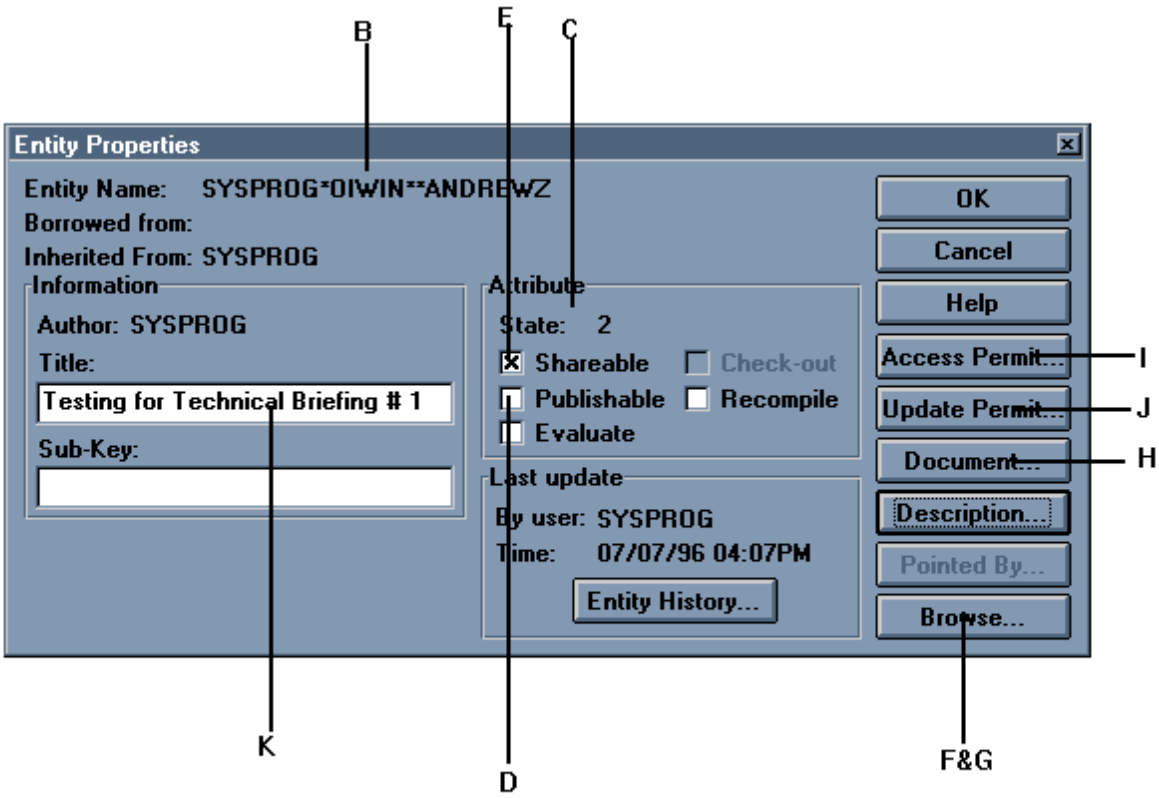

Figure 1 - Entity Properties Dialog With Repository Parameters Shown

Makeup of Window

The window structure is stored as a normal linear hash row. It is made up of four Record Mark delimited sections. Each section is further delimited depending upon the category of section being defined.

Within the Windows environment, every control on screen is a separate Windows object and thus must be described separately to the Windows API when the window is first created. Each of the sections in the window structure provides information which will ultimately be used by Presentation Server to instruct the WinAPI in the construction of the entry form.

The sections are

- Sizing information - this section contains information about the window as a whole including how many controls there are on the window.
- Window information - this section contains information about the parent window control itself, including size, location, title etc.
- Controls information - this section contains information about each child control of the parent window, including size, location etc.
- Menu information - this section contains information needed for constructing the Window menu.

Each of these sections is introduced below then explained in detail in the following chapters.

Note that the Window information section and the Controls information section share similar structures in that each element describes a control and controls share similar structures. Since this documentation is designed as a reference, each control type is described individually. Although specific attributes vary from control type to control type, the basic control structure does not change, allowing for the development of fairly generic utilities.

Chapter 3: Sizing Information

That which we refer to as “Sizing information” is stored in the first record mark delimited sub-string of the window row in SYSREPOSWINS. It is a field mark delimited array made up as follows:-

< 1 > Currently always set to 300. The version of the structure layout.

< 2 > Set to the number of controls in the window, excluding the window object itself. That is, the count of controls in “Controls Information” - the third section of the window row.

< 3 > The fully qualified entity ID (the key to the SYSREPOS table) for the window’s style sheet.

Chapter 4: Window information

That which we refer to as the “Window information” section is stored in the second record mark delimited sub-string of the window row in SYSREPOSWINS. It is a value mark delimited array made up as follows :-

< 0, 1 > The name of the Window control, which serves as the parent of the other controls.

< 0, 2 > The name of the icon to display when the window is iconised (e.g. HAMMER0). This may safely be left blank as the fuller icon entity name stored later is actually used. It would seem that this is a historical artefact.

< 0, 3 > The type of the control, always set to WINDOW.

< 0, 4 > The parent of the control, null in the case of windows.

< 0, 5 > The X location of the window in pixels with 0 as the origin at the top left corner of the screen.

< 0, 6 > The Y location of the window in pixels with 0 as the origin at the top left corner of the screen.

< 0, 7 > The width of the window in pixels.

< 0, 8 > The depth of the window in pixels.

< 0, 9 > The title of the window.

< 0, 10 > The SDK Style of the Window. For a discussion of SDK styles see Appendix A. For a list of SDK styles for Window controls see Appendix B.

If SDK styles make no sense to you check the aforementioned appendixes before continuing.

This is a value (stored in hex format by the Form Designer) which is used to specify to the Windows API what characteristics the window will be created with. The default window style within OpenInsight is 0x92ef000, which creates a window with the following characteristics:

WS_Overlapped	0x00000000
WS_MaximizeBox	0x00010000
WS_MinimizeBox	0x00020000
WS_ThickFrame	0x00040000
WS_SysMenu	0x00080000
WS_Caption	0x00C00000
WS_ClipChildren	0x02000000
WS_Visible	0x10000000
WS_Popup	0x80000000

These style values can be mixed and matched at the discretion of the developer. Most of them can be set using the form designer by way of check boxes (see Figure 2). The remaining styles can be set by changing the window structure, for example by editing the corresponding SYSREPOSWINS row. The desired styles are added using a bitwise OR. For example, to add the WS_MinimizeBox (0x00020000) style to 0x92ed0000, follow these steps:

1. Determine the place of the hex digit affected by the style. Since the WS_MinimizeBox style is 0x00020000, the 5th digit from the right is affected.

2. Determine the value of the hex digit affected by the style. Since the current style is 0x92ed0000 and the 5th digit from the right is affected, the value of the hex digit is “d”, which is the decimal value 13.

3. Decompose the hex digit into its binary factors; binary factors are any powers of 2, like 1, 2, 4 and 8, 16, 32 and so on. A hex digit can be composed of the binary factors 1, 2, 4 and 8. The hex digit “d” (decimal 13) decomposes to 1, 4 and 8 (the combination of binary factors which adds up to 13).

4. If the hex digit of the desired style (2) is not in the list of binary factors (1, 4 and 8), then add it to the list (1, 2, 4, 8). Sum the resultant list (15) and convert to hex (“f”).

5. Place the new hex digit (“f”) in the window style. Remember that the fifth digit from the right was affected, so the new style is 0x92ef0000.

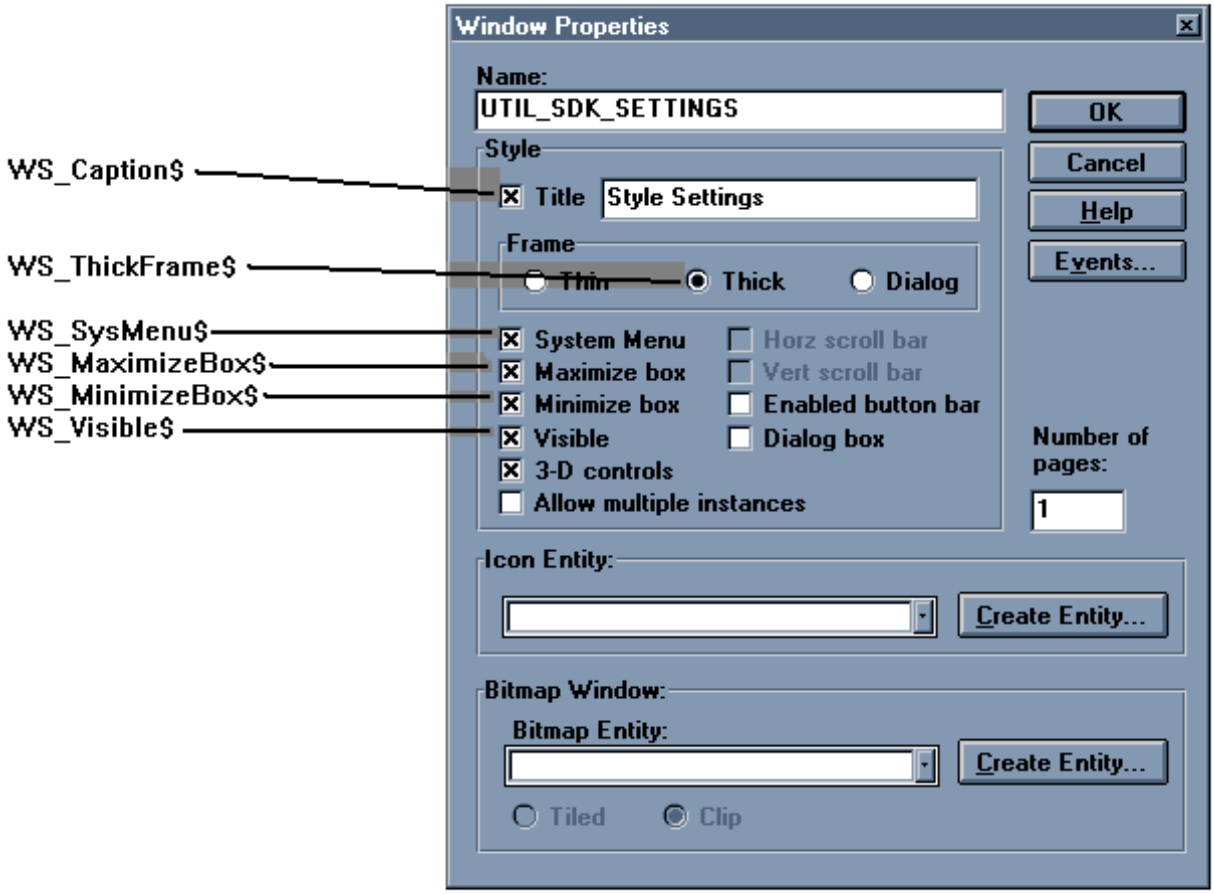

Figure 2 - Window SDK Styles

< 0, 11 > The Presentation Server (PS) style of the window. There are settings that are not adequately catered for by the Windows SDK style setting that PS still needs to be able to set. This additional style value allows these pieces of information to be sent to PS and thence to the WinAPI.

For details of the available PS SDK style bits see Appendix C.

< 0, 12 > The tab position for the window which is always set to 0.

< 0, 13 > Background colour of the window expressed as an RGB COLOUR VALUE value.

< 0, 14 > Unused. Set to 0.

< 0, 15 > Unused.

< 0, 16 > Unused.

< 0, 17 > Text mark delimited list of event handlers names for which a script has been defined. The event handler itself will be stored in SYSREPOSEVENTS with a key constructed as follows :-


```
AppName*EventName*WindowName.
```

e.g. SYSPROG*CLOSE*TEST.

< 0, 18 > Sub value delimited list of quick events . For quick event structure see Appendix D. Note that this array always has a trailing sub value mark.

< 0, 19 > Sub value delimited list of event handler names for which a quick event has been defined. Note that this array always has a trailing sub value mark.

< 0, 20 > Seems to be unused.

< 0, 21 > Seems to be unused.

< 0, 22 > Seems to be unused.

< 0, 23 > A text mark delimited list of tables used by the form. The Primary table is first followed by the other tables.

< 0, 24 > A sub valued array representing the IOOPTIONS properties of the form. There are ten options as follows:-

< 0, 0, 1 > Locking scheme

0 Pessimistic

1 Optimistic (not implemented)

< 0, 0, 2 > Nature of lock

For pessimistic 0 Exclusive

1 Shared

2 No lock

For optimistic 0 Compare all

1 Compare changes

2 No comparison

< 0, 0, 3 > Coordinate with file locks

0 No

1 Yes

< 0, 0, 4 > Ignore self locks flag (if true)

< 0, 0, 5 > Ignore subrow locks (if true - not implemented)

< 0, 0, 6 > Clear on write (if false)

< 0, 0, 7 > Transaction commit

0 Commit after DataSet commit

1 Don’t commit after DataSet commit

< 0, 0, 8 > Transaction rollback

0 Rollback if DataSet commit fails

1 Don’t rollback if DataSet commit fails

< 0, 0, 9 > Whether the form will automatically rollback transactions during CLEAR processing.

0 Do not rollback for DataSet bound forms

1 Do rollback for DataSet bound forms

< 0, 0, 10 > When required field processing is enforced

0 Do not allow users to tab off or otherwise leave empty required fields and check all required fields before saving.

1 Check all required fields only before saving, allowing the user to tab off of or otherwise leave empty required fields

< 0, 25 > Currently unused - set to 0

< 0, 26 > Currently unused - set to 0

< 0, 27 > Joined tables information

< 0, 0, 1, 1 > First Joined Table (the primary table of the form)

< 0, 0, 1, 2 > Null

< 0, 0, 1, 3, 1 > First Key Part

< 0, 0, 1, 3, 2 > Second Key Part

< 0, 0, 1, 3, 3 > Third Key Part

< 0, 0, 1, 3, n > Nth Key Part

< 0, 0, 1, 4 > Null

< 0, 0, 1, 5 > Null

< 0, 0, 1, 6 >	Null
< 0, 0, 1, 7 >	Allow insert if true
< 0, 0, 1, 8 >	Allow delete if true
< 0, 0, 1, 9 >	Explicit delete if true
< 0, 0, 1, 10 >	Implicit delete if true
< 0, 0, 2, 1 >	Second Joined Table
< 0, 0, 2, 2, 1 >	First Column to join to in Joined Table
< 0, 0, 2, 2, 2 >	Second Column to join to in Joined Table
< 0, 0, 2, 2, n >	Nth Column to join to in Joined Table
< 0, 0, 2, 3, 1 >	First Control to join from in Window
< 0, 0, 2, 3, 2 >	Second Control to join from in Window
< 0, 0, 2, 3, n >	Nth Control to join from in Window
< 0, 0, 2, 4 >	Null
< 0, 0, 2, 5, 1 >	Relationship between first column and control
	1 = Equals
	2 = Less than
	3 = Greater than
	4 = Less than or equal to
	5 = Greater than or equal to
	6 = Not equal to
< 0, 0, 2, 5, 2 >	Relationship between second column and control
< 0, 0, 2, 5, n >	Relationship between nth column and control
< 0, 0, 2, 6 >	Null
< 0, 0, 2, 7 >	Allow insert if true
< 0, 0, 2, 8 >	Allow delete if true

< 0, 0, 2, 9 >	Explicit delete if true
< 0, 0, 2, 10 >	Implicit delete if true
< 0, 0, n, 1 >	Nth Joined Table

< 0, 28 > Unused.

