

# **wIntegrate**

## **Host Subroutines Reference**

Version 6.0.0  
December 2004

IBM Corporation  
555 Bailey Avenue  
San Jose, CA 95141

Licensed Materials – Property of IBM

© Copyright International Business Machines Corporation 2004. All rights reserved.

AIX, DB2, DB2 Universal Database, Distributed Relational Database Architecture, NUMA-Q, OS/2, OS/390, and OS/400, IBM Informix®, C-ISAM®, Foundation.2000™, IBM Informix® 4GL, IBM Informix® DataBlade® module, Client SDK™, Cloudscape™, Cloudsync™, IBM Informix® Connect, IBM Informix® Driver for JDBC, Dynamic Connect™, IBM Informix® Dynamic Scalable Architecture™ (DSA), IBM Informix® Dynamic Server™, IBM Informix® Enterprise Gateway Manager (Enterprise Gateway Manager), IBM Informix® Extended Parallel Server™, i.Financial Services™, J/Foundation™, MaxConnect™, Object Translator™, Red Brick® Decision Server™, IBM Informix® SE, IBM Informix® SQL, InformiXML™, RedBack®, SystemBuilder™, U2™, UniData®, UniVerse®, wIntegrate® are trademarks or registered trademarks of International Business Machines Corporation.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Windows, Windows NT, and Excel are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Other company, product, and service names used in this publication may be trademarks or service marks of others.

Documentation Team: Alan Buckley, Tim Rasmussen, David Robertshaw

#### US GOVERNMENT USERS RESTRICTED RIGHTS

Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

---

<b>1. Reference - Host Subroutines.....</b>	<b>11</b>
WC .....	12
WIN.ACTIVATE .....	13
WIN.APP .....	15
WIN.APPDIR .....	16
WIN.ASCTOFT .....	17
WIN.ASSIGN .....	18
WIN.BARSUB .....	19
WIN.BKIMAGE .....	21
WIN.BOX .....	22
WIN.CBREAD .....	24
WIN.CBWRITE .....	26
WIN.CHECK .....	28
WIN.CLINFO .....	30
WIN.COLOR .....	31
WIN.COLOUR .....	33
WIN.COMLINE .....	35
WIN.COMSUB .....	36
WIN.CURSOR .....	37
WIN.DBANIM .....	38
WIN.DBATTACH .....	40
WIN.DBAXCTRL .....	41
WIN.DBBOX .....	43
WIN.DBBUTTON .....	45
WIN.DBCAPT .....	47
WIN.DBCHECK .....	48
WIN.DBCHILD .....	50
WIN.DBCOMBO .....	52
WIN.DBCTRL .....	55
WIN.DBDEL .....	57
WIN.DBDTIME .....	58
WIN.DBEDGET .....	60
WIN.DBEDIT .....	61

---

WIN.DBEDSET .....	63
WIN.DBENABLE .....	64
WIN.DBENALL .....	65
WIN.DBEND .....	67
WIN.DBESTACK .....	69
WIN.DBEVENT .....	70
WIN.DBEVENT2 .....	73
WIN.DBEVENTS .....	75
WIN.DBFETCH .....	77
WIN.DBFOCUS .....	79
WIN.DBGET .....	81
WIN.DBGETM .....	83
WIN.DBGETPRP .....	85
WIN.DBGRID .....	86
WIN.DBGROUP .....	88
WIN.DBHEADER .....	90
WIN.DBIMAGE .....	92
WIN.DBIMBUT .....	94
WIN.DBINIPRP .....	96
WIN.DBINIT .....	97
WIN.DBINPOK .....	98
WIN.DBLABEL .....	100
WIN.DBLIST .....	103
WIN.DBLISTVW .....	105
WIN.DBLOAD .....	107
WIN.DBMETHOD .....	108
WIN.DBMNSIZE .....	111
WIN.DBMOVE .....	112
WIN.DBMSGBOX .....	113
WIN.DBMSIZE .....	115
WIN.DBMVCTRL .....	117
WIN.DBNEVENT .....	118
WIN.DBNEW .....	120
WIN.DBOPTION .....	123
WIN.DBPANEL .....	125
WIN.DBPOST .....	127
WIN.DBPRGRES .....	129
WIN.DBRADIO .....	131
WIN.DBRECT .....	133

WIN.DBSCROLL.....	135
WIN.DBSELECT .....	137
WIN.DBSET .....	139
WIN.DBSETCOL.....	141
WIN.DBSETFNT .....	142
WIN.DBSETMNU.....	143
WIN.DBSETPRP.....	144
WIN.DBSHOW.....	145
WIN.DBSHOWPU.....	147
WIN.DBSTATUS.....	149
WIN.DBTAB .....	151
WIN.DBTABS.....	153
WIN.DBTEXT .....	154
WIN.DBTRACK.....	156
WIN.DBTREEVW.....	158
WIN.DBUNIT.....	160
WIN.DBUPDOWN.....	162
WIN.DDECLOSE .....	164
WIN.DDEEXEC.....	166
WIN.DDEEXEC2.....	168
WIN.DDEOPEN .....	170
WIN.DDEPOKE .....	172
WIN.DDEREQ.....	174
WIN.DDETIME.....	176
WIN.DISPLAY.....	177
WIN.DRARC .....	178
WIN.DRBRUSH .....	180
WIN.DRCHORD.....	182
WIN.DRELL .....	184
WIN.DRERASE.....	186
WIN.DRFONT .....	188
WIN.DRLINE.....	190
WIN.DRMOVE .....	191
WIN.DRPEN .....	192
WIN.DRPIE .....	195
WIN.DRPOLY .....	197
WIN.DRRECT .....	199
WIN.DRTEXT.....	201
WIN.EDIT.....	203

---

WIN.EFFECT .....	204
WIN.EFILL .....	206
WIN.EI.....	208
WIN.EI2.....	212
WIN.EVAL.....	215
WIN.EXPORT .....	216
WIN.FKEY.....	219
WIN.FSCRIPT.....	221
WIN.FTPCLOSE .....	222
WIN.FTPCON .....	223
WIN.FTPDEL .....	224
WIN.FTPDIR .....	225
WIN.FTPDISC.....	226
WIN.FTPFILE.....	227
WIN.FTPGET.....	228
WIN.FTPGETDR.....	229
WIN.FTPINFO.....	230
WIN.FTPLIST.....	231
WIN.FTPMKDIR.....	233
WIN.FTPOPEN .....	234
WIN.FTPPOS.....	236
WIN.FTPPUT .....	237
WIN.FTPREAD .....	238
WIN.FTPRMDIR .....	239
WIN.FTPSCR.....	240
WIN.FTPSCRIPT .....	241
WIN.FTPSETDR .....	243
WIN.FTPSETV .....	244
WIN.FTPWRITE.....	246
WIN.FTTOASC .....	248
WIN.GETDATA .....	249
WIN.GETLIST .....	250
WIN.GETPARAM.....	251
WIN.GETVAL .....	252
WIN.GETVAR .....	253
WIN.HELP.....	254
WIN.HELPID .....	255
WIN.HELPNAME .....	256
WIN.HGLASS .....	257

WIN.HOSTVER.....	258
WIN.HOTSPOT.....	259
WIN.HOTSPOT2.....	261
WIN.HSCRIPT .....	263
WIN.HSCRIPTC.....	264
WIN.ILADD .....	265
WIN.ILCOUNT .....	266
WIN.ILDELETE.....	267
WIN.ILICON .....	268
WIN.ILINFO .....	269
WIN.ILISIZE .....	270
WIN.ILLOAD .....	271
WIN.ILNEW.....	273
WIN.ILREMOVE .....	275
WIN.IMAGE .....	276
WIN.IMCHANGE.....	278
WIN.IMCLOSE.....	280
WIN.IMOPEN.....	282
WIN.IMPORT .....	284
WIN.INFOBOX.....	288
WIN.INPBOX .....	289
WIN.INVOKE .....	290
WIN.LI .....	291
WIN.LICINFO.....	292
WIN.LOOKUP .....	293
WIN.MENUATT.....	295
WIN.MENUDEL .....	297
WIN.MENUDET .....	299
WIN.MENUIN.....	301
WIN.MENULOAD.....	303
WIN.MLADDR.....	305
WIN.MLAVAIL.....	307
WIN.MLDELETE .....	308
WIN.MLFIND.....	310
WIN.MLLOOKUP .....	313
WIN.MLNEXT .....	315
WIN.MLREAD .....	318
WIN.MLSEND .....	321
WIN.MOUSE.....	324

---

WIN.MOUSEDEF .....	326
WIN.MOUSEIN .....	327
WIN.MSGBOX .....	329
WIN.MSTATE .....	332
WIN.OBEXIST .....	334
WIN.OBGET .....	335
WIN.OBGETPRP .....	337
WIN.OBMETHOD .....	338
WIN.OBNEW .....	340
WIN.OBREL .....	341
WIN.OBSET .....	342
WIN.OBSETPRP .....	343
WIN.OBVTYPE .....	344
WIN.PCBROWSE .....	345
WIN.PCCLOSE .....	347
WIN.PCCOPY .....	348
WIN.PCCREATE .....	349
WIN.PCDELETE .....	351
WIN.PCDIR .....	352
WIN.PCEDIT .....	354
WIN.PCEOF .....	356
WIN.PCFILE .....	357
WIN.PCINFO .....	359
WIN.PCLIST .....	360
WIN.PCMKDIR .....	362
WIN.PCMOVE .....	364
WIN.PCOPEN .....	365
WIN.PCPOS .....	366
WIN.PCPRINT .....	368
WIN.PCREAD .....	369
WIN.PCREADAL .....	371
WIN.PCREADLN .....	373
WIN.PCRMDIR .....	375
WIN.PCRUN .....	376
WIN.PCRUN2 .....	378
WIN.PCSCRIPT .....	381
WIN.PCWRITE .....	382
WIN.PIESUB .....	384
WIN.PLAYBACK .....	387



WIN.POPUPIN.....	389
WIN.PRINTOFF.....	391
WIN.PRINTON.....	392
WIN.PRTPAUSE.....	393
WIN.RECOFF.....	395
WIN.RECON.....	397
WIN.RECPAUSE.....	399
WIN.RSEXEC.....	401
WIN.RSEXIST.....	403
WIN.RSNAME.....	404
WIN.RSSCRIPT.....	405
WIN.RSSTART.....	407
WIN.SCREEN.....	409
WIN.SDUMP.....	410
WIN.SENDKEYS.....	411
WIN.SERIAL.....	413
WIN.SERVER.....	414
WIN.SETDATA.....	415
WIN.SETEFFECT.....	416
WIN.SETLIST.....	418
WIN.SETPARAM.....	419
WIN.SETVAL.....	420
WIN.SETVAR.....	421
WIN.SHARE.....	422
WIN.SHOW.....	423
WIN.SLOAD.....	424
WIN.SPOOL.....	426
WIN.SPULL.....	428
WIN.SPUSH.....	429
WIN.SREMOVE.....	430
WIN.SRESTORE.....	432
WIN.SSAVE.....	433
WIN.SSTATE.....	434
WIN.SSTORE.....	436
WIN.STACK.....	438
WIN.STACKOFF.....	439
WIN.STACKON.....	440
WIN.STATLINE.....	441
WIN.TASK.....	442

---

WIN.TCL .....	444
WIN.TITLE .....	445
WIN.TRANSFER.....	446
WIN.TWCLOSE .....	448
WIN.TWFOOT .....	450
WIN.TWMSG .....	451
WIN.TWOPEN .....	452
WIN.TWPULL .....	455
WIN.TWPUSH .....	457
WIN.TWUSE .....	459
WIN.USESTYLE .....	461
WIN.VERSION.....	463
WIN.XLADDWS .....	464
<b>Appendix A. Using Host Subroutines .....</b>	<b>467</b>
1. Bar Graphs.....	470
2. Pie Charts .....	471
3. Images .....	472
4. Drawing.....	473
5. Colors.....	478
6. Text Windows .....	479
7. HotSpots .....	481
8. Dynamic Data Exchange Demo.....	482
9. Menus .....	486
10. DialogBox lookup .....	488
11. Dialog Boxes.....	489
12. Chiselled Effects .....	493
<b>Appendix B. The Service Subroutine .....</b>	<b>495</b>
<b>Index .....</b>	<b>505</b>

# Reference - Host Subroutines

---

This chapter details each of the subroutines provided with wIntegrate. You can use these subroutines to develop your application. Many of these subroutines make a call to one or more wIntegrate scripts. The scripts are documented in another manual, Client Scripting Reference.