< 0, 29 > Unused.

< 0, 30 > Unused.

< 0, 31 > Unused.

< 0, 32 > A sub-valued list of PS Styles. This is used during window design and changing it makes very little difference. For default set to 0x7FFFE³0xF001E988.

< 0, 33 > The fully qualified repository identifier for the bitmap to display as the window's background.

< 0, 34 > The fully qualified repository identifier for the icon to display if the window is iconised.

< 0, 35 > Not used for windows

< 0, 36 > Not used for windows

< 0, 37 > A sub-valued list of PS Styles. This is used during window design and changing it makes very little difference. The only digits of significance for the developer are (from the right) digits 5 and above which describe how many pages are in the current window. e.g.

`0x10000`	One page
`0x40000`	Four pages
`0x130000`	Nineteen pages

< 0, 38 > A sub valued list of 1s corresponding to <0, 37>.

< 0, 39 > Not used in windows

< 0, 40 > Unknown but has the value null, 0, 1 or F.

< 0, 41 > Not used in windows

< 0, 42 > Not used in windows

< 0, 43 > Not used in windows

< 0, 44 > Not used in windows

< 0, 45 > Not used in windows

< 0, 46 > Not used in windows

< 0, 47 > Not used in windows

< 0, 48 > Not used in windows

< 0, 49 > Not used in windows

< 0, 50 > Not used in windows

Chapter 5 : Controls Information

That which we refer to as the “Controls information” section is stored in the third record mark delimited sub-string of the window row in SYSREPOSWINS. It is a field mark delimited array with one array element per window control. Each control in turn is value mark delimited. There are 16 control types to consider. In the order used in Form Designer these are

STATIC
PUSHBUTTON
PUSHBMP
ICON
GROUPBOX
BITMAP
CHECKBOX
CHECKBMP
EDITTABLE
EDITLINE
EDITBOX
COMBOBOX
LISTBOX
VSCROLL
HSCROLL
RADIOBUTTON
RADIOBMP.

Note when assembling windows programmatically or manually, groupboxes are always placed as the first controls in the controls information section.

As a further point, when controls are anchored right or bottom, the x and y positions are given as negative values indicating distance from the appropriate side of the window.

STATIC Control Type

< 0, 1 > The name of the Window control, unqualified by parent window name.

< 0, 2 > Null

< 0, 3 > The type of the control, always set to STATIC

< 0, 4 > The parent of the control, that is, the window name.

< 0, 5 > The X location of the static in pixels with 0 as the origin at the top left corner of the screen.

< 0, 6 > The Y location of the static in pixels with 0 as the origin at the top left corner of the screen.

< 0, 7 > The width of the static in pixels.

< 0, 8 > The depth of the static in pixels.

< 0, 9 > The text of the static.

< 0, 10 > The SDK Style of the Static. For a discussion of SDK styles see Appendix A. For a list of SDK styles for Static controls see Appendix E.

The default static style within OpenInsight is 0x50000000, which creates a static with the following characteristics

```
WS_Visible 0x10000000

WS_Child       0x40000000
```

in other words, a visible control which is a child of the parent window.

< 0, 11 > The Presentation Server (PS) style of the static. There are settings that are not adequately catered for by the Windows SDK style setting that PS still needs to be able to set. This additional style bit allows these pieces of information to be sent to PS and thence to the WinAPI.

For details of the available PS SDK style bits see Appendix C.

< 0, 12 > The tab position for the static which as statics may not be tabbed to is always set to 0.

< 0, 13 > Background colour of the static expressed as an RGB colour value. For default leave null.

< 0, 14 > Foreground colour of the static expressed as an RGB colour value. For default set to 0.

< 0, 15 > Font of the static. Text mark delimited array having structure as detailed in Appendix F. For default set to MS Sans Serif¹-11¹700¹0¹0¹0¹0¹34¹0¹1¹2¹1¹0¹0.

< 0, 16 > Unused

< 0, 17 > Unused.

< 0, 18 > Unused

< 0, 19 > Unused

< 0, 20 > Unused.

< 0, 21 > Unused.

< 0, 22 > Unused.

< 0, 23 > Unused

< 0, 24 > Unused

< 0, 25 > Unused. Set to 0.

< 0, 26 > Unused. Set to 0.

< 0, 27 > Unused

< 0, 28 > Unused. Set to <<None>>.

< 0, 29 > Unused. Set to <<None>>.

< 0, 30 > Unused. Set to <<None>>.

< 0, 31 > Unused. Set to 0.

< 0, 32 > A sub-valued list of PS Styles. This is used during window design and changing it makes very little difference. For default set to 0x7fffe³0xc001ee88

< 0, 33 > Unused.

< 0, 34 > Unused.

< 0, 35 > Unused.

< 0, 36 > Unused.

< 0, 37 > Unused.

< 0, 38 > Unused.

< 0, 39 > Unused.

< 0, 40 > Unused

< 0, 41 > Unused.

< 0, 42 > Unused.

< 0, 43 > Unused.

< 0, 44 > Unused.

< 0, 45 > DDE Link type. One of four literals, either OFF, WARM, HOT or AUTO.

< 0, 46 > DDE Item identifier.

< 0, 47 > DDE Topic identifier.

< 0, 48 > DDE Service identifier.

< 0, 49 > Unused.

< 0, 50 > Unused.

PUSHBUTTON Control Type

< 0, 1 > The name of the Window control, unqualified by parent window name.

< 0, 2 > Unused

< 0, 3 > The type of the control. Always set to PUSHBUTTON.

< 0, 4 > The parent of the control, that is, the window name.

< 0, 5 > The X location of the pushbutton in pixels with 0 as the origin at the top left corner of the screen.

< 0, 6 > The Y location of the pushbutton in pixels with 0 as the origin at the top left corner of the screen.

< 0, 7 > The width of the pushbutton in pixels.

< 0, 8 > The depth of the pushbutton in pixels.

< 0, 9 > The text of the pushbutton.

< 0, 10 > The SDK Style of the pushbutton. For a discussion of SDK styles see Appendix A. For a list of SDK styles for PushButton controls see Appendix H.

The default pushbutton style within OpenInsight is 0x50000000, which creates a pushbutton with the following characteristics

```
WS_Visible 0x10000000

WS_Child       0x40000000
```

in other words, a visible control which is a child of the parent window.

< 0, 11 > The Presentation Server (PS) style of the pushbutton. There are settings that are not adequately catered for by the Windows SDK style setting that PS still needs to be able to set. This additional style bit allows these pieces of information to be sent to PS and thence to the WinAPI.

For details of the available PS SDK style bits see Appendix C.

< 0, 12 > The tab position for the pushbutton within the window.

< 0, 13 > Not used.

< 0, 14 > Foreground colour of the pushbutton expressed as an RGB colour value. For default set to 0.

< 0, 15 > Font of the pushbutton. Text mark delimited array having structure as detailed in Appendix F. For default set to MS Sans Serif¹-11¹700¹0¹0¹0¹0¹34¹0¹1¹2¹1¹0¹.

< 0, 16 > Unused

< 0, 17 > Text mark delimited list of event handlers names for which a script has been defined. The event handler itself will be stored in SYSREPOSEVENTS with a key constructed as follows :-

```
AppName*EventName*WindowName.Control
```

e.g. SYSPROG*CLICK*TEST.BUTTON_1

< 0, 18 > Sub value delimited list of quick events . For quick event structure see Appendix D. Note that this array always has a trailing sub value mark.

< 0, 19 > Sub value delimited list of event handler names for which a quick event has been defined. Note that this array always has a trailing sub value mark.

< 0, 20 > Unused.

< 0, 21 > Unused.

< 0, 22 > Unused.

< 0, 23 > Unused.

< 0, 24 > Unused.

< 0, 25 > Unused. Set to 0.

< 0, 26 > Unused. Set to 0.

< 0, 27 > Unused.

< 0, 28 > Unused. For normal value set to <<None>>.

< 0, 29 > Unused. For normal value set to <<None>>.

< 0, 30 > Unused. For normal value set to <<None>>.

< 0, 31 > Unused. Set to 0.

< 0, 32 > A sub-valued list of PS Styles. This is used during window design and changing it makes very little difference. For default PushButton set to 0x7fffe³0xf0016100.

< 0, 33 > Unused.

< 0, 34 > Unused.

< 0, 35 > Unused.

< 0, 36 > Unused.

< 0, 37 > Unused.

< 0, 38 > Unused.

< 0, 39 > Unused.

< 0, 40 > Unused.

< 0, 41 > Unused.

< 0, 42 > Unused.

< 0, 43 > Unused.

< 0, 44 > Unused.

< 0, 45 > DDE Link type. One of four literals, either OFF, WARM, HOT or AUTO.

< 0, 46 > DDE Item identifier.

< 0, 47 > DDE Topic identifier.

< 0, 48 > DDE Service identifier.

< 0, 49 > Unused.

< 0, 50 > Unused.

PUSHBMP Control Type

< 0, 1 > The name of the Window control, unqualified by parent window name.

< 0, 2 > Unused

< 0, 3 > The type of the control. Always set to PUSHBMP.

< 0, 4 > The parent of the control, that is, the window name.

< 0, 5 > The X location of the pushbmp in pixels with 0 as the origin at the top left corner of the screen.

< 0, 6 > The Y location of the pushbmp in pixels with 0 as the origin at the top left corner of the screen.

< 0, 7 > The width of the pushbmp in pixels.

< 0, 8 > The depth of the pushbmp in pixels.

< 0, 9 > The text of the pushbmp. Note that the text may be pipe "|") delimited. The value before the pipe becomes available on the system STATUSLINE and the value after the pipe becomes available as bubble help. This applies unless PS Style value 400 is used to combine text (the portion after the pipe) and bitmap on the control.

< 0, 10 > The SDK Style of the pushbmp. For a discussion of SDK styles see Appendix A. For a list of SDK styles for Button controls see Appendix H.

The default pushbmp style within OpenInsight is 0x5000000b, which creates a pushbutton with the following characteristics

```
BS_OwnerDraw    0xB

WS_Visible 0x10000000

WS_Child        0x40000000
```

in other words, a visible control which is a child of the parent window with updates looked after by OpenInsight.

< 0, 11 > The Presentation Server (PS) style of the pushbmp. There are settings that are not adequately catered for by the Windows SDK style setting that PS still needs to be able to set. This additional style bit allows these pieces of information to be sent to PS and thence to the WinAPI.

For details of the available PS SDK style bits see Appendix C.

< 0, 12 > The tab position for the pushbmp within the window.

< 0, 13 > Unused.

< 0, 14 > Foreground colour of the pushbmp expressed as an RGB colour value. Used when combining text with bitmaps. For default set to 0.

< 0, 15 > Font of the pushbmp. Used when combining text with bitmaps. Text mark delimited array having structure as detailed in Appendix F. For default set to MS Sans Serif¹-11¹700¹0¹0¹0¹0¹34¹0¹1¹2¹1¹0¹.

< 0, 16 > Unused

< 0, 17 > Text mark delimited list of event handlers names for which a script has been defined. The event handler itself will be stored in SYSREPOSEVENTS with a key constructed as follows :-

```
AppName*EventName*WindowName.Control
```

e.g. SYSPROG*CLICK*TEST.BUTTON_1

< 0, 18 > Sub value delimited list of quick events . For quick event structure see Appendix D. Note that this array always has a trailing sub value mark.

< 0, 19 > Sub value delimited list of event handler names for which a quick event has been defined. Note that this array always has a trailing sub value mark.

< 0, 20 > Unused.

< 0, 21 > Unused.

< 0, 22 > Unused.

< 0, 23 > Unused.

< 0, 24 > Unused.

< 0, 25 > Unused. Set to 0.

< 0, 26 > Unused. Set to 0.

< 0, 27 > Unused.

< 0, 28 > Unused. For normal value set to <<None>>.

< 0, 29 > Unused. For normal value set to <<None>>.

< 0, 30 > Unused. For normal value set to <<None>>.

< 0, 31 > Unused. Set to 0.

< 0, 32 > A sub-valued list of PS Styles. This is used during window design and changing it makes very little difference. For default PushBmp set to 0x7fffe³0xf0016000.

< 0, 33 > Bitmap name. Fully qualified entity identifier. E.G. SYSPROG*IMAGE*BMP*OR_NEW.

< 0, 34 > Unused.

< 0, 35 > Unused.

< 0, 36 > Unused.

< 0, 37 > Unused.

< 0, 38 > Unused.

< 0, 39 > Unused.

< 0, 40 > Number of images in bitmap.

< 0, 41 > Unused.

< 0, 42 > Unused.

< 0, 43 > Unused.

< 0, 44 > Unused.

< 0, 45 > DDE Link type. One of four literals, either OFF, WARM, HOT or AUTO.

< 0, 46 > DDE Item identifier.

< 0, 47 > DDE Topic identifier.

< 0, 48 > DDE Service identifier.

< 0, 49 > Unused.

< 0, 50 > Unused.

Icon Control Type

< 0, 1 > The name of the Window control, unqualified by parent window name.

< 0, 2 > Set to an incrementing literal - the word ICONCTRL followed by a number followed by an underscore, representing the relative number of this icon within the window. The relative numbering starts at 2 so the first icon control is ICONCTRL2_, the second is ICONCTRL3_ etc.

< 0, 3 > The type of the control, always set to ICON.

< 0, 4 > The parent of the control, that is, the window name.

< 0, 5 > The X location of the icon in pixels with 0 as the origin at the top left corner of the screen.

< 0, 6 > The Y location of the icon in pixels with 0 as the origin at the top left corner of the screen.

< 0, 7 > Always set to 32.

< 0, 8 > Always set to 32..

< 0, 9 > The text of the static.

< 0, 10 > The SDK Style of the icon. For a discussion of SDK styles see Appendix A. Icons do not have specific SDK styles.

The default static style within OpenInsight is 0x70000000, which creates an icon with the following characteristics

```
WS_Visible 0x10000000

WS_Minimise     0x20000000

WS_Child        0x40000000
```

in other words, a visible minimised icon which is a child of the parent window.

< 0, 11 > The Presentation Server (PS) style of the icon. There are settings that are not adequately catered for by the Windows SDK style setting that PS still needs to be able to set. This additional style bit allows these pieces of information to be sent to PS and thence to the WinAPI.

For details of the available PS SDK style bits see Appendix C.

< 0, 12 > The tab position for the icon within the window.

< 0, 13 > Unused.

< 0, 14 > Unused.

< 0, 15 > Font of the icon. Text mark delimited array having structure as detailed in Appendix F. For default set to MS Sans Serif¹-11¹700¹0¹0¹0¹0¹34¹0¹1¹2¹1¹0¹0.

< 0, 16 > Unused

< 0, 17 > Unused.

< 0, 18 > Unused

< 0, 19 > Unused

< 0, 20 > Unused.

< 0, 21 > Unused.