# WC

## Syntax

WC

## Description

This is the abbreviation for the WIN.COMLINE command. It gives you the ability to directly enter script commands on the screen.

## Parameters

None

## Example

Change your screen colors

---

WC

This command produces a command line and prompt:

Enter ? for help

wIntegrate Command >

You can enter any wIntegrate command at this prompt. For example:

wIntegrate Command >Set Effect\_Normal RGB\_LightCyan, RGB\_Blue

This command changes your screen to a light blue foreground (text), with a darker blue background.

---

## Related Subroutines

[WIN.COMSUB](#), [WIN.HSCRIPT](#), [WIN.COMLINE](#)

# WIN.ACTIVATE

## Syntax

**WIN.ACTIVATE** ( *TASKNAME* )

## Description

This subroutine activates a window on the Windows desktop. The window that you want to show must be a running application (accessed by pressing ALT+TAB to scroll through running applications). The taskname must be the same as it displays in the Windows task list or the task bar.

## Parameters

The following table describes the parameters of the WIN.ACTIVATE command:

Parameter	Description
<i>TASKNAME</i>	The name of the window as it is shown in the Windows task list. If the taskname is null, the subroutine activates the wIntegrate window.

## Example

This example prints out a message if the task is not running. otherwise it inserts the report name on the first line.

```
* Verify the report "recrept.txt" is running
CALL WIN.TASK("Notepad - RECREPT.TXT", RUNNING)
IF RUNNING THEN
    GOSUB 100
END ELSE
    PRINT "recrept.txt is not running, open it and try again"
END
RETURN
*
100 * Change the active window from wIntegrate to Notepad
CALL WIN.ACTIVATE("Notepad - RECREPT.TXT")
* Send the report name to the Notepad report
CALL WIN.SENDKEYS("STUDENT REPORT", 1)
RETURN
*
END
```

## Related Subroutines

[WIN.TASK](#)

## Related Script Commands

Activate

# WIN.APP

## Syntax

**WIN.APP** ( *FILENAME*, *RUNNING* )

## Description

This subroutine checks if an application is running.

## Parameters

The following table describes the parameters of the WIN.APP command:

Parameter	Description
<i>FILENAME</i>	Specifies the name of the application to check
<i>RUNNING</i>	Returns: 0 If application is not running. 1 If application is running

## Example

The following checks for Windows Microsoft Word application:

---

The following checks for Windows Microsoft Word application:

```
CALL WIN.APP ("WINWORD.EXE", RUNNING)
```

```
IF RUNNING THEN PRINT "Word for Windows is running"
```

When this program is run, it returns nothing if Word is not running, and returns the following if Word is running:

```
Word for Windows is running
```

---

# WIN.APPDIR

## Syntax

WIN.APPDIR ( *DIR* )

## Description

This subroutine returns the name of the directory of the current wIntegrate session.

## Parameters

The following table describes the parameters of the WIN.APPDIR command:

Parameter	Description
<i>DIR</i>	Variable to hold the returned directory

## Example

This example finds wIntegrate's running directory.

---

This example finds wIntegrate's running directory.

```
DIR = ""
```

```
CALL WIN.APPDIR(DIR)
```

```
PRINT DIR
```

The output of this program is:

```
C:\Program Files\wIntegrate
```

---



# WIN.ASCTOFT

## Syntax

WIN.ASCTOFT ( *VALUE* )

## Description

This function converts an ASCII format string to its File Transfer equivalent. The file transfer format represents control codes and other special characters as one or two seven-bit values. The seven-bit data can be sent across any communications line.

## Parameters

The following table describes the parameters of the WIN.ASCTOFT command:

Parameter	Description
<i>VALUE</i>	The variable to convert to FT format

## Related Subroutines

[WIN.FTTOASC](#)

# WIN.ASSIGN

## Syntax

**WIN.ASSIGN** ( *VAR*, *VALUE* )

## Description

Assigns the given variable with the specified value. Unlike WIN.SETVAR this routine does not automatically create a global script variable.

## Parameters

The following table describes the parameters of the WIN.ASSIGN command:

Parameter	Description
<i>VAR</i>	The name of the variable to assign
<i>VALUE</i>	The value to assign to the variable

## Related Subroutines

[WIN.SETVAR](#)

# WIN.BARSUB

## Syntax

**WIN.BARSUB** ( *TITLE*, *LABELS*, *VALUES*, *OPTIONS* )

## Description

This subroutine uses the wIntegrate draw commands to create a bar chart.

## Parameters

The following table describes the parameters of the WIN.BARSUB command:

Parameter	Description
<i>TITLE</i>	Specifies the bar chart name.
<i>LABELS</i>	Labels the x-axis multivalue list
<i>VALUES</i>	Specifies values for each label.
<i>OPTIONS</i>	Options for the chart.

## Fields for OPTIONS

Field	Description
<i>1</i>	Specifies the sty. 1 - Normal, 2 - several data sets.
<i>2</i>	Screen position
<i>3</i>	Reserved
<i>4</i>	Colors for the bars
<i>5</i>	Y-axis labels: 5.1 minimum value, 5.2 - maximum value, 5.3 - step value

## Example

The following lines are taken from the demo program WIN.BARDEMO.

---

```
LABELS = ''
VALUES = ''
LABELS<1> = 'Computers'; VALUES<1> =200
LABELS<2> = 'Stationery'; VALUES<2> = 500
LABELS<3> = 'Accessories'; VALUES<3> = 50
*
R.OPTS = ''
R.OPTS<2,1> = 20
R.OPTS<2,2> = 3
R.OPTS<2,3> = 60
R.OPTS<2,4> = 15
*
CALL WIN.BARSUB("Products sold", LABELS, VALUES, R.OPTS)
```

---

# WIN.BKIMAGE

## Syntax

**WIN.BKIMAGE** ( *IMAGE*, *DISPLAY*, *OPTS* )

## Description

This command specifies a background image for the session screen.

## Parameters

The following table describes the parameters of the WIN.BKIMAGE command:

Parameter	Description
<i>IMAGE</i>	The file name of the image file used. If only the leaf name of the file name is used, the image is assumed to be in wIntegrate's backgrnd folder
<i>DISPLAY</i>	Tile or Stretch the image. Use literal values "Tile" or "Stretch" respectively
<i>OPTS</i>	Set to "". Reserved for future expansion

## Example

The following code extract is from the host program WIN.BKDEMO.

```
IF REDISP THEN
  IF DISP = "" THEN DISP = "Stretch"
  CALL WIN.BKIMAGE(IMAGE, DISP, "")
  CALL WIN.SETEFFECT("Normal",COL,0,"",0)
  PRINT CMSG:' CALL WIN.BKIMAGE('':IMAGE:'',':DISP:'', '')':
END
```

## Version

4.0.3 Original version

# WIN.BOX

## Syntax

**WIN.BOX** ( *LEFT*, *TOP*, *RIGHT*, *BOTTOM*, *STYLE* )

## Description

This subroutine draws a box on the screen.

## Parameters

The following table describes the parameters of the WIN.BOX command:

Parameter	Description
<i>LEFT</i>	Specifies the coordinate for the left side of the box.
<i>TOP</i>	Specifies the coordinate for the top of the box.
<i>RIGHT</i>	Specifies the coordinate for the right side of the box.
<i>BOTTOM</i>	Specifies the coordinate for the bottom of the box.
<i>STYLE</i>	Style of the box

## Values for STYLE

Enter the name in quotes or number for each style

Value	Description
<i>0 "SPACES"</i>	No visible border
<i>1 "SINGLE"</i>	Single line border
<i>2 "DOUBLE"</i>	Double line border
<i>3 "MIX"</i>	Alternating dot and block border
<i>4 "BLOCK"</i>	Block border
<i>+512 "MERGE"</i>	Merges box lines with lines already on the screen
<i>+256 "NOFILL"</i>	Leaves the box empty

## Example

This example draws a single line across line 1 on the screen

---

```
CALL WIN.BOX(0,1,79,1,1)
```

---

The preceding example could also be written:

---

```
CALL WIN.BOX(0,1,79,1,"SINGLE")
```

---

The following example is taken from WIN.DEMO

---

```
PRINT @(40,11):"12. Chiselled effects"  
CALL WIN.COLOR("Yellow","") ;* Set foreground colour  
CALL WIN.BOX(0,19,79,19,1) ;* Draw line across bottom
```

---

For a single line merged with the screen on line 11

---

```
CALL WIN.BOX(5,11,75,11,"SINGLE,MERGE")
```

---

For a box with double lines on the top and bottom and single lines on the sides:

---

```
CALL WIN.BOX(20,10,60,15,"DOUBLE, SINGLE")
```

---

## Related Script Commands

Display Box

# WIN.CBREAD

## Syntax

**WIN.CBREAD** ( *TEXT*, *RESP* )

## Description

This routine reads text from the Windows clipboard. It converts cr/lf to a field mark and tab to a value mark.

## Parameters

The following table describes the parameters of the WIN.CBREAD command:

Parameter	Description
<i>TEXT</i>	Variable to be set with the text from the clipboard
<i>RESP</i>	Error code, see table below

## Values for TEXT

Value	Description
<i>1</i>	Unable to connect to the clipboard
<i>2</i>	Unable to allocate memory to transfer data
<i>3</i>	Data on clipboard is not text
<i>4</i>	There is more than 30K of data on the clipboard

## Related Subroutines

[WIN.CBWRITE](#)

## Related Script Commands

Clipboard Read



## **Version**

### 4.1 Original version

# WIN.CBWRITE

## Syntax

WIN.CBWRITE ( *TEXT*, *RESP* )

## Description

This routine writes text on to the Windows clipboard. It converts Field marks to cr/lf pairs and value marks to tabs.

## Parameters

The following table describes the parameters of the WIN.CBWRITE command:

Parameter	Description
<i>TEXT</i>	The text to put on the clipboard.
<i>RESP</i>	Error code, see table below

## Values for TEXT

Value	Description
1	Unable to connect to the clipboard
2	Unable to allocate memory to transfer data

## Values for RESP

Error code returned in RESP

Value	Description
1	Unable to connect to the clipboard
2	Unable to allocate memory to transfer data

## Related Subroutines

[WIN.CBREAD](#)

## **Related Script Commands**

Clipboard Write

## **Version**

4.1 Original version

# WIN.CHECK

## Syntax

**WIN.CHECK** ( *RUNNING* )

## Description

This subroutine checks if wIntegrate is running.

WIN.CHECK itself is aimed at the GENERIC machine type while there is a WIN.CHECK.UD for UniData hosts and WIN.CHECK.UV for UniVerse hosts.

This subroutine has a side effect of moving the cursor to the beginning of the current line and erasing the first 10 characters (needed to erase the check sequence on non-wIntegrate terminals).

## Parameters

The following table describes the parameters of the WIN.CHECK command:

Parameter	Description
<i>RUNNING</i>	This value is set to 1 if wIntegrate is running and 0 if wIntegrate is not running.

## Example

Check if wIntegrate is running on UniVerse

---

```
RUNNING = 0
CALL WIN.CHECK.UV(RUNNING)
IF RUNNING THEN
    GOSUB 100;* Show GUI menu
END ELSE
    GOSUN 200;* Show character menu
END
```

---

## **Version**

5.1.2 Original

# WIN.CLINFO

## Syntax

WIN.CLINFO ( *INFO* )

## Description

This routine returns information on the client machine and user running this session

## Parameters

The following table describes the parameters of the WIN.CLINFO command:

Parameter	Description
<i>INFO</i>	Variable to be set with a dynamic array of the returned client information

## Fields for INFO

Field	Description
<i>1</i>	The name the user logged into Windows with
<i>2</i>	The computer name

## Version

4.2.1 Original

# WIN.COLOR

## Syntax

**WIN.COLOR** ( *FOREGROUND*, *BACKGROUND* )

## Description

Sets the current color on the terminal. If you set **FOREGROUND** or **BACKGROUND** to null, the component color does not change. To turn off colored text, set **FOREGROUND** to "Off". **FOREGROUND** and **BACKGROUND** can be set to a color name (Black, Blue, LightCyan) or to a number from 0 to 15. For descriptions of the colors corresponding to these numbers, see the Color table that follows the Parameters table.

## Parameters

The following table describes the parameters of the WIN.COLOR command:

Parameter	Description
<i>FOREGROUND</i>	Specifies the foreground color (text).
<i>BACKGROUND</i>	Specifies the background color.

## Values for FOREGROUND

Color values/names. Also applies to the background parameter

Value	Description
<i>0</i>	Black
<i>1</i>	Blue
<i>2</i>	Green
<i>3</i>	Cyan
<i>4</i>	Red
<i>5</i>	Magenta
<i>6</i>	Brown
<i>7</i>	LightGray or LightGrey

Value	Description
8	Gray or Grey
9	LightBlue
10	LightGreen
11	LightCyan
12	LightRed
13	LightMagenta
14	Yellow
15	White

## Example

The following example is taken from the host subroutine WIN.CEDEMO.

---

The following example is taken from the host subroutine WIN.CEDEMO.

```
* Set up box to display information
CALL WIN.COLOR("White","Blue")
CALL WIN.TWOPEN("INFO","Information",1,18,78,22,"DOUBLE")
CALL WIN.COLOR("Black","White")
CALL WIN.TWFOOT("INFO","Press <CR> to continue","R
CALL WIN.COLOR(TEXT.COL,"Blue")
TEXT.LEN = 77
*
```

This part of the subroutine refers to the Information box at the bottom of the screen:

---

## Related Subroutines

[WIN.COLOUR](#)

## Related Script Commands

Color



# WIN.COLOUR

## Syntax

**WIN.COLOUR** ( *FOREGROUND*, *BACKGROUND* )

## Description

Sets the current color on the terminal. If you set **FOREGROUND** or **BACKGROUND** to null, the component color does not change. To turn off colored text, set **FOREGROUND** to "Off". **FOREGROUND** and **BACKGROUND** can be set to a color name (Black, Blue, LightCyan) or to a number from 0 to 15. For descriptions of the colors corresponding to these numbers, see the Color table that follows the Parameters table.

This routine is identical to the **WIN.COLOR** subroutine

## Parameters

The following table describes the parameters of the **WIN.COLOUR** command:

Parameter	Description
<i>FOREGROUND</i>	Specifies the foreground color (text).
<i>BACKGROUND</i>	Specifies the background color.

## Values for FOREGROUND

Color values/names. Also applies to the background parameter

Value	Description
<i>0</i>	Black
<i>1</i>	Blue
<i>2</i>	Green
<i>3</i>	Cyan
<i>4</i>	Red
<i>5</i>	Magenta

Value	Description
<i>6</i>	Brown
<i>7</i>	LightGray or LightGrey
<i>8</i>	Gray or Grey
<i>9</i>	LightBlue
<i>10</i>	LightGreen
<i>11</i>	LightCyan
<i>12</i>	LightRed
<i>13</i>	LightMagenta
<i>14</i>	Yellow
<i>15</i>	White

## Related Subroutines

[WIN.COLOR](#)

## Related Script Commands

Colour

# WIN.COMLINE

## Syntax

WIN.COMLINE

## Description

This subroutine runs script commands and host script names from the host. In addition to using this subroutine in your host routines, you can use it to test command lines before incorporating them into your application code. When you see the prompt for a wIntegrate command, you can enter a question mark for help on how to use this subroutine.

## Parameters

None

## Example

Running WIN.COMLINE to change the screen color

---

WIN.COMLINE

This command produces a command line and prompt:

Enter ? for help

wIntegrate Command >

You can enter any wIntegrate command at this prompt. For example:

wIntegrate Command >Set Effect\_Normal RGB\_LightCyan, RGB\_Blue

This command changes your wIntegrate screen to a light blue foreground (text), with a darker blue background.

---

## Related Subroutines

[WC](#), [WIN.COMSUB](#), [WIN.HSCRIPT](#)

# WIN.COMSUB

## Syntax

WIN.COMSUB ( *REC* )

## Description

This subroutine executes a script on the PC.

Note: Every field in the dynamic array REC is executed as a separate script. To execute REC as a single script use the WIN.HSCRIPT subroutines.

## Parameters

The following table describes the parameters of the WIN.COMSUB command:

Parameter	Description
<i>REC</i>	The PC script to execute.

## Example

The following example is taken from the WIN.CEDEMO demonstration program.

```
* Have to temporarily put scrollregion back to whole screen
*
CALL WIN.COMSUB("Screen ScrollRegion")
```

## Related Subroutines

[WC](#), [WIN.COMLINE](#), [WIN.HSCRIPT](#)

# WIN.CURSOR

## Syntax

**WIN.CURSOR** ( *TYPE*, *TURN.ON* )

## Description

Sets the cursor shape and turns the cursor on or off.

## Parameters

The following table describes the parameters of the WIN.CURSOR command:

Parameter	Description
<i>TYPE</i>	Specifies the type of cursor you want to show.
<i>TURN.ON</i>	"ON" shows the cursor. "OFF" hides the cursor.

## Values for TYPE

Use one of the following literal values

Value	Description
<i>"LINE"</i>	displays a line cursor.
<i>"BLOCK"</i>	displays a line cursor.
<i>""</i>	displays a line cursor.

## Example

Turn on line cursor

---

```
CALL WIN.CURSOR("LINE", "ON")
```

This command turns on the cursor as a blinking line.

---

# WIN.DBANIM

## Syntax

**WIN.DBANIM** ( *DBX*, *NAME*, *X*, *Y*, *W*, *D*, *STYLE* )

## Description

This subroutine adds a animation control to a dialog box. The animation control displays a simple animation on a dialog box.

Use WIN.DBEVENT to set up the events to be returned by the control. The initial properties of a control can be set with WIN.DBINIPRP. After the dialog box containing the control has been shown properties can be set and retrieved using WIN.DBSETPRP and WIN.DBGETPRP. The methods of the control can be run at this time using WIN.DBMETHOD.

For a full list of properties, methods and events see the Client scripting reference.

## Parameters

The following table describes the parameters of the WIN.DBANIM command:

Parameter	Description
<i>DBX</i>	This is the wIntegrate "handle" for a dialog box.
<i>NAME</i>	Controls name
<i>X</i>	Column position
<i>Y</i>	Row position
<i>W</i>	Width
<i>D</i>	Depth
<i>STYLE</i>	See the Client scripting reference for details

## Related Subroutines

[WIN.DBINIPRP](#), [WIN.DBEVENTS](#), [WIN.DBSETPRP](#), [WIN.DBGETPRP](#),  
[WIN.DBMETHOD](#)

## **Related Script Commands**

DialogBox Animate

## **Version**

4.0.1 Original

# WIN.DBATTACH

## Syntax

**WIN.DBATTACH** ( *DLGNAME*, *PARENT*, *ERR* )

## Description

This subroutine attaches a modal dialog box to an existing dialog. When a dialog box is attached, the parent dialog cannot receive input until the attached box is closed.

## Parameters

The following table describes the parameters of the WIN.DBATTACH command:

Parameter	Description
<i>DLGNAME</i>	Specifies the dialog box to display.
<i>PARENT</i>	Specifies the main dialog box.
<i>ERR</i>	Returns an error code. 0 = No error 1 = Missing the dialog box name 2 = Dialog box is already displayed 3 = Dialog box is not loaded 4 = Unable to create dialog box window

## Example

This example prints an error message if there is an error attaching the dialog.

---

```
CALL WIN.DBATTACH(DBSURE, DBXNAME, ERR)
IF ERR THEN PRINT "CANT ATTACH SURE DIALOG ":ERR; RETURN
```

---

## Related Subroutines

[WIN.DBSHOW](#)

## Related Script Commands

DialogBox Show



# WIN.DBAXCTRL

## Syntax

**WIN.DBAXCTRL** ( *DBX*, *NAME*, *PROGID*, *X*, *Y*, *W*, *D*, *STYLE* )

## Description

This subroutine adds an Active X control to a dialog box.

Use WIN.DBEVENT to set up the events to be returned by the control. The initial properties of a control can be set with WIN.DBINIPRP. After the dialog box containing the control has been shown properties can be set and retrieved using WIN.DBSETPRP and WIN.DBGETPRP. The methods of the control can be run at this time using WIN.DBMETHOD.

## Parameters

The following table describes the parameters of the WIN.DBAXCTRL command:

Parameter	Description
<i>DBX</i>	This is the wIntegrate "handle" for a dialog box.
<i>NAME</i>	Controls name
<i>PROGID</i>	The PROGID or class id enclosed i n { } for the Active X control
<i>X</i>	Column position
<i>Y</i>	Row position
<i>W</i>	Width
<i>D</i>	Depth
<i>STYLE</i>	See the table below for a list of styles that can be used with Active X controls

## Values for STYLE

Value	Description
<i>WS_DISABLED</i>	Disables the control.

Value	Description
<i>WS_GROUP</i>	Makes this control the first in a group. Up and down arrows move to the previous or next control in the group.
<i>WS_TABSTOP</i>	Tab or back tab moves to the previous or next control with WS_TABSTOP style.
<i>WS_VISIBLE</i>	The control is visible (the default).

## Related Subroutines

[WIN.DBINIPRP](#), [WIN.DBEVENTS](#), [WIN.DBSETPRP](#), [WIN.DBGETPRP](#),  
[WIN.DBMETHOD](#)

## Related Script Commands

DialogBox AxControl

## Version

4.0.2 Original

# WIN.DBBOX

## Syntax

**WIN.DBBOX** ( *DBX*, *X*, *Y*, *W*, *D*, *STYLE* )

## Description

This subroutine adds a 3d box or line to a dialog box. This is used as a visual way to enclose or separate area of the dialog box.