< 0, 22 > Unused.

< 0, 23 > Unused

< 0, 24 > Unused

< 0, 25 > Unused. Set to 0.

< 0, 26 > Unused. Set to 0.

< 0, 27 > Unused

< 0, 28 > Unused. Set to <<None>>.

< 0, 29 > Unused. Set to <<None>>.

< 0, 30 > Unused. Set to <<None>>.

< 0, 31 > Unused. Set to 0.

< 0, 32 > A sub-valued list of PS Styles. This is used during window design and changing it makes very little difference. For default set to 0x7fffe³0xf0016d88.

< 0, 33 > Unused.

< 0, 34 > Icon name. Fully qualified entity identifier. E.G. SYSPROG*IMAGE*ICO*CSWS.

< 0, 35 > Unused.

< 0, 36 > Unused.

< 0, 37 > Unused.

< 0, 38 > Unused.

< 0, 39 > Unused.

< 0, 40 > Unused

< 0, 41 > Unused.

< 0, 42 > Unused.

< 0, 43 > Unused.

< 0, 44 > Unused.

< 0, 45 > Unused.

< 0, 46 > Unused.

< 0, 47 > Unused.

< 0, 48 > Unused.

< 0, 49 > Unused.

< 0, 50 > Unused.

GROUPBOX Control Type

< 0, 1 > The name of the Window control, unqualified by parent window name.

< 0, 2 > Null

< 0, 3 > The type of the control, always set to GROUPBOX

< 0, 4 > The parent of the control, that is, the window name.

< 0, 5 > The X location of the groupbox in pixels with 0 as the origin at the top left corner of the screen.

< 0, 6 > The Y location of the groupbox in pixels with 0 as the origin at the top left corner of the screen.

< 0, 7 > The width of the groupbox in pixels.

< 0, 8 > The depth of the groupbox in pixels.

< 0, 9 > The text of the groupbox.

< 0, 10 > The SDK Style of the Static. For a discussion of SDK styles see Appendix A. For a list of SDK styles for Static controls see Appendix E.

The default static style within OpenInsight is 0x50000007, which creates a static with the following characteristics

```
BS_GroupBox     0x7
WS_Visible 0x10000000
WS_Child        0x40000000
```

in other words, a visible groupbox which is a child of the parent window.

< 0, 11 > The Presentation Server (PS) style of the static. There are settings that are not adequately catered for by the Windows SDK style setting that PS still needs to be able to set. This additional style bit allows these pieces of information to be sent to PS and thence to the WinAPI.

For details of the available PS SDK style bits see Appendix C.

< 0, 12 > The tab position for the static which as groupboxes may not be tabbed to is always set to 0.

< 0, 13 > Background colour of the groupbox expressed as an RGB colour value. For default leave null.

< 0, 14 > Foreground colour of the groupbox expressed as an RGB colour value. For default set to 0.

< 0, 15 > Font of the groupbox. Text mark delimited array having structure as detailed in Appendix F. For default set to MS Sans Serif¹-11¹700¹0¹0¹0¹0¹34¹0¹1¹2¹1¹0¹0.

< 0, 16 > Unused

< 0, 17 > Unused.

< 0, 18 > Unused

< 0, 19 > Unused

< 0, 20 > Unused.

< 0, 21 > Unused.

< 0, 22 > Unused.

< 0, 23 > Unused

< 0, 24 > Unused

< 0, 25 > Unused. Set to 0.

< 0, 26 > Unused. Set to 0.

< 0, 27 > Unused

< 0, 28 > Unused. Set to <<None>>.

< 0, 29 > Unused. Set to <<None>>.

< 0, 30 > Unused. Set to <<None>>.

< 0, 31 > Unused. Set to 0.

< 0, 32 > A sub-valued list of PS Styles. This is used during window design and changing it makes very little difference. For default set to 0x7fffe³0xf001ec08.

< 0, 33 > Unused.

< 0, 34 > Unused.

< 0, 35 > Unused.

< 0, 36 > Unused.

< 0, 37 > Unused.

< 0, 38 > Unused.

< 0, 39 > Unused.

< 0, 40 > Unused

< 0, 41 > Unused.

< 0, 42 > Unused.

< 0, 43 > Unused.

< 0, 44 > Unused.

< 0, 45 > DDE Link type. One of four literals, either OFF, WARM, HOT or AUTO.

< 0, 46 > DDE Item identifier.

< 0, 47 > DDE Topic identifier.

< 0, 48 > DDE Service identifier.

< 0, 49 > Unused.

< 0, 50 > Unused.

BITMAP Control Type

< 0, 1 > The name of the Window control, unqualified by parent window name.

< 0, 2 > Null

< 0, 3 > The type of the control, always set to BITMAP.

< 0, 4 > The parent of the control, that is, the window name.

< 0, 5 > The X location of the bitmap in pixels with 0 as the origin at the top left corner of the screen.

< 0, 6 > The Y location of the bitmap in pixels with 0 as the origin at the top left corner of the screen.

< 0, 7 > The width of the bitmap in pixels.

< 0, 8 > The depth of the bitmap in pixels.

< 0, 9 > Unused.

< 0, 10 > The SDK Style of the bitmap. For a discussion of SDK styles see Appendix A. Bitmaps do not have specific SDK styles

The default bitmap style within OpenInsight is 0x5000000a, which creates a bitmap with the following characteristics

```
WS_Visible 0x10000000

WS_Child       0x40000000
```

in other words, a visible bitmap which is a child of the parent window.

< 0, 11 > The Presentation Server (PS) style of the bitmap. There are settings that are not adequately catered for by the Windows SDK style setting that PS still needs to be able to set. This additional style bit allows these pieces of information to be sent to PS and thence to the WinAPI.

For details of the available PS SDK style bits see Appendix C.

< 0, 12 > The tab position for the bitmap which as bitmaps may not be tabbed to is always set to 0.

< 0, 13 > Unused.

< 0, 14 > Unused. Set to 0.

< 0, 15 > Unused. Set to MS Sans Serif¹-11¹700¹0¹0¹0¹0¹34¹0¹1¹2¹1¹0¹.

< 0, 16 > Unused

< 0, 17 > Unused.

< 0, 18 > Unused

< 0, 19 > Unused

< 0, 20 > Unused.

< 0, 21 > Unused.

< 0, 22 > Unused.

< 0, 23 > Unused

< 0, 24 > Unused

< 0, 25 > Unused. Set to 0.

< 0, 26 > Unused. Set to 0.

< 0, 27 > Unused

< 0, 28 > Unused. Set to <<None>>.

< 0, 29 > Unused. Set to <<None>>.

< 0, 30 > Unused. Set to <<None>>.

< 0, 31 > Unused. Set to 0.

< 0, 32 > A sub-valued list of PS Styles. This is used during window design and changing it makes very little difference. For default set to 0x7fffc³ 0xf0016a08.

< 0, 33 > Bitmap name. Fully qualified entity identifier. E.G. SYSPROG*IMAGE*BMP*OR_NEW.

< 0, 34 > Unused.

< 0, 35 > Unused.

< 0, 36 > Unused.

< 0, 37 > Unused.

< 0, 38 > Unused.

< 0, 39 > Unused.

< 0, 40 > Unused

< 0, 41 > Unused.

< 0, 42 > Unused.

< 0, 43 > Unused.

< 0, 44 > Unused.

< 0, 45 > Unused.

< 0, 46 > Unused.

< 0, 47 > Unused.

< 0, 48 > Unused.

< 0, 49 > Unused.

< 0, 50 > Unused

CHECKBOX Control Type

< 0, 1 > The name of the Window control, unqualified by parent window name.

< 0, 2 > Unused

< 0, 3 > The type of the control. Always set to CHECKBOX.

< 0, 4 > The parent of the control, that is, the window name.

< 0, 5 > The X location of the checkbox in pixels with 0 as the origin at the top left corner of the screen.

< 0, 6 > The Y location of the checkbox in pixels with 0 as the origin at the top left corner of the screen.

< 0, 7 > The width of the checkbox in pixels.

< 0, 8 > The depth of the checkbox in pixels.

< 0, 9 > The text of the checkbox.

< 0, 10 > The SDK Style of the checkbox. For a discussion of SDK styles see Appendix A. For a list of SDK styles for Button controls see Appendix H.

The default checkbox style within OpenInsight is 0x50000003, which creates a checkbox with the following characteristics

```
BS_AutoCheckBox 0x3
WS_Visible      0x10000000
WS_Child        0x40000000
```

in other words, a visible checkbox which is a child of the parent window.

< 0, 11 > The Presentation Server (PS) style of the checkbox. There are settings that are not adequately catered for by the Windows SDK style setting that PS still needs to be able to set. This additional style bit allows these pieces of information to be sent to PS and thence to the WinAPI.

For details of the available PS SDK style bits see Appendix C.

< 0, 12 > The tab position for the checkbox within the window.

< 0, 13 > Background colour of the checkbox expressed as an RGB colour value. For default leave null.

< 0, 14 > Foreground colour of the checkbox expressed as an RGB colour value. For default set to 0.

< 0, 15 > Font of the checkbox.. Text mark delimited array having structure as detailed in Appendix F. For default set to MS Sans Serif¹-11¹700¹0¹0¹0¹0¹34¹0¹1¹2¹1¹0¹.

< 0, 16 > Unused.

< 0, 17 > Text mark delimited list of event handlers names for which a script has been defined. The event handler itself will be stored in SYSREPOSEVENTS with a key constructed as follows :-

```
AppName*EventName*WindowName.Control
```

e.g. SYSPROG*CLICK*TEST.CHECKBOX_1

< 0, 18 > Sub value delimited list of quick events . For quick event structure see Appendix D. Note that this array always has a trailing sub value mark.

< 0, 19 > Sub value delimited list of event handler names for which a quick event has been defined. Note that this array always has a trailing sub value mark.

< 0, 20 > Unused.

< 0, 21 > Unused.

< 0, 22 > Unused.

< 0, 23 > Table name that checkbox is associated with if this is a data aware control.

< 0, 24 > Column name that checkbox is associated with if this is a data aware control.

< 0, 25 > Unused. Set to 0.

< 0, 26 > Unused. Set to 0.

< 0, 27 > Unused.

< 0, 28 > Unused. For normal value set to <<None>>.

< 0, 29 > Unused. For normal value set to <<None>>.

< 0, 30 > Unused. For normal value set to <<None>>.

< 0, 31 > Unused. Set to 0.

< 0, 32 > A sub-valued list of PS Styles. This is used during window design and changing it makes very little difference. For default CheckBox set to 0x7fffe³0xf001ef88.

< 0, 33 > Unused

< 0, 34 > Unused.

< 0, 35 > Unused.

< 0, 36 > Value. 1 for checked. O for unchecked. Null by default..

< 0, 37 > Unused.

< 0, 38 > Unused.

< 0, 39 > Unused.

< 0, 40 > Unused.

< 0, 41 > Unused.

< 0, 42 > Unused.

< 0, 43 > Unused.

< 0, 44 > Unused.

< 0, 45 > Unused.

< 0, 46 > Unused.

< 0, 47 > Unused.

< 0, 48 > Unused.

< 0, 49 > Unused.

< 0, 50 > Unused.

< 0, 51 > Used in Lotus Notes connection to identify column to update. For structure see Appendix K.

CHECKBMP Control Type

< 0, 1 > The name of the Window control, unqualified by parent window name.

< 0, 2 > Unused

< 0, 3 > The type of the control. Always set to CHECKBMP.

< 0, 4 > The parent of the control, that is, the window name.

< 0, 5 > The X location of the checkbmp in pixels with 0 as the origin at the top left corner of the screen.

< 0, 6 > The Y location of the checkbmp in pixels with 0 as the origin at the top left corner of the screen.

< 0, 7 > The width of the checkbmp in pixels.

< 0, 8 > The depth of the checkbmp in pixels.

< 0, 9 > The text of the checkbmp. As with PushBmps, portion before | becomes statusline text, portion after | becomes bubble help.

< 0, 10 > The SDK Style of the checkbmp. For a discussion of SDK styles see Appendix A. For a list of SDK styles for Button controls see Appendix H.

The default checkbmp style within OpenInsight is 0x5000000b, which creates a checkbmp with the following characteristics

```
BS_OwnerDraw    0xB

WS_Visible 0x10000000

WS_Child        0x40000000
```

in other words, a visible control which is a child of the parent window with updates looked after by OpenInsight.

< 0, 11 > The Presentation Server (PS) style of the checkbmp. There are settings that are not adequately catered for by the Windows SDK style setting that PS still needs to be able to set. This additional style bit allows these pieces of information to be sent to PS and thence to the WinAPI.

For details of the available PS SDK style bits see Appendix C.

< 0, 12 > The tab position for the checkbmp within the window.

< 0, 13 > Unused.

< 0, 14 > Foreground colour of the checkbmp expressed as an RGB colour value. For default set to 0.

< 0, 15 > Font of the checkbmp. Used when combining text with bitmaps. Text mark delimited array having structure as detailed in Appendix F. For default set to MS Sans Serif¹-11¹700¹0¹0¹0¹0¹34¹0¹1¹2¹1¹0¹..

< 0, 16 > Unused.

< 0, 17 > Text mark delimited list of event handlers names for which a script has been defined. The event handler itself will be stored in SYSREPOSEVENTS with a key constructed as follows :-

```
AppName*EventName*WindowName.Control
```

e.g. SYSPROG*CLICK*TEST.CHECKBMP_1

< 0, 18 > Sub value delimited list of quick events . For quick event structure see Appendix D. Note that this array always has a trailing sub value mark.

< 0, 19 > Sub value delimited list of event handler names for which a quick event has been defined. Note that this array always has a trailing sub value mark.

< 0, 20 > Unused.

< 0, 21 > Unused.

< 0, 22 > Unused.

< 0, 23 > Table name that checkbmp is associated with if this is a data aware control.

< 0, 24 > Column name that checkbmp is associated with if this is a data aware control.

< 0, 25 > Unused. Set to 0.

< 0, 26 > Unused. Set to 0.

< 0, 27 > Unused.

< 0, 28 > Unused. For normal value set to <<None>>.

< 0, 29 > Unused. For normal value set to <<None>>.

< 0, 30 > Unused. For normal value set to <<None>>.

< 0, 31 > Unused. Set to 0.

< 0, 32 > A sub-valued list of PS Styles. This is used during window design and changing it makes very little difference. For default Checkbmp set to 0x7ffee³0xf0016080.

< 0, 33 > Bitmap name. Fully qualified entity identifier. E.G. SYSPROG*IMAGE*BMP*OR_NEW.

< 0, 34 > Unused.

< 0, 35 > Unused.

< 0, 36 > Value. 1 for checked. O for unchecked. Null by default

< 0, 37 > Unused.

< 0, 38 > Unused.

< 0, 39 > Unused.

< 0, 40 > Number of images in bitmap.

< 0, 41 > Unused.

< 0, 42 > Unused.

< 0, 43 > Unused.

< 0, 44 > Unused.

< 0, 45 > Unused.

< 0, 46 > Unused.

< 0, 47 > Unused.

< 0, 48 > Unused.

< 0, 49 > Unused.

< 0, 50 > Unused.

< 0, 51 > Used in Lotus Notes connection to identify column to update. For structure see Appendix K.

EDITTABLE Control Type

< 0, 1 > The name of the Window control, unqualified by parent window name.

< 0, 2 > Unused

< 0, 3 > The type of the control, always set to EDITTABLE.

< 0, 4 > The parent of the control, that is, the window name.

< 0, 5 > The X location of the edittable in pixels with 0 as the origin at the top left corner of the screen.

< 0, 6 > The Y location of the edittable in pixels with 0 as the origin at the top left corner of the screen.

< 0, 7 > The width of the edittable in pixels.

< 0, 8 > The depth of the edittable in pixels.

< 0, 9 > Unused.

< 0, 10 > The SDK Style of the edittable. For a discussion of SDK styles see Appendix A. Note that EditTables are in fact a licenced product from the ProtoView Corporation, therefore the SDK is licenced to them, not Microsoft. For a list of SDK styles for Edittable controls see Appendix L

The default edittable style within OpenInsight is 0x508041fc, which creates an edittable with the following characteristics

```
DTS_Resize 0x8
DTS_Hgrid       0x40
DTS_Vgrid       0x80
DTS_RowSelect   0x100
DTS_RowNumbers  0x4000
WS_Border       0x800000
WS_Visible 0x10000000
WS_Child        0x40000000
```

in other words, a visible edittable which is a child of the parent window, has both vertical and horizontal grids and a border, allows row selection, shows row numbers, permits resizing.

< 0, 11 > The Presentation Server (PS) style of the edittable. There are settings that are not adequately catered for by the Windows SDK style setting that PS still needs to be able to set. This additional style bit allows these pieces of information to be sent to PS and thence to the WinAPI.

For details of the available PS SDK style bits see Appendix C.

< 0, 12 > The tab position for the edittable within the window.

< 0, 13 > Background colour of the edittable expressed as an RGB colour value. For default leave null.

< 0, 14 > Foreground colour of the edittable expressed as an RGB colour value. For default set to 0.

< 0, 15 > Font of the edittable. Text mark delimited array having structure as detailed in Appendix F. For default set to MS Sans Serif¹-11¹700¹0¹0¹0¹0¹34¹0¹1¹2¹1¹0¹.

< 0, 16 > Unused

< 0, 17 > Text mark delimited list of event handlers names for which a script has been defined. The event handler itself will be stored in SYSREPOSEVENTS with a key constructed as follows :-

```
AppName*EventName*WindowName.Control
```

e.g. SYSPROG*LOSTFOCUS*TEST.EDITLINE_1

< 0, 18 > Sub value delimited list of quick events . For quick event structure see Appendix D. Note that this array always has a trailing sub value mark.

< 0, 19 > Sub value delimited list of event handler names for which a quick event has been defined. Note that this array always has a trailing sub value mark.

< 0, 20 > Unused.

< 0, 21 > Unused.

< 0, 22 > Unused.

< 0, 23 > Text mark delimited list of table names that the edittable is associated with (in edittable column order) if this is a data aware control.

< 0, 24 > Text mark delimited list of column names that the edittable is associated with (in edittable column order) if this is a data aware control.

< 0, 25 > Unused. Set to 0.

< 0, 26 > Unused. Set to 0.

< 0, 27 > Unused.

< 0, 28 > Text mark delimited list of iconvs for this edittable (in edittable column order). For normal value set to <<None>>. To default to dictionary conversion set to <<Default>>.

< 0, 29 > Text mark delimited list of oconvs for this edittable (in edittable column order). For normal value set to <<None>>. To default to dictionary conversion set to <<Default>>.

< 0, 30 > Text mark delimited list of defaults for this edittable (in edittable column order). . For normal value set to <<None>>. To default to dictionary default set to <<Default>>.

< 0, 31 > Text mark delimited list of zeroes - as many as there are columns.

< 0, 32 > A sub-valued list of PS Styles. This is used during window design and changing it makes very little difference. For default set to 0x7e27e³0xc001e308..

< 0, 33 > Unused.

< 0, 34 > Unused.

< 0, 35 > Unused.

< 0, 36 > Unused.

< 0, 37 > Unused.

< 0, 38 > Unused.

< 0, 39 > Text mark delimited list of data widths for the columns in the edittable.

< 0, 40 > Number of columns in the edittable

< 0, 41 > Row limit..

< 0, 42 > Text mark delimited list of decimal SDK styles for the columns in the edittable. For a list of SDK styles for Edittable controls see Appendix L. N.B. these values are in decimal rather than the more usual hexadecimal.

< 0, 43 > Text mark delimited list of column widths for the edittable. Note that the first value corresponds to the width of the

button, not the width of the first column. All other column widths are offset accordingly.

< 0, 44 > Text mark delimited set of column labels for edittable..

< 0, 45 > DDE Link type. One of four literals, either OFF, WARM, HOT or AUTO.

< 0, 46 > DDE Item identifier.

< 0, 47 > DDE Topic identifier.

< 0, 48 > DDE Service identifier.

< 0, 49 > Unused.

< 0, 50 > Unused.

< 0, 51 > Used in Lotus Notes connection to identify column(s) to update. For structure see Appendix K.

EDITLINE Control Type

< 0, 1 > The name of the Window control, unqualified by parent window name.

< 0, 2 > Unused

< 0, 3 > The type of the control, always set to EDITFIELD.

< 0, 4 > The parent of the control, that is, the window name.

< 0, 5 > The X location of the editline in pixels with 0 as the origin at the top left corner of the screen.

< 0, 6 > The Y location of the editline in pixels with 0 as the origin at the top left corner of the screen.

< 0, 7 > The width of the editline in pixels.

< 0, 8 > The depth of the editline in pixels.

< 0, 9 > The text of the editline.

< 0, 10 > The SDK Style of the editline. For a discussion of SDK styles see Appendix A. For a list of SDK styles for Editline controls see Appendix G.

The default editline style within OpenInsight is 0x50800080, which creates an editline with the following characteristics

```
ES_AutoHScroll  0x80

WS_Border       0x800000

WS_Visible 0x10000000

WS_Child        0x40000000
```

in other words, a visible control which is a child of the parent window, scrolls automatically horizontally and has a border.

< 0, 11 > The Presentation Server (PS) style of the editline. There are settings that are not adequately catered for by the Windows SDK style setting that PS still needs to be able to set. This additional style bit allows these pieces of information to be sent to PS and thence to the WinAPI.

For details of the available PS SDK style bits see Appendix C.

< 0, 12 > The tab position for the editline within the window.

< 0, 13 > Background colour of the editline expressed as an RGB colour value. For default leave null.

< 0, 14 > Foreground colour of the editline expressed as an RGB colour value. For default set to 0.

< 0, 15 > Font of the editline. Text mark delimited array having structure as detailed in Appendix F. For default set to MS Sans Serif¹-11¹700¹0¹0¹0¹0¹34¹0¹1¹2¹1¹0¹.

< 0, 16 > Unused

< 0, 17 > Text mark delimited list of event handlers names for which a script has been defined. The event handler itself will be stored in SYSREPOSEVENTS with a key constructed as follows :-

```
AppName*EventName*WindowName.Control
```

e.g. SYSPROG*LOSTFOCUS*TEST.EDITLINE_1

< 0, 18 > Sub value delimited list of quick events . For quick event structure see Appendix D. Note that this array always has a trailing sub value mark.

< 0, 19 > Sub value delimited list of event handler names for which a quick event has been defined. Note that this array always has a trailing sub value mark.

< 0, 20 > Unused.

< 0, 21 > Unused.

< 0, 22 > Unused.

< 0, 23 > Table name that editline is associated with if this is a data aware control.

< 0, 24 > Column name that editline is associated with if this is a data aware control.

< 0, 25 > Unused. Set to 0.

< 0, 26 > Unused. Set to 0.

< 0, 27 > Unused.

< 0, 28 > Iconv. For normal value set to <<None>>. To default to dictionary conversion set to <<Default>>.

< 0, 29 > Oconv. For normal value set to <<None>>. To default to dictionary conversion set to <<Default>>.

< 0, 30 > Default . For normal value set to <<None>>. To default to dictionary default set to <<Default>>.

< 0, 31 > Unused. Set to 0.

< 0, 32 > A sub-valued list of PS Styles. This is used during window design and changing it makes very little difference. For default set to 0x7fffe³0xf001ee88.

< 0, 33 > Unused.

< 0, 34 > Unused.

< 0, 35 > Unused.

< 0, 36 > Unused.

< 0, 37 > Unused.

< 0, 38 > Unused.

< 0, 39 > Unused.

< 0, 40 > Unused

< 0, 41 > Unused.

< 0, 42 > Unused.

< 0, 43 > Unused.

< 0, 44 > Unused.

< 0, 45 > DDE Link type. One of four literals, either OFF, WARM, HOT or AUTO.

< 0, 46 > DDE Item identifier.

< 0, 47 > DDE Topic identifier.

< 0, 48 > DDE Service identifier.

< 0, 49 > Unused.

< 0, 50 > Unused.

< 0, 51 > Used in Lotus Notes connection to identify column to update. For structure see Appendix K.

EDITBOX Control Type

< 0, 1 > The name of the Window control, unqualified by parent window name.

< 0, 2 > Unused

< 0, 3 > The type of the control, always set to EDITBOX.

< 0, 4 > The parent of the control, that is, the window name.

< 0, 5 > The X location of the editbox in pixels with 0 as the origin at the top left corner of the screen.

< 0, 6 > The Y location of the editbox in pixels with 0 as the origin at the top left corner of the screen.

< 0, 7 > The width of the editbox in pixels.

< 0, 8 > The depth of the editbox in pixels.

< 0, 9 > The text of the editbox.

< 0, 10 > The SDK Style of the editbox. For a discussion of SDK styles see Appendix A. For a list of SDK styles for Editbox controls see Appendix G.

The default editbox style within OpenInsight is 0x50b00044, which creates an editbox with the following characteristics

```
ES_MultiLine    0x4

ES_AutoVScroll  0x40

WS_Hscroll0x100000

WS_Vscroll0x200000

WS_Border       0x800000

WS_Visible0x10000000

WS_Child        0x40000000
```

in other words, a visible multiline edit control which is a child of the parent window, scrolls automatically vertically, has horizontal and vertical scroll bars and has a border.

< 0, 11 > The Presentation Server (PS) style of the editbox. There are settings that are not adequately catered for by the Windows SDK style setting that PS still needs to be able to set. This

additional style bit allows these pieces of information to be sent to PS and thence to the WinAPI.

For details of the available PS SDK style bits see Appendix C.

< 0, 12 > The tab position for the editbox within the window.

< 0, 13 > Background colour of the editbox expressed as an RGB colour value. For default leave null.

< 0, 14 > Foreground colour of the editbox expressed as an RGB colour value. For default set to 0.

< 0, 15 > Font of the editbox. Text mark delimited array having structure as detailed in Appendix F. For default set to MS Sans Serif¹-11¹700¹0¹0¹0¹0¹34¹0¹1¹2¹1¹0¹.

< 0, 16 > Unused

< 0, 17 > Text mark delimited list of event handlers names for which a script has been defined. The event handler itself will be stored in SYSREPOSEVENTS with a key constructed as follows :-

```
AppName*EventName*WindowName.Control
```

e.g. SYSPROG*GOTFOCUS*TEST.EDITBOX_1

< 0, 18 > Sub value delimited list of quick events . For quick event structure see Appendix D. Note that this array always has a trailing sub value mark.

< 0, 19 > Sub value delimited list of event handler names for which a quick event has been defined. Note that this array always has a trailing sub value mark.

< 0, 20 > Unused.

< 0, 21 > Unused.

< 0, 22 > Unused.

< 0, 23 > Table name that editbox is associated with if this is a data aware control.

< 0, 24 > Column name that editbox is associated with if this is a data aware control.

< 0, 25 > Unused. Set to 0.

< 0, 26 > Unused. Set to 0.

< 0, 27 > Unused.

< 0, 28 > Iconv. For normal value set to <<None>>. To default to dictionary conversion set to <<Default>>.

< 0, 29 > Oconv. For normal value set to <<None>>. To default to dictionary conversion set to <<Default>>.

< 0, 30 > Default . For normal value set to <<None>>. To default to dictionary default set to <<Default>>.

< 0, 31 > Unused. Set to 0.

< 0, 32 > A sub-valued list of PS Styles. This is used during window design and changing it makes very little difference. For default set to 0x7fffe³0xf001e688.

< 0, 33 > Unused.

< 0, 34 > Unused.

< 0, 35 > Unused.

< 0, 36 > Unused.

< 0, 37 > Unused.

< 0, 38 > Unused.

< 0, 39 > Unused.

< 0, 40 > Unused

< 0, 41 > Unused.

< 0, 42 > Unused.

< 0, 43 > Unused.

< 0, 44 > Unused.

< 0, 45 > DDE Link type. One of four literals, either OFF, WARM, HOT or AUTO.

< 0, 46 > DDE Item identifier.

< 0, 47 > DDE Topic identifier.

< 0, 48 > DDE Service identifier.

< 0, 49 > Unused.

< 0, 50 > Unused.

< 0, 51 > Used in Lotus Notes connection to identify column to update. For structure see Appendix K.

COMBOBOX Control Type

< 0, 1 > The name of the Window control, unqualified by parent window name.

< 0, 2 > Unused

< 0, 3 > The type of the control, always set to COMBOBOX.

< 0, 4 > The parent of the control, that is, the window name.

< 0, 5 > The X location of the combobox in pixels with 0 as the origin at the top left corner of the screen.

< 0, 6 > The Y location of the combobox in pixels with 0 as the origin at the top left corner of the screen.

< 0, 7 > The width of the combobox in pixels.

< 0, 8 > The depth of the combobox in pixels.

< 0, 9 > Default value for combobox..

< 0, 10 > The SDK Style of the combobox. For a discussion of SDK styles see Appendix A. For a list of SDK styles for combobox controls see Appendix I.

The default combobox style within OpenInsight is 0x50000003, which creates a combobox with the following characteristics

```
CBS_DropDownList     0x3

WS_Visible      0x10000000

WS_Child             0x40000000
```

in other words, a visible drop down list which is a child of the parent window.

< 0, 11 > The Presentation Server (PS) style of the combobox. There are settings that are not adequately catered for by the Windows SDK style setting that PS still needs to be able to set. This additional style bit allows these pieces of information to be sent to PS and thence to the WinAPI.

For details of the available PS SDK style bits see Appendix C.

< 0, 12 > The tab position for the combobox within the window.

< 0, 13 > Background colour of the combobox expressed as an RGB colour value. For default leave null.

< 0, 14 > Foreground colour of the combobox expressed as an RGB colour value. For default set to 0.

< 0, 15 > Font of the combobox. Text mark delimited array having structure as detailed in Appendix F. For default set to MS Sans Serif¹-11¹700¹0¹0¹0¹0¹34¹0¹1¹2¹1¹0¹.

< 0, 16 > Unused

< 0, 17 > Text mark delimited list of event handlers names for which a script has been defined. The event handler itself will be stored in SYSREPOSEVENTS with a key constructed as follows :-

```
AppName*EventName*WindowName.Control
```

e.g. SYSPROG*LOSTFOCUS*TEST.COMBOBOX_1

< 0, 18 > Sub value delimited list of quick events . For quick event structure see Appendix D. Note that this array always has a trailing sub value mark.