## Parameters

The following table describes the parameters of the WIN.DBBOX command:

Parameter	Description
<i>DBX</i>	This is the wIntegrate "handle" for a dialog box.
<i>X</i>	Column position
<i>Y</i>	Row position
<i>W</i>	Width
<i>D</i>	Depth
<i>STYLE</i>	See the following styles table

## Values for STYLE

Value	Description
<i>"SUNKEN" or "S"</i>	Sunken rectangle
<i>"RAISED" or "R"</i>	Raised rectangle
<i>"ETCHED" or "E"</i>	Etched rectangle
<i>"HORZ" or "H"</i>	Etched horizontal line
<i>"VERT" or "V"</i>	Etched vertical line
<i>"FRAME" or "F"</i>	Etched Frame (this is identical to "ETCHED")

## Example

The following defines a dialog box which shows all the box styles

---

```
*
SINK.TEXT = "";SINK.TEXT<3> = 1; * Sunken text option for text and
labels
*
CALL WIN.DBNEW(DBX, DLG.NAME, "Static control demo",8,8,252,106,"","")
CALL WIN.DBLABEL(DBX, "This is a normal label",5,8,78,12,"")
CALL WIN.DBLABEL(DBX, "This is a sunken label",141,8,78,12,SINK.TEXT)
CALL WIN.DBTEXT(DBX, "Text1", 5,22,78,12,"")
CALL WIN.DBINIPRP(DBX,"Text1","", "This is normal text")
CALL WIN.DBTEXT(DBX, "Text2", 141,22,78,12, SINK.TEXT)
CALL WIN.DBINIPRP(DBX,"Text2","", "This is sunken text")
CALL WIN.DBLABEL(DBX, "Sunken box:",5,46,53,12,"")
CALL WIN.DBBOX(DBX, 71,44,40,12,"SUNKEN")
CALL WIN.DBLABEL(DBX, "Raised box:",5,66,40,12,"")
CALL WIN.DBBOX(DBX, 71,65,40,12,"RAISED")
CALL WIN.DBLABEL(DBX, "Etched box:",5,86,40,12,"")
CALL WIN.DBBOX(DBX, 71,84,40,12,"ETCHED")
CALL WIN.DBLABEL(DBX, "Horizontal line:",128,46,54,12,"")
CALL WIN.DBBOX(DBX, 196,44,40,12,"HORZ")
CALL WIN.DBLABEL(DBX, "Vertical line:",128,66,40,12,"")
CALL WIN.DBBOX(DBX,196,65,40,12, "VERT")
CALL WIN.DBLABEL(DBX, "Etched Frame:",128,86,52,12,"")
CALL WIN.DBBOX(DBX, 196,84,40,12, "FRAME")
*
CALL WIN.DBBUTTON(DBX,"Cancel", "Close",111,120,40,12,1)
*
```

---

## Version

### 4.1.1 Original

# WIN.DBBUTTON

## Syntax

**WIN.DBBUTTON** ( *DBX*, *NAME*, *TEXT*, *X*, *Y*, *W*, *D*, *DEF* )

## Description

This subroutine adds a raised rectangle push button to a dialog box. Use push-buttons to initiate some kind of action in the dialog box.

The unit of measure for the parameters *X*, *Y*, *W*, and *D* is a dialog box unit. This is calculated as 1/4 of the average width and 1/8 of the average depth of the dialog box font.

## Parameters

The following table describes the parameters of the WIN.DBBUTTON command:

Parameter	Description
<i>DBX</i>	This is the wIntegrate "handle" for a dialog box.
<i>NAME</i>	Specifies the name of the button.
<i>TEXT</i>	Specifies the text of the button caption.
<i>X</i>	Specifies the button's column position.
<i>Y</i>	Specifies the button's row position.
<i>W</i>	Defines the button width. The default width is 40.
<i>D</i>	Defines the button depth. The default is 14.
<i>DEF</i>	Specifies that this push-button is the default button. An "OK" button is usually the default in a dialog box. 0 = Not the default button, 1 = Default button.

## Example

The following example comes from WIN.DBDEMO

```
CALL WIN.DBBUTTON(DBX,"OK","OK",158,4,'',' ',1)
CALL WIN.DBBUTTON(DBX,"Cancel","Cancel",158,20,'',' ',0)
```

The two lines listed above produce the "OK" and "Cancel" buttons in

the larger dialog box. The "OK" button is the default, displayed as a box with a defined border.

---

## **Related Subroutines**

[WIN.DBNEW](#), [WIN.DBLOAD](#), [WIN.DBCAPT](#), [WIN.DBEVENT](#)

## **Related Script Commands**

DialogBox Pushbutton, DialogBox DefPushButton

# WIN.DBCAPT

## Syntax

**WIN.DBCAPT** ( *DLGNAME*, *NAME*, *TEXT* )

## Description

This subroutine changes the caption of an existing window or button.

## Parameters

The following table describes the parameters of the WIN.DBCAPT command:

Parameter	Description
<i>DLGNAME</i>	The name of the dialog box.
<i>NAME</i>	Specifies the name of a button. Use "" to change the window title.
<i>TEXT</i>	Specifies the new caption.

## Example

This example changes the "Cancel" button text to "Close".

---

```
* Change the Cancel button name to Close  
CALL WIN.DBCAPT("DEMODLG", "CANCEL", "Close")
```

---

# WIN.DBCHECK

## Syntax

**WIN.DBCHECK** ( *DBX*, *NAME*, *TEXT*, *X*, *Y*, *W*, *D* )

## Description

This subroutine adds a square check box to a dialog box. A check box can be turned on or off. When a check box is turned on, it displays an "x", when it is turned off, it is empty.

Note: When you want the user to be able to select more than one of several options, use radio buttons in a group box.

## Parameters

The following table describes the parameters of the WIN.DBCHECK command:

Parameter	Description
<i>DBX</i>	This is the wIntegrate "handle" for a dialog box.
<i>NAME</i>	Specifies the checkbox name.
<i>TEXT</i>	Specifies the text for the check box caption.
<i>X</i>	Specifies the check box column position.
<i>Y</i>	Specifies the check box position.
<i>W</i>	Defines the check box width. The default width is 40
<i>D</i>	Defines the check box depth. The default is 14.

## Example

The following example as part of the WIN.DBDEMO program. Run this program from your database prompt

```
CALL WIN.DBNEW(DBX, DLG.NAME, "Demonstration of Dialog boxes",  
10,10,200,160, ' ', ' ' )  
*  
CALL WIN.DBLABEL(DBX, "Label 1", 4, 4, ' ', ' ', ' ' )  
CALL WIN.DBEDIT(DBX, "E1", 40, 4, 40, ' ', ' ' )
```



```
CALL WIN.DBTEXT(DBX,"T1",4,24,80,' ','C')  
CALL WIN.DBCHECK(DBX,"C1","Check one",4,36,60,'')  
*
```

---

## **Related Subroutines**

[WIN.DBNEW](#), [WIN.DBLOAD](#), [WIN.DBCAPT](#), [WIN.DBEVENT](#)

## **Related Script Commands**

DialogBox CheckBox, DialogBox ControlCommand

# WIN.DBCHILD

## Syntax

WIN.DBCHILD ( *DLGNAME*, *PARENT*, *ERR* )

## Description

Attach a modeless dialog box to an existing dialog box.

The child dialog should be given the WS\_CHILD style when it is created.

## Parameters

The following table describes the parameters of the WIN.DBCHILD command:

Parameter	Description
<i>DLGNAME</i>	The name of the dialog box to be attached
<i>PARENT</i>	The name of the dialog box to be attached to
<i>ERR</i>	Set to 0 if OK, otherwise an error number. See below.

## Values for ERR

Error Numbers returned

Value	Description
0	No error
1	Dialog box name is invalid
2	Dialog box is already displayed
3	Dialog box with this name has not been created
4	Unable to display dialog box

## **Related Subroutines**

[WIN.DBSHOW](#)

## **Related Script Commands**

DialogBox Window

## **Version**

4.0.1

# WIN.DBCOMBO

## Syntax

WIN.DBCOMBO ( *DBX*, *NAME*, *X*, *Y*, *W*, *D*, *OPTS* )

## Description

This subroutine adds a combo box to a dialog box. A combo box is a text field or edit field combined with a list box.

The list box can be displayed at all times (CBS\_SIMPLE), or pulled down by the user (CBS\_DROPDOWN or CBS\_DROPDOWNLIST).

Populate the box with data using WIN.DBSET, and select a default option with WIN.DBSELECT.

See the ComboBox control in the "Client Script Reference" manual for more details

## Parameters

The following table describes the parameters of the WIN.DBCOMBO command:

Parameter	Description
<i>DBX</i>	This is the wIntegrate "handle" for a dialog box.
<i>NAME</i>	Specifies the combo box name.
<i>X</i>	Specifies the combo box column position.
<i>Y</i>	Specifies the combo box row position.
<i>W</i>	Defines the combo box width. The default width is 40.
<i>D</i>	Defines the combo box depth. The default is 14.
<i>OPTS</i>	Options for the combo box. See below.

## Fields for OPTS

This is a dynamic array with two fields:

Field	Description
1	Style - Combo box style flags. See the combo box control "Reference - Script Control" in the "Client Scripting Reference" manual.
2	Maximum field length.

## Example

The following code is from WIN.DBDEMO host program. Notice that WIN.DBCOMBO creates the box, and later in the program, WIN.DBSET sets the data for the box. WIN.DBSELECT selects one of the data options as the default.

```
CALL WIN.DBLIST(DBX, "L1", 68,52,50,52, "WS_BORDER | WS_TABSTOP |  
LBS_NOTIFY")  
CALL WIN.DBCOMBO(DBX,"CB1",122,52,50,52, '')  
CALL WIN.DBSCROLL(DBX,"S1", 4,116,60, '', 'WS_TABSTOP', '', '')  
CALL WIN.DBIMAGE(DBX,"I1", "image\computer.wmf",68,116,100,40)  
.  
.  
.  
*  
IF ERR = 0 THEN  
  CALL WIN.DBSET(DLG.NAME,"T1","Centered Text")  
  CALL WIN.DBSET(DLG.NAME,"R1",1)  
  R.LIST = "Option 1"  
  R.LIST<-1>="Option 2"  
  R.LIST<-1>="Option 3"  
  CALL WIN.DBSET(DLG.NAME,"L1", R.LIST)  
  CALL WIN.DBSELECT(DLG.NAME,"L1", "Option 2")  
  CALL WIN.DBSET(DLG.NAME,"CB1",R.LIST)  
  CALL WIN.DBSELECT(DLG.NAME,"CB1", "Option 3")  
END
```

## Related Subroutines

[WIN.DBNEW](#), [WIN.DBLOAD](#), [WIN.DBSET](#), [WIN.DBSELECT](#), [WIN.DBEVENT](#)

## **Related Script Commands**

DialogBox ComboBox, DialogBox Validate

# WIN.DBCTRL

## Syntax

**WIN.DBCTRL** ( *DBX*, *NAME*, *CLASS*, *TEXT*, *X*, *Y*, *W*, *D*, *STYLE* )

## Description

This subroutine adds a control of the specified class to the dialog box. It is an advanced routine that is used when a specialist set of styles is required for a control which isn't available using the standard control routines (WIN.DBTEXT etc).

## Parameters

The following table describes the parameters of the WIN.DBCTRL command:

Parameter	Description
<i>DBX</i>	This is the wIntegrate "handle" for a dialog box.
<i>NAME</i>	Controls name
<i>CLASS</i>	The class name of the control
<i>TEXT</i>	The text for the control if this specific control class requires text
<i>X</i>	Column position
<i>Y</i>	Row position
<i>W</i>	Width
<i>D</i>	Depth
<i>STYLE</i>	Styles appropriate to the control

## Example

Add some etched lines to the dialog box

---

```
CALL WIN.DBCTRL(DBX, "None", "static" , "", 140, 70,6, 12,  
"SS_ETCHEDVERT|WS_VISIBLE")  
CALL WIN.DBCTRL(DBX, "HL", "static", "", 140, 70, 41,  
12,"SS_ETCHEDHORZ|WS_VISIBLE")  
CALL WIN.DBCTRL(DBX, "VL", "static", "", 100, 58, 6,  
14,"SS_ETCHEDVERT|WS_VISIBLE")
```

---

## **Related Script Commands**

DialogBox Control

## **Version**

4.1 Original



# WIN.DBDEL

## Syntax

WIN.DBDEL ( *DLGNAME* )

## Description

This subroutine deletes a dialog box that was loaded using WIN.DBLOAD. You must use this command to delete the dialog box from memory, or the next time you try to run it within the same session, you will get an error message that the box is already shown. Use the WIN.DBEND command to close the dialog box, then WIN.DBDEL to delete it from memory.

## Parameters

The following table describes the parameters of the WIN.DBDEL command:

Parameter	Description
<i>DLGNAME</i>	The name of the dialog box to delete.

## Example

Find the following example as part of the WIN.PRODMD2 program.

---

```
*  
CALL WIN.DBEND(DBXNAME,0)  
CALL WIN.DBDEL(DBXNAME)
```

---

# WIN.DBDDTIME

## Syntax

WIN.DBDDTIME ( *DBX*, *NAME*, *X*, *Y*, *W*, *D*, *STYLE* )

## Description

This subroutine adds a date time control to a dialog box. The date time control displays a control into which provides entry of a date, a time or both.

Use WIN.DBEVENT to set up the events to be returned by the control. The initial properties of a date time control can be set with WIN.DBINIPRP. After the dialog box containing the control has been shown properties can be set and retrieved using WIN.DBSETPRP and WIN.DBGETPRP. The methods of the control can be run at this time using WIN.DBMETHOD.

For a full list of properties, methods and events see the Client scripting reference.

## Parameters

The following table describes the parameters of the WIN.DBDDTIME command:

Parameter	Description
<i>DBX</i>	This is the wIntegrate "handle" for a dialog box.
<i>NAME</i>	Controls name
<i>X</i>	Column position
<i>Y</i>	Row position
<i>W</i>	Width
<i>D</i>	Depth
<i>STYLE</i>	See the following styles table

## Values for STYLE

Value	Description
<i>DTS_CALENDAR</i>	Provide a drop down calendar

Value	Description
<i>DTS_UPDOWN</i>	Provide a spin control

## Related Subroutines

[WIN.DBINIPRP](#), [WIN.DBEVENTS](#), [WIN.DBSETPRP](#), [WIN.DBGETPRP](#),  
[WIN.DBMETHOD](#)

## Related Script Commands

DialogBox DateTime

## Version

4.1 Original

# WIN.DBEDGET

## Syntax

**WIN.DBEDGET** ( *DLG.NAME*, *NAME*, *VALUE*, *RET.MV* )

## Description

Use this routine to get data from a multi-line edit box. If the dialog box is shown (using WIN.DBSHOW), then the value returned is the current value in the dialog box. If it is not shown, the value is the value stored in memory. Carriage returns can be typed only into an edit control if the control is created as ES\_WANTRETURN. Each carriage return in the edit control is returned as a field mark (character 254), unless the RET.MV flag is true. Then each carriage return is returned as a value mark (character 253).

## Parameters

The following table describes the parameters of the WIN.DBEDGET command:

Parameter	Description
<i>DLG.NAME</i>	The name of the loaded dialog box.
<i>NAME</i>	The name of the control.
<i>VALUE</i>	The current value of the dialog box variable.
<i>RET.MV</i>	Returns multivalued

## Example

Print the lines from an edit control

---

```
CALL WIN.DBEDGET(DLG.NAME, "E1", VALUE, 0)
N = DCOUNT(VALUE, CHAR(254))
FOR J = 1 TO N
PRINT "Paragraph ":J: = ":VALUE<J>
NEXT J
```

---

## Related Subroutines

[WIN.DBGET](#), [WIN.DBEDSET](#), [WIN.DBSET](#)

# WIN.DBEDIT

## Syntax

**WIN.DBEDIT** ( *DBX*, *NAME*, *TEXT*, *X*, *Y*, *W*, *D*, *OPTS* )

## Description

This subroutine adds a box that you can use to edit or enter text to a dialog box.

See the Edit control in the Script Reference for details

## Parameters

The following table describes the parameters of the WIN.DBEDIT command:

Parameter	Description
<i>DBX</i>	This is the wIntegrate "handle" for a dialog box.
<i>NAME</i>	Specifies the edit box name.
<i>TEXT</i>	Specifies the text for the edit box caption.
<i>X</i>	Specifies the edit box column position.
<i>Y</i>	Specifies the edit box position.
<i>W</i>	Defines the edit box width. The default width is 40.
<i>D</i>	Defines the edit box depth. The default is 14.
<i>OPTS</i>	This is a dynamic array with two fields. See below.

## Fields for OPTS

Field	Description
<i>1</i>	Style. One or a combination of flags. (See Script Reference - Edit Control)
<i>2</i>	Maximum length

## Example

The following example is part of the WIN.DBDEMO program

---

```
CALL WIN.DBNEW(DBX, DLG.NAME, "Demonstration of Dialog boxes",  
10,10,200,160, ' ', ' ')  
*  
CALL WIN.DBLABEL(DBX, "Label 1", 4, 4, ' ', ' ', ' ')  
CALL WIN.DBEDIT(DBX, "E1", 40, 4, 40, ' ', ' ')  
CALL WIN.DBTEXT(DBX, "T1", 4, 24, 80, ' ', 'C')  
CALL WIN.DBCHECK(DBX, "C1", "Check one", 4, 36, 60, ' ')
```

---

## Related Subroutines

[WIN.DBNEW](#), [WIN.DBLOAD](#), [WIN.DBSET](#), [WIN.DBEVENT](#), [WIN.DBTEXT](#),  
[WIN.DBEDGET](#), [WIN.DBGET](#), [WIN.DBEDSET](#), [WIN.DBSET](#)

## Related Script Commands

DialogBox EditText, DialogBox Validate

# WIN.DBEDSET

## Syntax

**WIN.DBEDSET** ( *DLGNAME*, *NAME*, *VALUE* )

## Description

This subroutine sets the value of a multi-line edit control in a dialog box.

If the dialog box is currently shown, the dialog box display is updated. If the dialog box is not shown, the change is stored in memory to display on the dialog box when it is next shown. Both field marks (character 254) and value marks (character 253) are converted to carriage returns in the edit control.

## Parameters

The following table describes the parameters of the WIN.DBEDSET command:

Parameter	Description
<i>DLGNAME</i>	The name of the shown dialog box.
<i>NAME</i>	The control name.
<i>VALUE</i>	The new value for the control.

## Example

Put two lines of text in the control

---

```
TEXT = "This is a multi-line edit control"  
TEXT<-1> = "This is the second line"  
CALL WIN.DBEDSET(DLG.NAME, "E1", TEXT)
```

---

## Related Subroutines

[WIN.DBNEW](#), [WIN.DBLOAD](#), [WIN.DBEDIT](#), [WIN.DBSET](#), [WIN.DBEVENT](#),  
[WIN.DBTEXT](#), [WIN.DBEDGET](#), [WIN.DBGET](#), [WIN.DBSET](#)

# WIN.DBENABLE

## Syntax

**WIN.DBENABLE** ( *DLGNAME*, *NAMES*, *ENABLE* )

## Description

This subroutine enables or disables dialog box controls. When a control is enabled, the user can select it, and when a control is disabled, it displays as grey and cannot be selected.

## Parameters

The following table describes the parameters of the WIN.DBENABLE command:

Parameter	Description
<i>DLGNAME</i>	The name of the dialog box that has been with WIN.DBLOAD.
<i>NAMES</i>	The control names as a multi-field array.
<i>ENABLE</i>	Enable or disables controls. 0 = Disables controls. 1 = Enables controls.

## Example

\* Disable Id and Lookup fields

---

```
R.CTRL$=""  
R.CTRL$<-1> = "LookupButton"  
R.CTRL$<-1> = "CustId"  
CALL WIN.DBENABLE("CustMaint", R.CTRL$, 0)
```

---

## Related Subroutines

[WIN.DBENALL](#)

## Related Script Commands

DialogBox Enable



# WIN.DBENALL

## Syntax

**WIN.DBENALL** ( *DLGNAME*, *OPTS*, *ENABLE* )

## Description

This subroutine is similar to WIN.DBENABLE, but it enables or disables the entire dialog box window or all the named controls in the dialog box.

Named controls are those given a name other than "None" in the control definition.

## Parameters

The following table describes the parameters of the WIN.DBENALL command:

Parameter	Description
<i>DLGNAME</i>	The name of the dialog box that has been loaded WIN.DBLOAD.
<i>OPTS</i>	The options to enable or disable the window or controls. 0 = Enables or disables the dialog box window, 1 = Enables or disables dialog box controls.
<i>ENABLE</i>	Enable or disable controls. 0 = Disables window or controls, 1 = Enables window or controls.

## Example

Disable all controls except the ProdRef edit text

---

```
CALL WIN.DBENALL(DBXNAME, 1, FALSE)
CALL WIN.DBENABLE(DBXNAME, "ProdRef", TRUE)
```

---

## Related Subroutines

[WIN.DBENABLE](#)

## Related Script Commands

DialogBox EnableAll

# WIN.DBEND

## Syntax

**WIN.DBEND** ( *DLGNAME*, *UPDATE* )

## Description

This subroutine ends the display of a dialog box. You must use this command to remove the dialog box from the screen, and then use WIN.DBDEL to remove the dialog box from memory.

## Parameters

The following table describes the parameters of the WIN.DBEND command:

Parameter	Description
<i>DLGNAME</i>	The name of the displayed dialog box.
<i>UPDATE</i>	This variable specifies if the dialog box values are updated. If they are updated, the values from the controls are copied to the memory values. Otherwise, the memory stays the same as when the dialog was first shown. 0 = Do not update dialog variable 1 = Update dialog variable

## Example

This example is part of the WIN.DBDEMO demonstration program.

---

```
* Process dialog DlgDemo events
1000 BEGIN CASE
CASE CTRL='OK'
  CALL WIN.DBEND(DLG.NAME, TRUE)
  FINISHED=TRUE
CASE CTRL='Cancel'
  CALL WIN.DBEND(DLG.NAME, FALSE)
  FINISHED=TRUE
END CASE
* RETURN
```

---

## Related Subroutines

[WIN.DBSHOW](#), [WIN.DBDEL](#), [WIN.DBGET](#), [WIN.DBSET](#)

## Related Script Commands

DialogBox End

# WIN.DBESTACK

## Syntax

WIN.DBESTACK ( *ESTACK* )

## Description

This subroutine reads the next event from a dialog box or menu defined using the host routines and stores it in a variable for future processing by WIN.DBEVENT2.

## Parameters

The following table describes the parameters of the WIN.DBESTACK command:

Parameter	Description
<i>ESTACK</i>	Variable to store the event in

## Related Subroutines

[WIN.DBEVENT2](#)

## Version

4.0.1 Original

# WIN.DBEVENT

## Syntax

WIN.DBEVENT ( *DLGNAME*, *CTRL*, *ETYPE* )

## Description

This subroutine waits for an event and returns the event type.

If event arguments are required use WIN.DBEVENT2.

## Parameters

The following table describes the parameters of the WIN.DBEVENT command:

Parameter	Description
<i>DLGNAME</i>	The name of a dialog box, designated in WIN.DBNEW.
<i>CTRL</i>	The returned control name for an event.
<i>ETYPE</i>	See the Event Types table below.

## Values for ETYPE

The following event types are returned

Value	Description
<i>U</i>	Unknown - Unrecognised line input. The line is returned in the CTRL parameter
<i>M</i>	Menu - The menu option selected is returned
<i>D</i>	Default - The control has received the focus. This event is produced by the edittext, listbox and combobox controls.
<i>V</i>	Validate - The control has lost the input focus.
<i>C</i>	Button click - The control has been clicked.

Value	Description
<i>DC</i>	Double click - Returns that a list box with the "LBS_NOTIFY" has been double clicked Double click - A list box with the LBS_NOTIFY style has been double clicked
<i>S</i>	Scroll - A scroll bar has been scrolled
<i>number</i>	Event number set up by WIN.DBEVENTS

## Example

This example is a part of the WIN.DBDEMO demonstration program.