< 0, 19 > Sub value delimited list of event handler names for which a quick event has been defined. Note that this array always has a trailing sub value mark.

< 0, 20 > Unused.

< 0, 21 > Unused.

< 0, 22 > Unused.

< 0, 23 > Table name that combobox is associated with if this is a data aware control.

< 0, 24 > Column name that combobox is associated with if this is a data aware control.

< 0, 25 > Unused. Set to 0.

< 0, 26 > Unused. Set to 0.

< 0, 27 > Unused.

< 0, 28 > Iconv. For normal value set to <<None>>. To default to dictionary conversion set to <<Default>>.

< 0, 29 > Oconv. For normal value set to <<None>>. To default to dictionary conversion set to <<Default>>.

< 0, 30 > Unused. For normal value set to <<None>>. To default to dictionary default set to <<Default>>.

< 0, 31 > Unused. Set to 0.

< 0, 32 > A sub-valued list of PS Styles. This is used during window design and changing it makes very little difference. For default set to 0x7fffe³0xf001ef88.

< 0, 33 > Unused.

< 0, 34 > Unused.

< 0, 35 > Text mark delimited set of values for combobox.

< 0, 36 > Unused.

< 0, 37 > Unused.

< 0, 38 > Unused.

< 0, 39 > Unused.

< 0, 40 > Unused

< 0, 41 > Unused.

< 0, 42 > If set to 8 forces uppercase in combobox.

< 0, 43 > Unused.

< 0, 44 > Unused.

< 0, 45 > DDE Link type. One of four literals, either OFF, WARM, HOT or AUTO.

< 0, 46 > DDE Item identifier.

< 0, 47 > DDE Topic identifier.

< 0, 48 > DDE Service identifier.

< 0, 49 > Unused.

< 0, 50 > Unused.

< 0, 51 > Used in Lotus Notes connection to identify column to update. For structure see Appendix K.

LISTBOX Control Type

< 0, 1 > The name of the Window control, unqualified by parent window name.

< 0, 2 > Unused

< 0, 3 > The type of the control, always set to LISTBOX.

< 0, 4 > The parent of the control, that is, the window name.

< 0, 5 > The X location of the listbox in pixels with 0 as the origin at the top left corner of the screen.

< 0, 6 > The Y location of the listbox in pixels with 0 as the origin at the top left corner of the screen.

< 0, 7 > The width of the listbox in pixels.

< 0, 8 > The depth of the listbox in pixels.

< 0, 9 > Unused.

< 0, 10 > The SDK Style of the listbox. For a discussion of SDK styles see Appendix A. For a list of SDK styles for listbox controls see Appendix J.

The default listbox style within OpenInsight is 0x50a00183, which creates a checkbox with the following characteristics

```
LBS_Notify           0x1
LBS_Sort                  0x2
LBS_UseTabStops      0x80
LBS_NoIntegralHeight 0x100
WS_Vscroll           0x200000
WS_Border                 0x800000
WS_Visible           0x10000000
WS_Child                  0x40000000
```

in other words, a visible sorted listbox with a border and scroll bars if necessary which is a child of the parent window.

< 0, 11 > The Presentation Server (PS) style of the listbox. There are settings that are not adequately catered for by the Windows

SDK style setting that PS still needs to be able to set. This additional style bit allows these pieces of information to be sent to PS and thence to the WinAPI.

For details of the available PS SDK style bits see Appendix C.

< 0, 12 > The tab position for the listbox within the window.

< 0, 13 > Background colour of the combobox expressed as an RGB colour value. For default leave null.

< 0, 14 > Foreground colour of the combobox expressed as an RGB colour value. For default set to 0.

< 0, 15 > Font of the combobox. Text mark delimited array having structure as detailed in Appendix F. For default set to MS Sans Serif¹-11¹700¹0¹0¹0¹0¹34¹0¹1¹2¹1¹0¹.

< 0, 16 > Unused

< 0, 17 > Text mark delimited list of event handlers names for which a script has been defined. The event handler itself will be stored in SYSREPOSEVENTS with a key constructed as follows :-

```
AppName*EventName*WindowName.Control
```

e.g. SYSPROG*LOSTFOCUS*TEST.LISTBOX_1

< 0, 18 > Sub value delimited list of quick events . For quick event structure see Appendix D. Note that this array always has a trailing sub value mark.

< 0, 19 > Sub value delimited list of event handler names for which a quick event has been defined. Note that this array always has a trailing sub value mark.

< 0, 20 > Unused.

< 0, 21 > Unused.

< 0, 22 > Unused.

< 0, 23 > Table name that listbox is associated with if this is a data aware control.

< 0, 24 > Column name that listbox is associated with if this is a data aware control.

< 0, 25 > Unused. Set to 0.

< 0, 26 > Unused. Set to 0.

< 0, 27 > Unused.

< 0, 28 > Iconv. For normal value set to <<None>>. To default to dictionary conversion set to <<Default>>.

< 0, 29 > Oconv. For normal value set to <<None>>. To default to dictionary conversion set to <<Default>>.

< 0, 30 > Unused. For normal value set to <<None>>. To default to dictionary default set to <<Default>>.

< 0, 31 > Unused. Set to 0.

< 0, 32 > A sub-valued list of PS Styles. This is used during window design and changing it makes very little difference. For default set to 0x7fff6³0xf001c208.

< 0, 33 > The fully qualified repository identifier for the bitmap to use within the listbox on each line.

< 0, 34 > Unused.

< 0, 35 > Text mark delimited set of values for listbox.

< 0, 36 > Unused.

< 0, 37 > Unused.

< 0, 38 > Unused.

< 0, 39 > Unused.

< 0, 40 > Number of images in bitmap.

< 0, 41 > Unused.

< 0, 42 > Unused.

< 0, 43 > Unused.

< 0, 44 > Unused.

< 0, 45 > Unused.

< 0, 46 > Unused.

< 0, 47 > Unused.

< 0, 48 > Unused.

< 0, 49 > Unused.

< 0, 50 > Unused.

< 0, 51 > Used in Lotus Notes connection to identify column to update. For structure see Appendix K.

VSCROLLBAR Control Type

< 0, 1 > The name of the Window control, unqualified by parent window name.

< 0, 2 > Unused

< 0, 3 > The type of the control, always set to VSCROLLBAR.

< 0, 4 > The parent of the control, that is, the window name.

< 0, 5 > The X location of the vScrollBar in pixels with 0 as the origin at the top left corner of the screen.

< 0, 6 > The Y location of the vScrollBar in pixels with 0 as the origin at the top left corner of the screen.

< 0, 7 > The width of the vScrollBar in pixels.

< 0, 8 > The depth of the vScrollBar in pixels.

< 0, 9 > Unused..

< 0, 10 > The SDK Style of the vScrollBar. For a discussion of SDK styles see Appendix A. For a list of SDK styles for ScrollBar controls see Appendix M.

The default vScrollBar style within OpenInsight is 0x50000001, which creates an vScrollBar with the following characteristics

```
SBS_Vert        0x1

WS_Visible 0x10000000

WS_Child        0x40000000
```

in other words, a visible vertical scroll bar which is a child of the parent window.

< 0, 11 > The Presentation Server (PS) style of the vScrollBar. There are settings that are not adequately catered for by the Windows SDK style setting that PS still needs to be able to set. This additional style bit allows these pieces of information to be sent to PS and thence to the WinAPI.

For details of the available PS SDK style bits see Appendix C.

< 0, 12 > The tab position for the vScrollBar within the window.

< 0, 13 > Background colour of the vScrollBar expressed as an RGB colour value. For default leave null.

< 0, 14 > Unused. For default set to 0.

< 0, 15 > Unused. For default set to MS Sans Serif¹-11¹700¹0¹0¹0¹0¹34¹0¹1¹2¹1¹0¹.

< 0, 16 > Unused.

< 0, 17 > Text mark delimited list of event handlers names for which a script has been defined. The event handler itself will be stored in SYSREPOSEVENTS with a key constructed as follows :-

```
AppName*EventName*WindowName.Control
```

e.g. SYSPROG*LOSTFOCUS*TEST.EDITLINE_1

< 0, 18 > Sub value delimited list of quick events . For quick event structure see Appendix D. Note that this array always has a trailing sub value mark.

< 0, 19 > Sub value delimited list of event handler names for which a quick event has been defined. Note that this array always has a trailing sub value mark.

< 0, 20 > Unused.

< 0, 21 > Unused.

< 0, 22 > Unused.

< 0, 23 > Table name that vScrollBar is associated with if this is a data aware control.

< 0, 24 > Column name that vScrollBar is associated with if this is a data aware control.

< 0, 25 > Unused. Set to 0.

< 0, 26 > Unused. Set to 0.

< 0, 27 > Unused.

< 0, 28 > Iconv. For normal value set to <<None>>. To default to dictionary conversion set to <<Default>>.

< 0, 29 > Oconv. For normal value set to <<None>>. To default to dictionary conversion set to <<Default>>.

< 0, 30 > Unused . For normal value set to <<None>>. To default to dictionary default set to <<Default>>.

< 0, 31 > Unused. Set to 0.

< 0, 32 > A sub-valued list of PS Styles. This is used during window design and changing it makes very little difference. For default set to 0x7fffe³0xd001e788.

< 0, 33 > Unused.

< 0, 34 > Unused.

< 0, 35 > Unused.

< 0, 36 > Unused.

< 0, 37 > A 4 byte word, with the first two bytes being the top range of the scroll bar and the final two bytes being the lower range of the scroll bar..

To interpret use the following code snippet

```
FourBytes = Oconv(Number, “MB”)

FourBytes = Oconv(FourBytes, “R(0)#32”)

Upper = Iconv(FourBytes[1, 16], “MB”)

Lower = Iconv(FourBytes[17,16], “MB”)
```

< 0, 38 > A 4 byte word, with the first two bytes being the numerator and the final two bytes being the denominator. The resultant fraction is how many increments to move when the scroll bar area is clicked in, rather than the scroll button being pushed.

< 0, 39 > Unused.

< 0, 40 > Unused

< 0, 41 > Unused.

< 0, 42 > Unused.

< 0, 43 > Unused.

< 0, 44 > Unused.

< 0, 45 > DDE Link type. One of four literals, either OFF, WARM, HOT or AUTO.

< 0, 46 > DDE Item identifier.

< 0, 47 > DDE Topic identifier.

< 0, 48 > DDE Service identifier.

< 0, 49 > Unused.

< 0, 50 > Unused.

< 0, 51 > Used in Lotus Notes connection to identify column to update. For structure see Appendix K.

HSCROLLBAR Control Type

< 0, 1 > The name of the Window control, unqualified by parent window name.

< 0, 2 > Unused

< 0, 3 > The type of the control, always set to HSCROLLBAR.

< 0, 4 > The parent of the control, that is, the window name.

< 0, 5 > The X location of the hScrollBar in pixels with 0 as the origin at the top left corner of the screen.

< 0, 6 > The Y location of the hScrollBar in pixels with 0 as the origin at the top left corner of the screen.

< 0, 7 > The width of the hScrollBar in pixels.

< 0, 8 > The depth of the hScrollBar in pixels.

< 0, 9 > Unused..

< 0, 10 > The SDK Style of the hScrollBar. For a discussion of SDK styles see Appendix A. For a list of SDK styles for ScrollBar controls see Appendix M.

The default hScrollBar style within OpenInsight is 0x50000000, which creates an hScrollBar with the following characteristics

```
SBS_Horz        0x0

WS_Visible 0x10000000

WS_Child        0x40000000
```

in other words, a visible horizontal scroll bar which is a child of the parent window.

< 0, 11 > The Presentation Server (PS) style of the hScrollBar. There are settings that are not adequately catered for by the Windows SDK style setting that PS still needs to be able to set. This additional style bit allows these pieces of information to be sent to PS and thence to the WinAPI.

For details of the available PS SDK style bits see Appendix C.

< 0, 12 > The tab position for the hScrollBar within the window.

< 0, 13 > Background colour of the hScrollBar expressed as an RGB colour value. For default leave null.

< 0, 14 > Unused. For default set to 0.

< 0, 15 > Unused. For default set to MS Sans Serif¹-11¹700¹0¹0¹0¹0¹34¹0¹1¹2¹1¹0¹.

< 0, 16 > Unused.

< 0, 17 > Text mark delimited list of event handlers names for which a script has been defined. The event handler itself will be stored in SYSREPOSEVENTS with a key constructed as follows :-

```
AppName*EventName*WindowName.Control
```

e.g. SYSPROG*LOSTFOCUS*TEST.EDITLINE_1

< 0, 18 > Sub value delimited list of quick events . For quick event structure see Appendix D. Note that this array always has a trailing sub value mark.

< 0, 19 > Sub value delimited list of event handler names for which a quick event has been defined. Note that this array always has a trailing sub value mark.

< 0, 20 > Unused.

< 0, 21 > Unused.

< 0, 22 > Unused.

< 0, 23 > Table name that hScrollBar is associated with if this is a data aware control.

< 0, 24 > Column name that hScrollBar is associated with if this is a data aware control.

< 0, 25 > Unused. Set to 0.

< 0, 26 > Unused. Set to 0.

< 0, 27 > Unused.

< 0, 28 > Iconv. For normal value set to <<None>>. To default to dictionary conversion set to <<Default>>.

< 0, 29 > Oconv. For normal value set to <<None>>. To default to dictionary conversion set to <<Default>>.

< 0, 30 > Unused . For normal value set to <<None>>. To default to dictionary default set to <<Default>>.

< 0, 31 > Unused. Set to 0.

< 0, 32 > A sub-valued list of PS Styles. This is used during window design and changing it makes very little difference. For default set to 0x7fffe³0xf001ef88.

< 0, 33 > Unused.

< 0, 34 > Unused.

< 0, 35 > Unused.

< 0, 36 > Unused.

< 0, 37 > A 4 byte word, with the first two bytes being the top range of the scroll bar and the final two bytes being the lower range of the scroll bar..

To interpret use the following code snippet

```
FourBytes = Oconv(Number, “MB”)

FourBytes = Oconv(FourBytes, “R(0)#32”)

Right     = Iconv(FourBytes[1, 16], “MB”)

Left      = Iconv(FourBytes[17,16], “MB”)
```

< 0, 38 > A 4 byte word, with the first two bytes being the denominator and the final two bytes being the divisor. The resultant fraction is how many increments to move when the scroll bar area is clicked in, rather than the scroll button being pushed.

< 0, 39 > Unused.

< 0, 40 > Unused

< 0, 41 > Unused.

< 0, 42 > Unused.

< 0, 43 > Unused.

< 0, 44 > Unused.

< 0, 45 > DDE Link type. One of four literals, either OFF, WARM, HOT or AUTO.

< 0, 46 > DDE Item identifier.

< 0, 47 > DDE Topic identifier.

< 0, 48 > DDE Service identifier.

< 0, 49 > Unused.

< 0, 50 > Unused.

< 0, 51 > Used in Lotus Notes connection to identify column to update. For structure see Appendix K..

RADIOBUTTON Control Type

< 0, 1 > The name of the Window control, unqualified by parent window name.