---

This example is a part of the WIN.DBDEMO demonstration program.

```
CALL WIN.DBEVENT(DLG,CTRL,EVENT)
*
BEGIN CASE
CASE EVENT = 'U'; PRINT 'Unknown':
CASE EVENT = 'M'; PRINT 'Menu':
CASE EVENT = 'V'; PRINT 'Validation':
CASE EVENT = 'D'; PRINT 'Default':
CASE EVENT = 'C'; PRINT 'Button Click':
CASE EVENT = 'DC'; PRINT 'Double Click':
CASE EVENT = 'S'; PRINT 'Scroll':
CASE 1
PRINT 'Event ':EVENT:
END CASE
*
IF DLG # '' THEN
PRINT ' dialog ':DLG:' Control ':CTRL
END ELSE
PRINT ' data ':CTRL
END
*
BEGIN CASE
CASE DLG = ''; IF CTRL = '*' THEN FINISHED=TRUE
CASE DLG = DLG.NAME; GOSUB 1000; * Process DlgDemo events
END CASE
*
UNTIL FINISHED DO
REPEAT
*
CALL WIN.TWCLOSE("DLGDEMO")
```

---

## **Related Subroutines**

[WIN.DBNEVENT](#), [WIN.DBFETCH](#), [WIN.DBEVENT2](#), [WIN.DBEVENTS](#)



# WIN.DBEVENT2

## Syntax

**WIN.DBEVENT2** ( *DLG.NAME*, *CTRL*, *ETYPE*, *ARGS*, *ESTACK* )

## Description

This subroutine returns the next event from a dialog box or menu defined using the host routines.

It should be used in preference to WIN.DBEVENT as it allows the return of events that return additional values.

## Parameters

The following table describes the parameters of the WIN.DBEVENT2 command:

Parameter	Description
<i>DLG.NAME</i>	The name of the dialog box the event occurred on
<i>CTRL</i>	The name of the control which generated the event
<i>ETYPE</i>	The event type. See below
<i>ARGS</i>	Comma separated list of values returned by the event
<i>ESTACK</i>	Event stack. See WIN.DBESTACK

## Values for ETYPE

Value	Description
<i>Letter</i>	See WIN.DBEVENT for a list of the standard values
<i>number</i>	Event number set up by WIN.DBEVENTS

## Related Subroutines

[WIN.DBESTACK](#), [WIN.DBEVENT](#), [WIN.DBEVENTS](#)

## **Version**

4.0.1 Original

# WIN.DBEVENTS

## Syntax

**WIN.DBEVENTS** ( *DBX*, *NAME*, *EVENTS* )

## Description

This subroutine sets up the events to be returned to the host for the dialog box controls.

These events are retrieved with WIN.DBEVENT2.

For details on the events generated see the Client scripting reference.

## Parameters

The following table describes the parameters of the WIN.DBEVENTS command:

Parameter	Description
<i>DBX</i>	The wIntegrate "handle" for the dialog box
<i>NAME</i>	The name of the control
<i>EVENTS</i>	Dynamic array of the events to be returned for this control. WIN.DBEVENT2 will return the position of the event in this array when the dialog box generates an event on this control. See below for the format of each field in the array.

## Fields for EVENTS

The events array contains the following multi-values

Field	Description
<i>1</i>	The event name
<i>2</i>	The event arguments. The name of the arguments associated with the event (if required).

## Example

Set up events for a tab control Tab1 and tree view Oak.

---

```
CALL WIN.DBEVENTS(DBX, 'Tab1', 'SelChanged')  
CALL WIN.DBEVENTS(DBX, 'Oak', 'SelChanged':AM:'EndLabelEdit')
```

---

## Related Subroutines

[WIN.DBEVENT2](#)

## Version

4.0.1 Original

# WIN.DBFETCH

## Syntax

**WIN.DBFETCH** ( *DLG*, *CTRL*, *TYPE* )

## Description

This routine checks the next event if there is one pending, or returns `TYPE = ""` if no event exists. It returns the same value as the `WIN.DBEVENT` command.

You need to use this routine after a validation event is received by the host. Then the PC sends the next event automatically. The controlling host program may accept or reject the event depending on the results of the validation. It allows the controlling host program to preserve the event that could be lost by the validation process. If the event returned by `WIN.DBFETCH` requires action, repost it to the event queue with `WIN.DBPOST` when the validation completes.

## Parameters

The following table describes the parameters of the `WIN.DBFETCH` command:

Parameter	Description
<i>DLG</i>	The name of the dialog box.
<i>CTRL</i>	The control name for the next event.
<i>TYPE</i>	The event type of the next event. See the Event table in <code>WIN.DBEVENT</code>

## Example

This example is taken from the `WIN.CUSTMD` demonstration program.

---

```
* Validation events
* These events are often immediately followed by a default for the
* Next field or a button click. So we must use WIN.DBFETCH to check
* the next event and WIN.DBPOST to re-queue it (if required).
*
300 CALL WIN.DBFETCH(NEXT.DLG, NEXT.CTRL, NEXT.ETYPE)
*
* If next event is click on cancel don't bother with validation
IF NEXT.ETYPE = "C" AND NEXT.CTRL = "Cancel" THEN
```

```
PRINT "Ignoring last event as it is followed by:-"

BEGIN CASE
CASE FLD = "CustRef"; GOSUB 1050; * Validate customer number
CASE FLD = "Name"; GOSUB 1150; * Validate Name
CASE FLD = "Street"; CALL WIN.DBGET(DBX.NAME, FLD, CUST.STREET)
CASE FLD = "City" ; CALL WIN.DBGET(DBX.NAME, FLD, CUST.CITY)
CASE FLD = 'State'; CALL WIN.DBGET(DBX.NAME, FLD, CUST.STATE)
CASE FLD = 'Zip'; CALL WIN.DBGET(DBX.NAME, FLD, CUST.ZIP)
CASE FLD = 'Tel'; CALL WIN.DBGET(DBX.NAME, FLD, CUST.TEL)
CASE FLD = 'Fax'; CALL WIN.DBGET(DBX.NAME, FLD, CUST.FAX)
CASE FLD = 'Salesman'; CALL WIN.DBGET(DBX.NAME, FLD, CUST.SALESMAN)
CASE FLD = 'Comment'; CALL WIN.DBGET(DBX.NAME, FLD, CUST.COMMENT)
CASE FLD = "Discount"; GOSUB 1250; * Validate discount
END CASE
IF NEXT.ETYPE # "" THEN CALL WIN.DBPOST(NEXT.DLG, NEXT.CTRL,
NEXT.ETYPE)
END
*
RETURN
*
```

---

## Related Subroutines

[WIN.DBEVENT](#)

# WIN.DBFOCUS

## Syntax

WIN.DBFOCUS ( *DLGNAME*, *NAME* )

## Description

This subroutine sets the input focus to a dialog box control.

## Parameters

The following table describes the parameters of the WIN.DBFOCUS command:

Parameter	Description
<i>DLGNAME</i>	The name of the dialog box, set in WIN.DBNEW.
<i>NAME</i>	The name of the control.

## Example

This example is a part of the WIN.PRODMD2 demonstration program.

```
* Enable/Disable fields
* For first input only ProdRef and cancel button are enabled
* While amending, everything is enabled except ProdRef
* Also modify cancel button caption to specify its function
5150 CALL WIN.DBENALL(DBXNAME,1, AMENDING)
CALL WIN.DBENABLE(DBXNAME, "ProdRef", NOT(AMENDING))
*
IF AMENDING THEN
    CALL WIN.DBFOCUS(DBXNAME, "Name")
    CALL WIN.DBCAPT(DBXNAME, "Cancel", "Void")
END ELSE
    * Always enable Cancel
    CALL WIN.DBENABLE(DBXNAME, "Cancel", TRUE)
    CALL WIN.DBCAPT(DBXNAME, "Cancel", "Exit")
    CALL WIN.DBFOCUS(DBXNAME, "ProdRef")
END
*
RETURN
```

## Related Subroutines

[WIN.DBSHOW](#), [WIN.DBEVENT](#)



# WIN.DBGET

## Syntax

**WIN.DBGET** ( *DLG.NAME*, *NAME*, *VALUE* )

## Description

This subroutine gets the current value of a control.

If the dialog box is shown by WIN.DBSHOW, then the value returned is the current value in the dialog box. Otherwise the value is the value stored in memory.

Use WIN.DBEDGET to return values from a multi-lined edit control.

## Parameters

The following table describes the parameters of the WIN.DBGET command:

Parameter	Description
<i>DLG.NAME</i>	The name of the loaded dialog box.
<i>NAME</i>	The name of the control
<i>VALUE</i>	The value of the dialog box variable.

## Example

This example is part of the WIN.DBDEMO demonstration program.

```
This example is part of the WIN.DBDEMO demonstration program.
* Display result of dialog
1100 PRINT @(0,3):@(-3):"You clicked OK or pressed return to end the
dialog"
PRINT
PRINT 'On the dialog you had set:-'
CALL WIN.DBGET(DLG.NAME, "E1", VALUE)
PRINT 'The Edit field to ":VALUE:'
CALL WIN.DBGET(DLG.NAME, "C1", VALUE)
PRINT "The check box was ":
IF VALUE THEN PRINT "Checked" ELSE PRINT "Unchecked"
CALL WIN.DBGET(DLG.NAME, "G1", VALUE)
PRINT 'You selected radio option ':
*
```

```
BEGIN CASE
  CASE VALUE = "R1"; PRINT "Radio one":
  CASE VALUE = "R2"; PRINT "Radio two":
  CASE VALUE = "R3"; PRINT "Radio three":
END CASE
PRINT ''
*
CALL WIN.DBGET(DLG.NAME, "L1", VALUE)
PRINT 'List box option ": VALUE : ''
CALL WIN.DBGET(DLG.NAME, "CB1", VALUE)
PRINT 'Combo box value ": VALUE : ''
CALL WIN.DBGET(DLG.NAME, "S1", VALUE)
PRINT 'Scroll bar position ": VALUE : ''
```

---

## Related Subroutines

[WIN.DBSET](#), [WIN.DBEDGET](#), [WIN.DBGETPRP](#), [WIN.DBGETM](#)

# WIN.DBGETM

## Syntax

**WIN.DBGETM** ( *DLG.NAME*, *NAMES*, *VALUES* )

## Description

The subroutine gets the value of multiple controls in a dialog box in one call. Using this routines is faster than multiple calls to WIN.DBGET.

If the dialog box is displayed the values returned are the current values in the dialog box, otherwise the value is the value stored in memory.

Use WIN.DBGET to get the value of an individual control.

## Parameters

The following table describes the parameters of the WIN.DBGETM command:

Parameter	Description
<i>DLG.NAME</i>	The name of the loaded dialog box.
<i>NAMES</i>	A dynamic array with the names of the controls to return the values for. The name can also be specified as "control_name.property_name" (i.e. the property name added to the control name with a period to separate them) to return the value for a property of the control.
<i>VALUES</i>	The variable to receive the control/property values. This is a dynamic array matched to the NAMES parameter. cr (char 13) is converted to a value mark and tab (char 9) is converted to a sub value mark in the returned field.

## Example

Using WIN.DBGETM instead of WIN.DBGET in the WIN.DBDEMO program

---

```
* Get values from the dialog box used in WIN.DBDEMO using WIN.DBGETM
1130 VALUES = ""
NAMES = "E1"
NAMES<2> = "C1"
NAMES<3> = "G1"
NAMES<4> = "L1"
NAMES<5> = "CB1"
NAMES<6> = "S1"
CALL WIN.DBGETM(DLG.NAME, NAMES, VALUES)
*
PRINT 'The Edit field to "':VALUES<1>:''
PRINT "The check box was ":
IF VALUES<2> THEN PRINT "Checked" ELSE PRINT "Unchecked"
PRINT 'You selected radio option "':
*
BEGIN CASE
CASE VALUES<3> = "R1"; PRINT "Radio one":
CASE VALUES<3> = "R2"; PRINT "Radio two":
CASE VALUES<3> = "R3"; PRINT "Radio three":
END CASE
PRINT '""
*
PRINT 'List box option "': VALUES<4> : '""
PRINT 'Combo box value "': VALUES<5> : '""
PRINT 'Scroll bar position "': VALUES<6> : '""
RETURN
```

---

## Related Subroutines

[WIN.DBGET](#), [WIN.DBEDGET](#), [WIN.DBGETPRP](#), [WIN.DBSET](#)

## Version

5.1.2 Added

# WIN.DBGETPRP

## Syntax

**WIN.DBGETPRP** ( *DLG.NAME*, *NAME*, *PROPERTY*, *VALUE* )

## Description

This subroutine gets the value for a controls property after it has been loaded to the PC with WIN.DBLOAD.