< 0, 2 > Unused

< 0, 3 > The type of the control. Always set to RADIOBUTTON.

< 0, 4 > The parent of the control, that is, the window name.

< 0, 5 > The X location of the radiobutton in pixels with 0 as the origin at the top left corner of the screen.

< 0, 6 > The Y location of the radiobutton in pixels with 0 as the origin at the top left corner of the screen.

< 0, 7 > The width of the radiobutton in pixels.

< 0, 8 > The depth of the radiobutton in pixels.

< 0, 9 > Unused.

< 0, 10 > The SDK Style of the radiobutton. For a discussion of SDK styles see Appendix A. For a list of SDK styles for PushButton controls see Appendix H.

The default radiobutton style within OpenInsight is 0x52000000, which creates a radiobutton with the following characteristics

```
WS_ClipChildren 0x2000000

WS_Visible      0x10000000

WS_Child              0x40000000
```

in other words, a visible control which is a child of the parent window.

< 0, 11 > The Presentation Server (PS) style of the radiobutton. By default this is set to 0x20. There are settings that are not adequately catered for by the Windows SDK style setting that PS still needs to be able to set. This additional style bit allows these pieces of information to be sent to PS and thence to the WinAPI.

For details of the available PS SDK style bits see Appendix C.

< 0, 12 > The tab position for the radiobutton within the window.

< 0, 13 > Background colour of the radiobutton expressed as an RGB colour value. For default leave null.

< 0, 14 > Foreground colour of the radiobutton expressed as an RGB colour value. For default set to 0.

< 0, 15 > Font of the radiobutton. Text mark delimited array having structure as detailed in Appendix F. For default set to MS Sans Serif¹-11¹700¹0¹0¹0¹0¹34¹0¹1¹2¹1¹0¹.

< 0, 16 > Unused

< 0, 17 > Text mark delimited list of event handlers names for which a script has been defined. The event handler itself will be stored in SYSREPOSEVENTS with a key constructed as follows :-

```
AppName*EventName*WindowName.Control
```

e.g. SYSPROG*CLICK*TEST.RADIOBUTTON_1

< 0, 18 > Sub value delimited list of quick events . For quick event structure see Appendix D. Note that this array always has a trailing sub value mark.

< 0, 19 > Sub value delimited list of event handler names for which a quick event has been defined. Note that this array always has a trailing sub value mark.

< 0, 20 > Unused.

< 0, 21 > Unused.

< 0, 22 > Unused.

< 0, 23 > Table name that radiobutton is associated with if this is a data aware control.

< 0, 24 > Column name that radiobutton is associated with if this is a data aware control.

< 0, 25 > Unused. Set to 0.

< 0, 26 > Unused. Set to 0.

< 0, 27 > Unused.

< 0, 28 > Unused. For normal value set to <<None>>.

< 0, 29 > Unused. For normal value set to <<None>>.

< 0, 30 > Unused. For normal value set to <<None>>.

< 0, 31 > Unused. Set to 0.

< 0, 32 > A sub-valued list of PS Styles. This is used during window design and changing it makes very little difference. For default radiobutton set to 0x7fffe³0xf001ef88.

< 0, 33 > Unused.

< 0, 34 > Unused.

< 0, 35 > Text mark delimited set of values for radio buttons.

< 0, 36 > Default value for radio button.

< 0, 37 > Unused.

< 0, 38 > Unused.

< 0, 39 > Unused.

< 0, 40 > Unused.

< 0, 41 > Unused.

< 0, 42 > Unused.

< 0, 43 > Unused.

< 0, 44 > Labels for radio buttons.

< 0, 45 > DDE Link type. One of four literals, either OFF, WARM, HOT or AUTO.

< 0, 46 > DDE Item identifier.

< 0, 47 > DDE Topic identifier.

< 0, 48 > DDE Service identifier.

< 0, 49 > Unused.

< 0, 50 > Sub value delimited list of sizes and positions for each of the radio button children. Each sub value corresponds to one button entry and is text mark delimited as follows :-

< 0, 0, 0, 1 > X pos relative to radio button in pixels.

< 0, 0, 0, 2> Y pos relative to radio button in pixels..

< 0, 0, 0, 3 > Width in pixels

< 0, 0, 0, 4 > Depth in pixels.

Note that there is always a trailing sub value mark.

< 0, 51 > Used in Lotus Notes connection to identify column to update. For structure see Appendix K.

RADIOBMP Control Type

< 0, 1 > The name of the Window control, unqualified by parent window name.

< 0, 2 > Unused

< 0, 3 > The type of the control. Always set to RADIOBMP.

< 0, 4 > The parent of the control, that is, the window name.

< 0, 5 > The X location of the radiokbmp in pixels with 0 as the origin at the top left corner of the screen.

< 0, 6 > The Y location of the radiobmp in pixels with 0 as the origin at the top left corner of the screen.

< 0, 7 > The width of the radiobmp in pixels.

< 0, 8 > The depth of the radiobmp in pixels.

< 0, 9 > Unused.

< 0, 10 > The SDK Style of the radiobmp. For a discussion of SDK styles see Appendix A. For a list of SDK styles for Button controls see Appendix H.

The default radiobmp style within OpenInsight is 0x52000000, which creates a radiobmp with the following characteristics

```
WS_ClipChildren 0x2000000
WS_Visible      0x10000000
WS_Child            0x40000000
```

in other words, a visible control which is a child of the parent window.

< 0, 11 > The Presentation Server (PS) style of the radiobmp. There are settings that are not adequately catered for by the Windows SDK style setting that PS still needs to be able to set. This additional style bit allows these pieces of information to be sent to PS and thence to the WinAPI.

For details of the available PS SDK style bits see Appendix C.

< 0, 12 > The tab position for the radiobmp within the window.

< 0, 13 > Unused.

< 0, 14 > Unused.

< 0, 15 > Unused.

< 0, 16 > Unused.

< 0, 17 > Text mark delimited list of event handlers names for which a script has been defined. The event handler itself will be stored in SYSREPOSEVENTS with a key constructed as follows :-

```
AppName*EventName*WindowName.Control
```

e.g. SYSPROG*CLICK*TEST.RADIOBMP_1

< 0, 18 > Sub value delimited list of quick events . For quick event structure see Appendix D. Note that this array always has a trailing sub value mark.

< 0, 19 > Sub value delimited list of event handler names for which a quick event has been defined. Note that this array always has a trailing sub value mark.

< 0, 20 > Unused.

< 0, 21 > Unused.

< 0, 22 > Unused.

< 0, 23 > Table name that checkbmp is associated with if this is a data aware control.

< 0, 24 > Column name that checkbmp is associated with if this is a data aware control.

< 0, 25 > Unused. Set to 0.

< 0, 26 > Unused. Set to 0.

< 0, 27 > Unused.

< 0, 28 > Unused. For normal value set to <<None>>.

< 0, 29 > Unused. For normal value set to <<None>>.

< 0, 30 > Unused. For normal value set to <<None>>.

< 0, 31 > Unused. Set to 0.

< 0, 32 > A sub-valued list of PS Styles. This is used during window design and changing it makes very little difference. For default Radiobmp set to 0x7ffee³0xf0016080.

< 0, 33 > Bitmap name. Fully qualified entity identifier. E.G. SYSPROG*IMAGE*BMP*OR_NEW.

< 0, 34 > Unused.

<< 0, 35 > Text mark delimited set of values for radio buttons.

< 0, 36 > Default value for radio button.

< 0, 37 > Unused.

< 0, 38 > Unused.

< 0, 39 > Unused.

< 0, 40 > Number of images in bitmap.

< 0, 41 > Unused.

< 0, 42 > Unused.

< 0, 43 > Unused

< 0, 44 > Unused.

< 0, 45 > DDE Link type. One of four literals, either OFF, WARM, HOT or AUTO.

< 0, 46 > DDE Item identifier.

< 0, 47 > DDE Topic identifier.

< 0, 48 > DDE Service identifier.

< 0, 49 > Unused.

< 0, 50 > Sub value delimited list of sizes and positions for each of the radio button children. Each sub value corresponds to one button entry and is text mark delimited as follows :-

< 0, 0, 0, 1 > X pos relative to radio button in pixels.

< 0, 0, 0, 2> Y pos relative to radio button in pixels..

< 0, 0, 0, 3 > Width in pixels

< 0, 0, 0, 4 > Depth in pixels.

Note that there is always a trailing sub value mark.

< 0, 51 > Used in Lotus Notes connection to identify column to update. For structure see Appendix K.

Chapter 5: Menu Information

That which we refer to as “Menu information” is stored in the fourth record mark delimited sub-string of the window row in SYSREPOSWINS. It is made up of two field mark delimited sub strings with the first containing the *menu appearance information* and with the second containing the *menu event information*. Each of these substrings is in turn made up of further delimited sub strings.

< 1 > Menu appearance information

< 2 > Menu event information

Menu Appearance Information

Within the first field mark delimited sub string, there are multiple value mark delimited sub strings, describing either the Menu itself (of which there is only one), a Popup (one or more describing each menu popup which contains sub options), an Item (the actual part of the menu that does the work) or a Separator (a purely aesthetic device). If there is no menu the first field mark delimited sub string is null. Note that the term “Popup” in the context of a menu has no relation to the popups designed in the User Interface workspace.

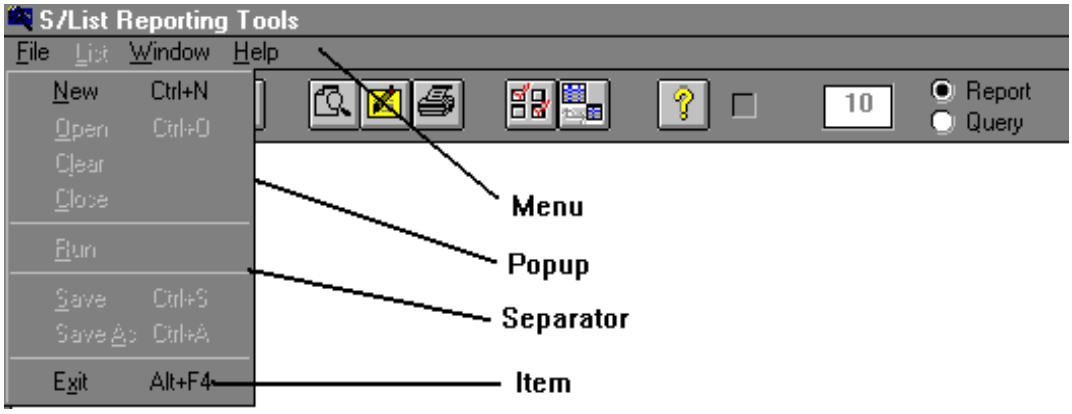

Figure 3 - Menu components

The normal make up of a menu could be represented as follows

```
MENU
        POPUP1
                ITEM1
                ITEM2
                SEPARATOR
                ITEM3
        POPUP2
                ITEM1
                ITEM2
                ITEM3
                SEPARATOR
                ITEM4
                ITEM5
```


MENU Component Structure

< 0, 0, 1 >	The literal “MENU”
< 0, 0, 2 >	Count of how many POPUPS there are in the menu.
< 0, 0, 3 >	Null
< 0, 0, 4 >	Null
< 0, 0, 5 >	0
< 0, 0, 6 >	0
< 0, 0, 7 >	0
< 0, 0, 8 >	0
< 0, 0, 9 >	Total number of popups, items and separators in Menu, including the Menu itself.
< 0, 0, 10 >	Null
< 0, 0, 11 >	0

POPUP Component Structure

< 0, 0, 1 >	The literal “POPUP”
< 0, 0, 2 >	Count of items and separators in popup not including items and separators which are children of popups from this popup.
< 0, 0, 3 >	The name of the popup to use when getting and setting property. If the default name is to be used this is normally left blank.
< 0, 0, 4 >	String literal for popup text
< 0, 0, 5 >	Disabled flag (1 true, 0 false)
< 0, 0, 6 >	0
< 0, 0, 7 >	Hidden flag (1 true, 0 false)
< 0, 0, 8 >	Null
< 0, 0, 9 >	Line of help text for popup
< 0, 0, 10 >	Null
< 0, 0, 11 >	Null

ITEM Component Structure

< 0, 0, 1 > The literal “ITEM”

< 0, 0, 2 > Null

< 0, 0, 3 > The name of the item to use when getting and setting property. If the default name is to be used this is normally left blank

< 0, 0, 4 > String literal for item text

< 0, 0, 5 > Disabled flag (1 true, 0 false)

< 0, 0, 6 > Checked flag (1 true, 0 false)

< 0, 0, 7 > Hidden flag (1 true, 0 false)

< 0, 0, 8 > ASCII representation of the virtual code for the accelerator key for this menu option. If < 255 then this can just be interpreted as the virtual code for a key (See Appendix N). If > 255 then this is a two byte representation with the first byte being the key-state and the second being the key itself.

In this representation, Shift is 0100h, Ctrl is 0200h and Alt is 0400h, in ASCII, 256, 512 and 1024 respectively.

Thus a value of 801 would be 0321h. Referring to Appendix A we see that 21 is Page Up, and 03 is made up of 01h (shift) plus 02h (ctrl). Thus the accelerator key would be Ctrl-Shift-PageUp.

< 0, 0, 9 > Line of help text for item

< 0, 0, 10 > Null

< 0, 0, 11 > Integer flag - made up of combination of

1 Auto-checked

2 Beginning of group

4 End of group

16 Don’t generate event

32 Generate LostFocus event

64 Pass event to MDI frame

128 Use MDI Frame properties

SEPARATOR Component Structure

< 0, 0, 1 >	The literal “SEPARATOR”
< 0, 0, 2 >	Null
< 0, 0, 3 >	Null
< 0, 0, 4 >	A unique separator key for this menu item of form SEPnnn
< 0, 0, 5 >	0
< 0, 0, 6 >	0
< 0, 0, 7 >	Hidden flag (1 true, 0 false)
< 0, 0, 8 >	0
< 0, 0, 9 >	Null
<0, 0, 10 >	Null
< 0, 0, 11 >	0

Menu Event Information

The menu event information is a value mark delimited list having four elements as follows :-

< 0, 1 > Event handler scripts. A text mark delimited list of the names of the event handlers scripts for the menu item. Note that these are stored in SYSREPOSEVENTS with an id of format

Account*MENU*WindowName.MENU.Name

where name is the event name given here.

< 0, 2 > Quick event definitions. One quickevent definition per sub-value, corresponding to the control names in < 0, 3 >. The definition is text mark delimited with a structure as detailed in Appendix D.

< 0, 3 > Menu control names corresponding to the quickevents in <0, 2>.

< 0, 4 > 0 - very infrequently 1 but at time of going to press unsure of why.

Glossary of Terms

API	Application Programming Interface. A published set of documentation which permits developers to access another program by calls to a standard interface.
Field Mark	@FM - An ASCII 254 - the field delimiter in OpenInsight
Presentation Server (PS)	That part of OINSIGHT.EXE which actually deals with translating painted forms into Windows objects and deals with handling messages returned back from the Windows API.
Record Mark	@RM - An ASCII 255 - the record delimiter in OpenInsight
REVBOOT	The directory from which OpenInsight is launched.
RGB colour value	A large integer which Windows uses to derive a colour. It is created by taking the values for Red Green and Blue, between 0 and 255 and factoring them to create the number as follows `RGB = R + (G * 256) + (B * 256 * 256)`
SDK	Software Development Kit. A resource intended to document all of the many routines available for accessing within an API. Includes definitions of the constants and variables required as well as documentation on what each API entry point does.
Sub sub text mark	An ASCII 249 - the sub sub text delimiter in OpenInsight
Sub Text Mark	@STM - An ASCII 250 - the sub text delimiter in OpenInsight.
Sub Value Mark	@SVM - An ASCII 252 - the sub value delimiter in OpenInsight
Text Mark	@TM - An ASCII 251 - the text delimiter in OpenInsight
Value Mark	@VM - An ASCII 253 - the multi-value delimiter in OpenInsight
WinAPI	The Windows API.

Appendix A - SDK Styles

Within each Windows control type there are so many parameters that can be set that it becomes impractical to set them all individually from calls to the API. Imagine trying to describe the average Window set-up … “OK, WinAPI, I want you to set me up a Window and give me the handle - let me know when you’re done”. “OK, you’ve done that? Now set the Window style to WS_OVERLAPPEDWINDOW and let me know when you’re done”. “OK, you’ve done that? Now set the Window style to WS_VISIBLE and let me know when you’re done”. “OK, you’ve done that? Now set the Window style to WS_POPUP and let me know when you’re done”. “OK, you’ve done that? Now set the Window style to WS_CLIPCHILDREN and let me know when you’re done”. …

This would make development such a chore that no-one would bother!

To get around this Microsoft implemented the concept of “Styles”. A style is a way of condensing all of the above information into a single value. It essentially assigns each different characteristic of a control to a different integer value, such that no two integer values can be added together to give any other. Thus in the above example, WC_OVERLAPPEDWINDOW could be assigned 1, WS_VISIBLE 2, WS_POPUP 4 and WS_CLIPCHILDREN 8. The WinAPI could therefore be told to create the window as a 15 type (1 + 2 + 4 + 8) and it could get on and do it quickly.

The technically minded amongst you will quickly realise that these values are better represented as binary as this makes bit masking possible, thus in the above example

WS_OVERLAPPEDWINDOW	0001
WS_VISIBLE	0010
WS_POPUP	0100
WS_CLIPCHILDREN	1000
and all four	1111

Binary is more conveniently represented as Hex, and thus style values are generally expressed as hexadecimal numbers.

Appendix B - Window SDK Styles

0x0000	WS_Overlapped.	Creates an overlapped window. An overlapped window has a caption and a border.
0x10000	WS_MaximizeBox	Creates a window that has a Maximize box.
0x20000	WS_MinimizeBox	Creates a window that has a Minimize box.
0x40000	WS_ThickFrame	Creates a window with a thick frame that can be used to size the window.
0x80000	WS_SysMenu.	Creates a window that has a System-menu box in its title bar. Used only for windows with title bars. If used with a child window, this style creates a Close box instead of a System-menu box.
0x100000	WS_Hscroll.	Creates a window that has a horizontal scroll bar.
0x200000	WS_Vscroll.	Creates a window that has a vertical scroll bar.
0x400000	WS_DlgFrame.	Creates a window with a modal dialog box frame but no title.
0x800000	WS_Border.	Creates a window that has a border.
0xC00000	WS_Caption.	Creates a window that has a title bar (implies the WS_BORDER style).
0x1000000	WS_Maximize.	Creates a window of maximum size.
0x2000000	WS_ClipChildren.	Excludes the area occupied by child windows when drawing within the parent window. Used when creating the parent window.
0x4000000	WS_ClipSiblings.	Clips child windows relative to each other; that is, when a particular child

		window receives a WM_PAINT message, this style clips all other top-level child windows out of the region of the child window to be updated. (If the WS_CLIPSIBLINGS style is not given and child windows overlap, it is possible, when drawing in the client area of a child window, to draw in the client area of a neighbouring child window.) For use with the WS_CHILD style only.
0x8000000	WS_Disabled.	Creates a window that is initially disabled
0x10000000	WS_Visible.	Creates a window that is initially visible. This applies to overlapping and pop-up windows
0x20000000	WS_Minimize.	Creates a window of minimum size..
0x40000000	WS_Child.	Creates a child window. It cannot be used with the WS_Popup style.
0x80000000	WS_Popup.	Creates a pop-up window. It cannot be used with the WS_Child style

Appendix C - PS Styles

Note that PS Styles may mean different things under different circumstances.

0x0	PS_Normal	RESIZE for bitmaps.
0x1	PS_Default	Indicates that this is the default pushbutton.
0x2	PS_Cancel	Indicates that this is the cancel pushbutton.
0x4	PS_First	Indicates that this is the control which PS ought to give focus to when the window starts.
0x10	PS_Win_BBr	Indicates that the button bar is enabled for this window
0x10	PS_Horiz	Indicates that the control should be displayed horizontally rather than vertically.
0x20	PS_Win_Tile	Indicates that the bitmap on the window should be tiled.
0x20	PS_Vert	Indicates that the control should be displayed vertically rather than horizontally.
0x80	PS_NoBubbleHelp	Disables the display of bubble help on buttons.
0x100	PS_Clip	Indicates that the bitmap should be clipped. (Default state for windows, regardless of whether a bitmap is attached or not - an anomaly caused by the fact that the radio button is set by default in the form painter).
0x100	PS_RTF	Indicate that the editbox is a rich text format control.

0x100	PS_Hierarchical	Indicates that the listbox is hierarchical.
0x100	PS_KeepScrollBars	Indicates that a control should always display scroll bars even when the amount of data in the control does not warrant it.
0x200	PS_AutoIconArrange	If set, an MDI frame will not allow the user to select the position of the iconised child windows. It will insist on automatically arranging them. Note, the icons are not fixed in place, rather as the user moves them, the system restores them to their original location.
0x400	PS_Combine	Combine text and bitmap on a control
0x400	PS_Overlap	Indicates overlapping tabs on a listbox.
0x800	PS_IgnoreAccelerator	If this style bit is set, accelerator keys defined for controls (NB controls *not* menus) will be ignored.
0x1000	PS_DialogBox	Indicates that the window is a dialog box.
0x1000	PS_RightAnchor	Anchor the control to the right.
0x2000	PS_Pages	Indicates that this is a multi-page window
0x2000	PS_BottomAnchor	Anchor the control to the bottom
0x4000	PS_MultiInstance	Allow multiple instances of the window in an MDI frame.
0x4000	PS_AutoSizeWidth	Automatically size the width of the control.
0x8000	PS_No3D	Indicates that 3D controls ought not to be enabled for this window.
0x8000	PS_AutoSizeDepth	Automatically size the depth of the control.

0x1000000	PS_Enter	Indicates that Enter should be treated as a double click on a listbox
0x4000000	PS_InitCollapsed	Indicates that a hierarchical listbox ought to start off collapsed rather than expanded.

Appendix D - Quick Event Structures

The quick event is a text mark delimited array containing six values structured as follows :-

< 0, 0, 0, 1 >	The literal “E” if the quick event is sending an event to a control (and thus using Send_Event) or the literal “R” if a message is being sent to an entity.
< 0, 0, 0, 2 >	The event (if <0, 0, 0, 1> is “E”) or message (if < 0, 0, 0, 1 > is “R”) to send.
< 0, 0, 0, 3 >	The control (if <0, 0, 0, 1> is “E”) or entity (if < 0, 0, 0, 1 > is “R”) to send to.
< 0, 0, 0, 4 >	The parameters to pass. These will obviously differ depending on the event/message being sent, but the most common parameters are detailed below.
< 0, 0, 0, 5 >	The name of the control to return the value in.
< 0, 0, 0, 6 >	The property of the destination control to update with the returned value.

The parameters in text value 4 are sub text value delimited. The most common parameters for quick events are documented below. For all others examine the quick event builder in form designer.

Start a window	< 0, 0, 0, 0, 1 >	Create Parameter for the window
	< 0, 0, 0, 0, 2 >	Parent window name (this window)
Execute a popup	< 0, 0, 0, 0, 1 >	Parent window name (this window)
Execute a procedure		Whatever the procedure requires
Display quick help	< 0, 0, 0, 0, 1 >	Parent window name (this window)
Display a message	< 0, 0, 0, 0, 1 >	Parent window name (this window)
Index lookup	< 0, 0, 0, 0, 1 >	The table name to look up on
	< 0, 0, 0, 0, 2 >	A sub sub text mark delimited list of columns to look up on

< 0, 0, 0, 0, 3 >	A sub sub text mark delimited list of columns to display in the resulting popup.
< 0, 0, 0, 0, 4 >	If null, single selection popup. If the literal “MULTI” then a multi-selection popup.

Appendix E - Static Style Settings

0x0	SS_Left	Designates a simple rectangle and displays the given text left-aligned in the rectangle. The text is formatted before it is displayed. Words that would extend past the end of a line are automatically wrapped to the beginning of the next left-aligned line.
0x1	SS_Centre	Designates a simple rectangle and displays the given text centred in the rectangle. The text is formatted before it is displayed. Words that would extend past the end of a line are automatically wrapped to the beginning of the next centred line.
0x2	SS_Right	Designates a simple rectangle and displays the given text right-aligned in the rectangle. The text is formatted before it is displayed. Words that would extend past the end of a line are automatically wrapped to the beginning of the next right-aligned line.
0x3	SS_Icon	Designates an icon displayed in the dialog box. The given text is the name of an icon (not a filename) defined elsewhere in the resource file. The nWidth and nHeight parameters are ignored; the icon automatically sizes itself.
0x4	SS_BlackRect	Specifies a rectangle filled with the colour used to draw window frames. This colour is black in the default Windows colour scheme.
0x5	SS_GrayRect	Specifies a rectangle filled with the colour used to fill the screen background. This colour is grey in the default Windows colour scheme.

0x6	SS_WhiteRect	Specifies a rectangle filled with the colour used to fill window backgrounds. This colour is white in the default Windows colour scheme.
0x7	SS_BlackFrame	Specifies a box with a frame drawn in the same colour as window frames. This colour is black in the default Windows colour scheme.
0x8	SS_GrayFrame	Specifies a box with a frame drawn with the same colour as the screen background (desktop). This colour is grey in the default Windows colour scheme.
0x9	SS_WhiteFrame	Specifies a box with a frame drawn in the same colour as window backgrounds. This colour is white in the default Windows colour scheme.
0xB	SS_Simple	Designates a simple rectangle and displays a single line of text left-aligned in the rectangle. The line of text cannot be shortened or altered in any way.
0xC	SS_LeftNoWord Wrap	Designates a simple rectangle and displays the given text left-aligned in the rectangle. Tabs are expanded but words are not wrapped. Text that extends past the end of a line is clipped.
0x80	SS_NoPrefix	Prevents interpretation of any & characters in the control's text as accelerator prefix characters (which are displayed with the & removed and the next character in the string underlined). This static control style may be included with any of the defined static controls.

Appendix F - Font Structure

< 1, 1, 1, 1 > FaceName

< 1, 1, 1, 2 > Height. Specifies the height of character cells.

< 1, 1, 1, 3 > Weight. Specifies the weight of the font. This member can be one of the following values:

Constant	Value
FW_DONTCARE	0
FW_THIN	100
FW_EXTRALIGHT	200
FW_ULTRALIGHT	200
FW_LIGHT	300
FW_NORMAL	400
FW_REGULAR	400
FW_MEDIUM	500
FW_SEMIBOLD	600
FW_DEMIBOLD	600
FW_BOLD	700
FW_EXTRABOLD	800
FW_ULTRABOLD	800
FW_BLACK	900
FW_HEAVY	900

< 1, 1, 1, 4> Italic. Specifies an italic font if it is non-zero.

< 1, 1, 1, 5> Underline. Specifies an underlined font if it is non-zero.

< 1, 1, 1, 6> Width. Specifies the average width of characters in the font. For ANSI_CHARSET fonts, this is a weighted average of the characters "a" through "z" and the space character. For other character sets, this value is an unweighted average of all characters in the font.

< 1, 1, 1, 7> CharSet. Specifies the character set of the font. The following values are defined:

Constant	Value
ANSI_CHARSET	0
DEFAULT_CHARSET	1
SYMBOL_CHARSET	2
SHIFTJIS_CHARSET	128
OEM_CHARSET	255

< 1, 1, 1, 8> PitchAndFamily Specifies the pitch (angle) and family (for example, Roman or Swiss) of the selected font.

< 1, 1, 1, 9> StrikeOut. Specifies a "struckout" font, if it is non-zero. (A horizontal line is drawn through the middle of the text.)

< 1, 1, 1, 10> OutPrecision

< 1, 1, 1, 11> ClipPrecision

< 1, 1, 1, 12> Quality

Appendix G - Edit Control Style Settings

0x0	ES_Left	Left aligns text.
0x1	ES_Center	Centres text in a multiline edit control.
0x2	ES_Right	Right aligns text in a multiline edit control.
0x4	ES_Multiline	Designates a multiline edit control. (The default is single-line edit control.) When the multiline edit control is in a dialog box, the default response to pressing the ENTER key is to activate the default button. To use the ENTER key as a carriage return, an application should use the ES_WANTRETURN style. When the multiline edit control is not in a dialog box and the ES_AUTOVSCROLL style is specified, the edit control shows as many lines as possible and scrolls vertically when the user presses the ENTER key. If ES_AUTOVSCROLL is not specified, the edit control shows as many lines as possible and beeps if the user presses ENTER when no more lines can be displayed. If the ES_AUTOHSCROLL style is specified, the multiline edit control automatically scrolls horizontally when the caret goes past the right edge of the control. To start a new line, the user must press ENTER. If ES_AUTOHSCROLL is not specified, the control automatically wraps words to the beginning of the next line when necessary. A new line is also started if the user presses ENTER. The position of the wordwrap is determined by the window size. If the window size

		changes, the wordwrap position changes and the text is redisplayed. Multiline edit controls can have scroll bars. An edit control with scroll bars processes its own scroll bar messages. Edit controls without scroll bars scroll as described in the previous two paragraphs and process any scroll messages sent by the parent window.
0x8	ES_Uppercase	Converts all characters to uppercase as they are typed into the edit control.
0x10	ES_Lowercase	Converts all characters to lowercase as they are typed into the edit control.
0x20	ES_Password	Displays all characters as an asterisk (*) as they are typed into the edit control. An application can use the EM_SETPASSWORDCHAR message to change the character that is displayed.
0x40	ES_AutoVScroll	Automatically scrolls text up one page when the user presses ENTER on the last line.
0x80	ES_AutoHScroll	Automatically scrolls text to the right by 10 characters when the user types a character at the end of the line. When the user presses the ENTER key, the control scrolls all text back to position zero.
0x100	ES_NoHideSel	Negates the default behaviour for an edit control. The default behaviour is to hide the selection when the control loses the input focus and invert the selection when the control receives the input focus.
0x400	ES_OemConvert	Converts text entered in the edit control from the Windows character set to the OEM character set and then back to the Windows set. This ensures proper character conversion when the application calls the AnsiToOem function to convert a Windows string in the edit control to OEM characters. This style is most useful for edit controls that contain filenames.