For a full list of properties for each control see the Client scripting reference.

## Parameters

The following table describes the parameters of the WIN.DBGETPRP command:

Parameter	Description
<i>DLG.NAME</i>	The name of the loaded dialog box
<i>NAME</i>	Controls name
<i>PROPERTY</i>	Name of the property to set
<i>VALUE</i>	The value returned for the property

## Related Subroutines

[WIN.DBSETPRP](#), [WIN.DBGET](#), [WIN.DBEDGET](#)

## Version

4.0.1 Original

# WIN.DBGRID

## Syntax

**WIN.DBGRID** ( *DBX*, *NAME*, *X*, *Y*, *W*, *D*, *STYLE* )

## Description

This subroutine adds a grid control to a dialog box. The grid control displays a grid of cells. The data cells start at row 1 column 1. You can change the header information for each row or column by modifying row 0 or column 0.

Use WIN.DBEVENT to set up the events to be returned by the grid. The initial properties of a grid can be set with WIN.DBINIPRP. After the dialog box containing the grid has been shown properties can be set and retrieved using WIN.DBSETPRP and WIN.DBGETPRP. The methods of the grid can be run at this time using WIN.DBMETHOD.

For a full list of properties, methods and events see the Client scripting reference.

## Parameters

The following table describes the parameters of the WIN.DBGRID command:

Parameter	Description
<i>DBX</i>	This is the wIntegrate "handle" for a dialog box.
<i>NAME</i>	Controls name
<i>X</i>	Column position
<i>Y</i>	Row position
<i>W</i>	Width
<i>D</i>	Depth
<i>STYLE</i>	"" or combination of WS_TABSTOP and WS_DISABLED

## Related Subroutines

[WIN.DBEVENTS](#), [WIN.DBINIPRP](#), [WIN.DBSETPRP](#), [WIN.DBGETPRP](#),

WIN.DBMETHOD

## **Related Script Commands**

DialogBox Grid

## **Version**

4.0.1 Original

# WIN.DBGROUP

## Syntax

**WIN.DBGROUP** ( *DBX*, *NAME*, *TEXT*, *X*, *Y*, *W*, *D* )

## Description

This subroutine adds a rectangle group box to a dialog box. A group box contains a group of controls that logically belong together.

Use a group box to group radio buttons. WIN.DBGROUP must be used before WIN.DBRADIO to create a box with radio buttons.

## Parameters

The following table describes the parameters of the WIN.DBGROUP command:

Parameter	Description
<i>DBX</i>	This is the wIntegrate "handle" for a dialog box.
<i>NAME</i>	The name of the groupbox.
<i>TEXT</i>	The text for the group box caption.
<i>X</i>	Specifies the group box column position.
<i>Y</i>	Specifies the group box position.
<i>W</i>	Defines the group box width.
<i>D</i>	Defines the group box depth.

## Example

This example is part of the WIN.DBDEMO demonstration program.

```
* The group box is the part of the dialog box that contains the radio buttons.
```

```
*
```

```
CALL WIN.DBGROUP(DBX,"G1","Group box",4,50,60,52)
CALL WIN.DBRADIO(DBX,"R1","Radio One",6,58,50,'',1)
CALL WIN.DBRADIO(DBX,"R2","Radio Two",6,72,50,'',0)
CALL WIN.DBRADIO(DBX,"R3","Radio Three",6,88,50,'',0)
```

```
*
```



## **Related Subroutines**

[WIN.DBNEW](#), [WIN.DBLOAD](#), [WIN.DBCAPT](#), [WIN.DBEVENT](#), [WIN.DBRADIO](#)

# WIN.DBHEADER

## Syntax

**WIN.DBHEADER** ( *DBX*, *NAME*, *X*, *Y*, *W*, *D*, *STYLE* )

## Description

This subroutine adds a header control to a dialog box.

Use WIN.DBEVENT to set up the events to be returned by the control. The initial properties of a control can be set with WIN.DBINIPRP. After the dialog box containing the control has been shown properties can be set and retrieved using WIN.DBSETPRP and WIN.DBGETPRP. The methods of the control can be run at this time using WIN.DBMETHOD.

For a full list of properties, methods and events see the Client scripting reference.

## Parameters

The following table describes the parameters of the WIN.DBHEADER command:

Parameter	Description
<i>DBX</i>	This is the wIntegrate "handle" for a dialog box.
<i>NAME</i>	Controls name
<i>X</i>	Column position
<i>Y</i>	Row position
<i>W</i>	Width
<i>D</i>	Depth
<i>STYLE</i>	See the Client Scripting Reference

## Related Subroutines

[WIN.DBINIPRP](#), [WIN.DBEVENTS](#), [WIN.DBSETPRP](#), [WIN.DBGETPRP](#),  
[WIN.DBMETHOD](#)

## **Related Script Commands**

DialogBox Header

## **Version**

4.0.1 Original

# WIN.DBIMAGE

## Syntax

**WIN.DBIMAGE** ( *DBX*, *NAME*, *FILE*, *X*, *Y*, *W*, *D* )

## Description

This subroutine adds an image to a dialog box. You can use this subroutine if you want to add a graphic to a dialog box to illustrate a point or to add a logo.

Refer to the table in this section listing the image formats supported by wIntegrate. You must include the full path of the image if it does not reside in the wintegrate directory.

## Parameters

The following table describes the parameters of the WIN.DBIMAGE command:

Parameter	Description
<i>DBX</i>	This is the wIntegrate "handle" for a dialog box.
<i>NAME</i>	Specifies the image name.
<i>FILE</i>	Specifies the PC file name of the image.
<i>X</i>	Specifies the image column position.
<i>Y</i>	Specifies the image row position.
<i>W</i>	Defines the image width.
<i>D</i>	Defines the image depth.

## Values for FILE

This table details the image formats that are supported

Value	Description
<i>.ico</i>	Windows icon format
<i>.bmp</i>	Windows bitmap format
<i>.wmf</i>	Windows metafile format

Value	Description
<i>.jpg</i>	Joint Photographic Expert Group
<i>.pcx</i>	ZSoft image format

## Example

The following example is taken from the WIN.DBDEMO program.

---

```
The following example is taken from the WIN.DBDEMO program.  
CALL WIN.DBLIST(DBX, "L1", 68,52,50,52, "WS_BORDER | WS_TABSTOP |  
LBS_NOTIFY")  
CALL WIN.DBCOMBO(DBX, "CB1", 122,52,50,52, '')  
CALL WIN.DBSCROLL(DBX, "S1", 4,116,60, '', 'WS_TABSTOP', '', '')  
CALL WIN.DBIMAGE(DBX, "I1", "image\computer.wmf", 68,116,100,40)
```

---

## Related Subroutines

[WIN.DBNEW](#), [WIN.DBLOAD](#), [WIN.DBEVENT](#), [WIN.IMAGE](#)

## Related Script Commands

DialogBox Graphic

# WIN.DBIMBUT

## Syntax

**WIN.DBIMBUT** ( *DBX*, *NAME*, *FILE*, *X*, *Y*, *W*, *D* )

## Description

This subroutine adds an image as a push button to a dialog box. To add functionality to the graphic, you must use the WIN.DBEVENT subroutine.

## Parameters

The following table describes the parameters of the WIN.DBIMBUT command:

Parameter	Description
<i>DBX</i>	This is the wIntegrate "handle" for a dialog box.
<i>NAME</i>	Specifies the image name.
<i>FILE</i>	Specifies the PC file name of the image.
<i>X</i>	Specifies the image column position.
<i>Y</i>	Specifies the image row position.
<i>W</i>	Defines the image width.
<i>D</i>	Defines the image depth.

## Example

The following example is part of the WIN.DBDEMO demonstration program.

The following example is part of the WIN.DBDEMO demonstration program.  
`CALL WIN.DBIMBUT(DBX,"IB","image\normal.bmp",158,36,' ','')`

## Related Subroutines

[WIN.DBNEW](#), [WIN.DBLOAD](#), [WIN.DBEVENT](#), [WIN.DBIMAGE](#), [WIN.IMAGE](#)

## **Related Script Commands**

DialogBox Graphic, DialogBox GraphicButton

# WIN.DBINIPRP

## Syntax

**WIN.DBINIPRP** ( *DBX*, *NAME*, *PROPERTY*, *VALUE* )

## Description

This subroutine sets the initial value for a controls property when it is loaded to the PC with WIN.DBLOAD.

For a full list of properties for each control see the Client scripting reference.

## Parameters

The following table describes the parameters of the WIN.DBINIPRP command:

Parameter	Description
<i>DBX</i>	This is the wIntegrate "handle" for a dialog box.
<i>NAME</i>	Controls name
<i>PROPERTY</i>	Name of the property to set
<i>VALUE</i>	New value of the property

## Related Subroutines

[WIN.DBSETPRP](#)

## Version

4.0.1 Original



# WIN.DBINIT

## Syntax

**WIN.DBINIT** ( *DBX*, *INIT.CMND* )

## Description

Use this subroutine to assign a script command to be run whenever a dialog box is shown.

## Parameters

The following table describes the parameters of the WIN.DBINIT command:

Parameter	Description
<i>DBX</i>	The wIntegrate "handle" to the dialog box being build
<i>INIT.CMND</i>	The script text to be executed when the dialog is shown

## Related Script Commands

DialogBox InitCommand

# WIN.DBINPOK

## Syntax

**WIN.DBINPOK** ( *DBXNAME*, *OK*, *WIN.DBINPOK* )

## Description

This subroutine verifies the input after a validate event.

## Parameters

The following table describes the parameters of the WIN.DBINPOK command:

Parameter	Description
<i>DBXNAME</i>	The name of the dialog box.
<i>OK</i>	Returns whether input was correct. 0 = False 1 = True
<i>WIN.DBINPOK</i>	Parameters

## Example

This example is a part of the WIN.PRODMD2 example program located in WIN.PROGS directory.

```
* Cancel was clicked, all we need to do is skip the fields case
statement
* as the event will arrive in the main loop normally
  PRINT "Cancel clicked, skipping validation"
END ELSE
  BEGIN CASE
    CASE ECTRL = "ProdRef"; GOSUB 1000; * Read product record
    CASE ECTRL = "Name"; GOSUB 1100; * Validate name
    CASE ECTRL = "Price"; GOSUB 1300; * Validate price
  END CASE
END
*
IF ERRMSG = "" THEN
  CALL WIN.DBINPOK(DBXNAME, TRUE)
END ELSE
  DUM = ""
  CALL WIN.DBMSGBOX(DBXNAME, ERRMSG, "Product Maintenance",
"MB_ICONHAND|MB_OK", DUM)
```

```
CALL WIN.DBINPOK(DBXNAME, FALSE)
END
*
```

---

# WIN.DBLABEL

## Syntax

**WIN.DBLABEL** ( *DBX*, *TEXT*, *X*, *Y*, *W*, *D*, *OPTIONS* )

## Description

This subroutine adds a label to a dialog box. Use this when you want to display the name of a text, edit, or list box.