0x800	ES_ReadOnly	Prevents the user from typing or editing text in the edit control.
0x1000	ES_WantReturn	Specifies that a carriage return be inserted when the user presses the ENTER key while entering text into a multiline edit control in a dialog box. If this style is not specified, pressing the ENTER key has the same effect as pressing the dialog box's default push button. This style has no effect on a single-line edit control.

Appendix H - Button Control Style Settings

0x0	BS_PushButton	Creates a push button that posts a WM_COMMAND message to the owner window when the user selects the button.
0x1	BS_DefPushButton	Creates a button that has a heavy black border. The user can select this button by pressing the ENTER key. This style is useful for enabling the user to quickly select the most likely option (the default option).
0x2	BS_CheckBox	Creates a small square that has text displayed to its right (unless this style is combined with the BS_LEFTTEXT style).
0x3	BS_AutoCheckBox	Creates a button that is the same as a check box, except that an X appears in the check box when the user selects the box; the X disappears (is cleared) the next time the user selects the box.
0x4	BS_RadioButton	Creates a small circle that has text displayed to its right (unless this style is combined with the BS_LEFTTEXT style). Radio buttons are usually used in groups of related but mutually exclusive choices.
0x5	BS_3State	Creates a button that is the same as a check box, except that the box can be greyed (dimmed) as well as checked. The greyed state is used to show that the state of the check box is not determined.
0x6	BS_Auto3State	Creates a button that is the same as a three-state check box, except that the box changes its state when the user selects it. The state cycles through checked, greyed, and normal.
0x7	BS_GroupBox	Creates a rectangle in which other controls can be grouped. Any text

		associated with this style is displayed in the rectangle's upper-left corner.
0x9	BS_AutoRadio Button	Creates a button that is the same as a radio button, except that when the user selects it, the button automatically highlights itself and clears (removes the selection from) any other buttons in the same group.
0xA	BS_PushBox	creates a push-box control, which is identical to a pushbutton, except that it does not display a button face or frame; only the text appears..
0xB	BS_OwnerDraw	Creates an owner-drawn button. The owner window receives a WM_MEASUREITEM message when the button is created, and it receives a WM_DRAWITEM message when a visual aspect of the button has changed. The BS_OWNERDRAW style cannot be combined with any other button styles.
0x20	BS_LeftText	Places text on the left side of the radio button or check box when combined with a radio button or check box style.

Appendix I - ComboBox Control Style Settings

0x1	CBS_Simple	Displays the list box at all times. The current selection in the list box is displayed in the edit control.
0x2	CBS_DropDown	Similar to CBS_SIMPLE, except that the list box is not displayed unless the user selects an icon next to the edit control.
0x3	CBS_DropDownList	Similar to CBS_DROPDOWN, except that the edit control is replaced by a static text item that displays the current selection in the list box.
0x10	CBS_OwnerDrawFixed	Specifies that the owner of the list box is responsible for drawing its contents and that the items in the list box are all the same height. The owner window receives a WM_MEASUREITEM message when the combo box is created and a WM_DRAWITEM message when a visual aspect of the combo box has changed.
0x20	CBS_OwnerDrawVariable	Specifies that the owner of the list box is responsible for drawing its contents and that the items in the list box are variable in height. The owner window receives a WM_MEASUREITEM message for each item in the combo box when the combo box is created and a WM_DRAWITEM message whenever the visual aspect of the combo box changes.
0x40	CBS_AutoHScroll	Automatically scrolls the text in the edit control to the right when

		the user types a character at the end of the line. If this style is not set, only text that fits within the rectangular boundary is allowed.
0x80	CBS_OEMConvert	Converts text entered in the combo-box edit control from the Windows character set to the OEM character set and then back to the Windows set. This ensures proper character conversion when the application calls the AnsiToOem function to convert a Windows string in the combo box to OEM characters. This style is most useful for combo boxes that contain filenames and applies only to combo boxes created with the CBS_SIMPLE or CBS_DROPDOWN styles.
0x100	CBS_Sort	Automatically sorts strings entered into the list box.
0x200	CBS_HasStrings	Specifies that an owner-drawn combo box contains items consisting of strings. The combo box maintains the memory and pointers for the strings so the application can use the CB_GETLBTEXT message to retrieve the text for a particular item.
0x400	CBS_NoIntegralHeight	Specifies that the size of the combo box is exactly the size specified by the application when it created the combo box. Normally, Windows sizes a combo box so that the combo box does not display partial items.
0x800	CBS_DisableNoScroll	Shows a disabled vertical scroll bar in the list box when the box does not contain enough items to scroll. Without this style, the scroll bar is hidden when the list box does not contain enough items.

Appendix J - ListBox Control Style Settings

0x1	LBS_Notify	Notifies the parent window with an input message whenever the user clicks or double-clicks a string.
0x2	LBS_Sort	Sorts strings in the list box alphabetically.
0x4	LBS_NoRedraw	Specifies that the list box's appearance is not updated when changes are made. This style can be changed at any time by sending a WM_SETREDRAW message.
0x8	LBS_MultipleSel	Turns string selection on or off each time the user clicks or double-clicks the string. Any number of strings can be selected.
0x10	LBS_OwnerDrawFixed	Specifies that the owner of the list box is responsible for drawing its contents and that the items in the list box are the same height. The owner window receives a WM_MEASUREITEM message when the list box is created and a WM_DRAWITEM message when a visual aspect of the list box has changed.
0x20	LBS_OwnerDrawVariable	Specifies that the owner of the list box is responsible for drawing its contents and that the items in the list box are variable in height. The owner window receives a WM_MEASUREITEM message for each item in the combo box

		when the combo box is created and a WM_DRAWITEM message whenever the visual aspect of the combo box changes.
0x40	LBS_HasStrings	Specifies that a list box contains items consisting of strings. The list box maintains the memory and pointers for the strings so the application can use the LB_GETTEXT message to retrieve the text for a particular item. By default, all list boxes except owner-drawn list boxes have this style. An application can create an owner-drawn list box either with or without this style.
0x80	LBS_UseTabStops	Allows a list box to recognize and expand tab characters when drawing its strings. The default tab positions are 32 dialog box units. (A dialog box unit is a horizontal or vertical distance. One horizontal dialog box unit is equal to one-fourth of the current dialog box base width unit. The dialog box base units are computed based on the height and width of the current system font. The GetDialogBaseUnits function returns the current dialog box base units in pixels.)
0x100	LBS_NoIntegralHeight	Specifies that the size of the list box is exactly the size specified by the application when it created the list box. Normally, Windows sizes a list box so that the list box does not display partial items.
0x200	LBS_MultiColumn	Specifies a multicolumn list box that is scrolled horizontally. The LB_SETCOLUMNWIDTH message sets the width of the columns.

0x400	LBS_WantKeyBoardInput	Specifies that the owner of the list box receives WM_VKEYTOITEM or WM_CHARTOITEM messages whenever the user presses a key and the list box has the input focus. This allows an application to perform special processing on the keyboard input. If a list box has the LBS_HASSTRINGS style, the list box can receive WM_VKEYTOITEM messages but not WM_CHARTOITEM messages. If a list box does not have the LBS_HASSTRINGS style, the list box can receive WM_CHARTOITEM messages but not WM_VKEYTOITEM messages.
0x800	LBS_ExtendedSel	Allows multiple items to be selected by using the SHIFT key and the mouse or special key combinations.
0x1000	LBS_DisableNoScroll	Shows a disabled vertical scroll bar for the list box when the box does not contain enough items to scroll. If this style is not specified, the scroll bar is hidden when the list box does not contain enough items.

Appendix K - Notes Column Link Structure

The column link structure is a sub valued dynamic array. Its structure is polymorphic, representing both Notes Forms and Views.

Form Structure

< 0, 0, 1 > The fully qualified entity identifier for the Notes Form e.g. EXAMPLES*DBCOMPONENT*NOTESFORM*RESP

< 0, 0, 2 > The column name in the Notes Form

< 0, 0, 3 > The datatype, being either Number, NumberList, RichText, Text, TextList, TimeDate, TimeDateList, UserId.

< 0, 0, 4 > Unused. Always set to 0.

View Structure

< 0, 0, 1 > The fully qualified entity identifier for the Notes View e.g. EXAMPLES*DBCOMPONENT*NOTESVIEW*MAIL

< 0, 0, 2 > Text mark delimited list of columns to display.

< 0 ,0, 3 > Text mark delimited list of formulas.

< 0, 0, 4 > Text mark delimited list of unknowns.

< 0, 0, 5 > Text mark delimited list of column widths.

Appendix L - EditTable Control Style Settings

Note that there are two types of sub-control within the EditTable, the Table itself and all of the columns. Each type has its own styles.

Table Styles

0x8	DTS_Resize	Whether users are allowed to resize columns.
0x40	DTS_Hgrid	Whether horizontal grid lines should be displayed
0x80	DTS_Vgrid	Whether vertical grid lines should be displayed.
0x100	DTS_RowSelect	Whether rows or tuples (column/row intersections) are selected.
0x200	DTS_MultiRow	Whether multiple rows may be selected (as in a multi-choice popup).
0x400	DTS_ColSelect	Permit selection of entire column.
0x1000	DTS_LargeData	Whether data ought to be stored in Virtual Memory and thus be allowed to be greater than 64K. Equivalent to setting “No Rows” to -1.
0x2000	DTS_RowButtons	Whether rows ought to display buttons down the left of the table.
0x4000	DTS_RowNumbers	Whether the buttons displayed by DTS_RowButtons ought to have numbers.

Column Styles

0x1	DTCS_Resize	Whether this column may be resized by the user.
0x2	DTCS_Fixed	
0x4	DTCS_Edit	

0x8	DTCS_Protect	Whether the column should be protected from amendment.
0x20	DTCS_Hidden	Whether the column is present but hidden from user interaction.
0x40	DTCS_Centre	Whether the data is centred.
0x80	DTCS_Right	Whether the data is right justified.
0x100	DTCS_HeadCentre	Whether the heading is centred.
0x200	DTCS_HeadRight	Whether the heading is right justified.
0x2000	DTCS_Locked	Whether the column is locked (to prevent scrolling).
0x4000	DTCS_SortAsc	Whether the column should be sorted in ascending left justified order.
0x800	DTCS_SortDes	Whether the column should be sorted in descending left justified order.

Appendix M - ScrollBar Control Style Settings

0x0	SBS_Horz	Designates a horizontal scroll bar. If neither the SBS_BOTTOMALIGN nor SBS_TOPALIGN style is specified, the scroll bar has the height, width, and position specified by the CreateWindow parameters.
0x1	SBS_Vert	Designates a vertical scroll bar. If neither the SBS_RIGHTALIGN nor SBS_LEFTALIGN style is specified, the scroll bar has the height, width, and position specified by the CreateWindow parameters.
0x2	SBS_TopAlign	Aligns the top edge of the scroll bar with the top edge of the rectangle defined by the CreateWindow parameters. The scroll bar has the default height for system scroll bars. Used with the SBS_HORZ style.
0x2	SBS_LeftAlign	Aligns the left edge of the scroll bar with the left edge of the rectangle defined by the CreateWindow parameters. The scroll bar has the default width for system scroll bars. Used with the SBS_VERT style.
0x4	SBS_BottomAlign	Aligns the bottom edge of the scroll bar with the bottom edge of the rectangle defined by the following CreateWindow parameters: x, y, nWidth, and nHeight. The scroll bar has the default height for system scroll bars. Used with the SBS_HORZ style.
0x4	SBS_RightAlign	Aligns the right edge of the scroll bar with the right edge of the rectangle defined by the CreateWindow parameters. The scroll bar has the default width for system scroll bars. Used with the SBS_VERT style.

0x2	SBS_SizeBoxTopLeftAlign	Aligns the upper-left corner of the size box with the upper-left corner of the rectangle specified by the following CreateWindow parameters: x, y, nWidth, and nHeight. The size box has the default size for system size boxes. Used with the SBS_SIZEBOX style.
0x4	SBS_SizeBoxBottonRightAlign	Aligns the lower-right corner of the size box with the lower-right corner of the rectangle specified by the CreateWindow parameters. The size box has the default size for system size boxes. Used with the SBS_SIZEBOX style.
0x8	SBS_SizeBox	Designates a size box. If neither the SBS_SIZEBOXBOTTOMRIGHTALIGN nor SBS_SIZEBOXTOPLEFTALIGN style is specified, the size box has the height, width, and position specified by the CreateWindow parameters.

Appendix N - Virtual Key Codes

01	Left mouse button
02	Right mouse button
03	Used for control-break processing
04	Middle mouse button (three-button mouse)
05-07	Undefined
08	BACKSPACE key
09	TAB key
0A–0B	Undefined
0C	CLEAR key
0D	ENTER key
0E–0F	Undefined
10	SHIFT key
11	CTRL key
12	ALT key
13	PAUSE key
14	CAPS LOCK key
15-19	Reserved for Kanji systems
1A	Undefined
1B	ESC key
1C–1F	Reserved for Kanji systems
20	SPACEBAR
21	PAGE UP key
22	PAGE DOWN key
23	END key
24	HOME key
25	LEFT ARROW key
26	UP ARROW key
27	RIGHT ARROW key
28	DOWN ARROW key
29	SELECT key
2A	OEM specific
2B	EXECUTE key
2C	PRINT SCREEN key for Windows 3.0 and later
2D	INS key
2E	DEL key
2F	HELP key
30	0 key
31	1 key

32	2 key
33	3 key
34	4 key
35	5 key
36	6 key
37	7 key
38	8 key
39	9 key
3A–40	Undefined
41	A key
42	B key
43	C key
44	D key
45	E key
46	F key
47	G key
48	H key
49	I key
4A	J key
4B	K key
4C	L key
4D	M key
4E	N key
4F	O key
50	P key
51	Q key
52	R key
53	S key
54	T key
55	U key
56	V key
57	W key
58	X key
59	Y key
5A	Z key
5B–5F	Undefined
60	Numeric keypad 0 key
61	Numeric keypad 1 key
62	Numeric keypad 2 key
63	Numeric keypad 3 key
64	Numeric keypad 4 key
65	Numeric keypad 5 key
66	Numeric keypad 6 key
67	Numeric keypad 7 key
68	Numeric keypad 8 key
69	Numeric keypad 9 key
6A	Multiply key
6B	Add key

6C	Separator key
6D	Subtract key
6E	Decimal key
6F	Divide key
70	F1 key
71	F2 key
72	F3 key
73	F4 key
74	F5 key
75	F6 key
76	F7 key
77	F8 key
78	F9 key
79	F10 key
7A	F11 key
7B	F12 key
7C	F13 key
7D	F14 key
7E	F15 key
7F	F16 key
80H	F17 key
81H	F18 key
82H	F19 key
83H	F20 key
84H	F21 key
85H	F22 key
86H	F23 key
87H	F24 key
88-8F	Unassigned
90	NUM LOCK key
91	SCROLL LOCK key
92–B9	Unassigned
BA–C0	OEM specific
C1–DA	Unassigned
DB–E4	OEM specific
E5	Unassigned
E6	OEM specific
E7–E8	Unassigned
E9–F5	OEM specific
F6–FE	Unassigned

INDEX