## Parameters

The following table describes the parameters of the WIN.DBLABEL command:

Parameter	Description
<i>DBX</i>	This is the wIntegrate "handle" for a dialog box.
<i>TEXT</i>	The text to display
<i>X</i>	Column position
<i>Y</i>	Row position
<i>W</i>	Width. The default is the length of the text multiplied by 4
<i>D</i>	Depth. The default depth is 12
<i>OPTIONS</i>	See the following options table

## Fields for OPTIONS

The options are specified in a dynamic array of the following fields:

Field	Description
<i>1</i>	Alignment: L - Left, C - Center, R - Right
<i>2</i>	Fixed number of dialog units to add to the width
<i>3</i>	Put the label in a sunken rectangle if set to 1

## Example

### Set up a normal and sunken label

---

```
* Set up a normal and sunken label
SINK.TEXT = "";SINK.TEXT<3> = 1; * Sunken text option for text and
labels
CALL WIN.DBLABEL(DBX, "This is a normal label",5,8,78,12,"")
CALL WIN.DBLABEL(DBX, "This is a sunken label",141,8,78,12,SINK.TEXT)
```

---

### Add two dialog units to the width of the label

---

```
OPTS = "L"
OPTS<2> = 2
CALL WIN.DBLABEL(DBX, "Customer", 10, 20, "", "", OPTS)
```

---

### The following example is part of the WIN.DBDEMO program

---

```
* Create and show dialog
100 DBX=' '
DLG.NAME="DemoDlg"
CALL WIN.TWMSG("Creating Dialog ":DLG.NAME:"...", "Red", "Yellow",
"DOUBLE")
*
CALL WIN.DBNEW(DBX, DLG.NAME, "Demonstration of Dialog boxes",
10,10,200,160,
'', '')
*
CALL WIN.DBLABEL(DBX,"Label 1",4,4,','', '')
CALL WIN.DBEDIT(DBX,"E1",40,4,40,','', '')
CALL WIN.DBTEXT(DBX,"T1",4,24,80,','', 'C')
CALL WIN.DBCHECK(DBX,"C1","Check one",4,36,60, '')
```

---

## Related Subroutines

[WIN.DBNEW](#)

## Related Script Commands

DialogBox LText

## **Version**

4.1.1 Added Sunken option

# WIN.DBLIST

## Syntax

**WIN.DBLIST** ( *DBX*, *NAME*, *X*, *Y*, *W*, *D*, *STYLE* )

## Description

This subroutine adds a list box to a dialog box. A list box is a rectangular area that contains a list of selectable options. Set the initial data for the list box with WIN.DBSET, and select an option from the initial list with WIN.DBSELECT. To set tab stops within a list box, use the WIN.DBTABS subroutine.

See the listbox control in the Script Reference for details.

## Parameters

The following table describes the parameters of the WIN.DBLIST command:

Parameter	Description
<i>DBX</i>	This is the wIntegrate "handle" for a dialog box.
<i>NAME</i>	The name for the list box.
<i>X</i>	Specifies the list box column position.
<i>Y</i>	Specifies the list box row position.
<i>W</i>	Defines the list box width.
<i>D</i>	Defines the list box depth.
<i>STYLE</i>	Specifies the style of the list box. The default is "LBS_NOTIFY WS_BORDER". (See the Script Reference for a full list of styles)

## Example

This example is a part of the WIN.DBDEMO demonstration program.

```
* The list box is created with WIN.DBLIST, then populated with data
later
* in the line "CALL WIN.DBSET(DLG.NAME,"L1", R.LIST).
CALL WIN.DBLIST(DBX, "L1",68,52,50,52,"WS_BORDER | WS_TABSTOP |
LBS_NOTIFY")
```

```
CALL WIN.DBCOMBO(DBX,"CB1",122,52,50,52,'')
CALL WIN.DBSCROLL(DBX,"S1", 4,116,60,'','WS_TABSTOP','')
CALL WIN.DBIMAGE(DBX,"I1", "image\computer.wmf",68,116,100,40)
.
.
.
*
IF ERR = 0 THEN
CALL WIN.DBSET(DLG.NAME,"T1","Centered Text")
CALL WIN.DBSET(DLG.NAME,"R1",1)
R.LIST = "Option 1"
R.LIST<-1>="Option 2"
R.LIST<-1>="Option 3"
CALL WIN.DBSET(DLG.NAME,"L1", R.LIST)
CALL WIN.DBSELECT(DLG.NAME,"L1", "Option 2")
CALL WIN.DBSET(DLG.NAME,"CB1",R.LIST)
CALL WIN.DBSELECT(DLG.NAME,"CB1", "Option 3")
END
```

---

## Related Subroutines

[WIN.DBNEW](#), [WIN.DBLOAD](#), [WIN.DBSET](#), [WIN.DBSELECT](#), [WIN.DBEVENT](#),  
[WIN.DB TABS](#), [WIN.LOOKUP](#)

## Related Script Commands

DialogBox Listbox, DialogBox Validate, DialogBox ControlCommand



# WIN.DBLISTVW

## Syntax

**WIN.DBLISTVW** ( *DBX*, *NAME*, *X*, *Y*, *W*, *D*, *STYLE* )

## Description

This subroutine adds a list view control that displays a list of items with optional images and additional information

Use WIN.DBEVENT to set up the events to be returned by the control. The initial properties of a control can be set with WIN.DBINIPRP. After the dialog box containing the control has been shown properties can be set and retrieved using WIN.DBSETPRP and WIN.DBGETPRP. The methods of the control can be run at this time using WIN.DBMETHOD.

For a full list of properties, methods and events see the Client scripting reference.

## Parameters

The following table describes the parameters of the WIN.DBLISTVW command:

Parameter	Description
<i>DBX</i>	This is the wIntegrate "handle" for a dialog box.
<i>NAME</i>	Controls name
<i>X</i>	Column position
<i>Y</i>	Row position
<i>W</i>	Width
<i>D</i>	Depth
<i>STYLE</i>	See the Client Scripting Reference

## Related Subroutines

[WIN.DBINIPRP](#), [WIN.DBEVENTS](#), [WIN.DBSETPRP](#), [WIN.DBGETPRP](#),  
[WIN.DBMETHOD](#)

## Related Script Commands

DialogBox ListView

## Version

4.0 Original

# WIN.DBLOAD

## Syntax

**WIN.DBLOAD** ( *DBX*, *ERR* )

## Description

This subroutine loads a dialog box. It must be used to load the dialog box to the PC in order to display it on the wIntegrate screen with WIN.DBSHOW. To remove the dialog box, use WIN.DBDEL.

## Parameters

The following table describes the parameters of the WIN.DBLOAD command:

Parameter	Description
<i>DBX</i>	This is the wIntegrate "handle" for a dialog box.
<i>ERR</i>	The error codes returned. 0 = No error 1 = The dialog definition is missing a name 2 = The dialog box is shown 3 = The dialog box is loaded 4 = Unable to create the dialog box

## Example

The following example is a part of the WIN.DBDEMO demonstration program.

```
CALL WIN.DBLOAD(DBX,ERR)
IF ERR THEN
ERR.MSG='Loading with WIN.DBLOAD'
* Just delete dialog and try again if we get already exist error
IF ERR = 3 THEN
CALL WIN.DBDEL(DLG.NAME)
CALL WIN.DBLOAD(DBX,ERR)
END
END
```

## Related Subroutines

[WIN.DBNEW](#), [WIN.DBDEL](#), [WIN.DBSHOW](#)

# WIN.DBMETHOD

## Syntax

**WIN.DBMETHOD** ( *DLG.NAME*, *NAME*, *METHOD*, *ARG.FLAGS*, *MAT ARGS*, *RET.FLAG*, *RET.VALUE* )

## Description

This subroutine runs a method on a control. The dialog box containing the control must have been shown (e.g. with WIN.DBSHOW).

For a full list of methods for each control see the Client scripting reference.

## Parameters

The following table describes the parameters of the WIN.DBMETHOD command:

Parameter	Description
<i>DLG.NAME</i>	The name of the loaded dialog box
<i>NAME</i>	Controls name
<i>METHOD</i>	The name of the method to run
<i>ARG.FLAGS</i>	String describing the arguments. See below
<i>MAT ARGS</i>	20 element array to hold the arguments
<i>RET.FLAG</i>	Flag to specify if the method returns a value
<i>RET.VALUE</i>	Value returned from the method if RET.FLAG is set to 1.

## Values for ARG.FLAGS

The ARG.FLAGS is a string of the following letters, one for each argument for the method

Value	Description
<i>I</i>	Input argument. The value from the corresponding element of ARG.S is used to run the method

Value	Description
<i>O</i>	Output argument. The value from the corresponding element of ARGS is set to the value returned from the method for this argument
<i>B</i>	Both ways. The value from the corresponding element of ARGS is used to run the method and the value of the argument is returned after the method has run

## Example

### Fill a range of cells in a grid

---

```
* Fill a range of cells use the FillRange method (find it in the
client scripting reference).
ARGS(1) = 1 ;* start row
ARGS(2) = 1 ;* start column
ARGS(3) = 4 ;* end row
ARGS(4) = 6 ;* end column
ARGS(5) = "Hello" ;* Fill with the word hello
CALL WIN.DBMETHOD(MY.DLGNAME, MY.GRID, "FillRange", "IIIII",MAT ARGS,0,
RET.VAL)
* Get the background colour of a cell 2,2
ARGS(1) = 2
ARGS(2) = 2
CALL WIN.DBMETHOD(MY.DLGNAME, MY.GRID,
"GetCellBackColor","II",MATARGS, 1, BACK.COLOR)
```

---

### Fill a tree view control with some dummy values

---

```
BRANCH = ''
RESP = ''
FOR J = 1 TO 10
    METHOD.ARGS(1) = "Root Branch ":J
    CALL WIN.DBMETHOD(NAME, "TV1", "InsertLine", "I", MAT
METHOD.ARGS, 1, BRANCH)
    FOR K = 1 TO J
        METHOD.ARGS(1) = "Sub branch ":K
        METHOD.ARGS(2) = 0
        METHOD.ARGS(3) = 0
        METHOD.ARGS(4) = BRANCH
        CALL WIN.DBMETHOD(NAME, "TV1", "InsertLine", "IIII", MAT
```

```
METHOD.ARGS, 0, RESP)  
    NEXT K  
NEXT J
```

---

See also WIN.DBDEMO3

## Related Subroutines

[WIN.DBSETPRP](#), [WIN.DBGETPRP](#)

## Version

4.0.1 Original

# WIN.DBMNSIZE

## Syntax

**WIN.DBMNSIZE** ( *DLGNAME*, *WIDTH*, *DEPTH* )

## Description

This subroutine specifies the minimum size a dialog box can be resized to.

The unit of measure for the *WIDTH* and *DEPTH* parameters is in dialog box units. This is calculated as 1/4 of the average width and 1/8 of the average depth of the dialog box font.

## Parameters

The following table describes the parameters of the WIN.DBMNSIZE command:

Parameter	Description
<i>DLGNAME</i>	The name of the dialog box
<i>WIDTH</i>	The minimum width the dialog box can be resized to. Use -1 to use the default
<i>DEPTH</i>	The minimum depth the dialog box can be resized to. Use -1 to use the default

## Related Subroutines

[WIN.DBPANEL](#)

## Related Script Commands

DialogBox MinSize

## Version

4.2.1 Original

# WIN.DBMOVE

## Syntax

WIN.DBMOVE ( *DLGNAME*, *X*, *Y* )

## Description

This subroutine moves the position of a dialog box

## Parameters

The following table describes the parameters of the WIN.DBMOVE command:

Parameter	Description
<i>DLGNAME</i>	The name of the dialog box to move
<i>X</i>	The new X-coordinate for the dialog box
<i>Y</i>	The new Y-coordinate for the dialog box

## Related Script Commands

DialogBox Move

## Version

4.0.2 Original



# WIN.DBMSGBOX

## Syntax

**WIN.DBMSGBOX** ( *NAME*, *TEXT*, *TITLE*, *FLAGS*, *RESP* )

## Description

This subroutine displays a message box that is attached to a dialog box. The message box stops all data from going to the dialog box until the message box is closed. If you want a message box that is not attached to a dialog, see WIN.MSGBOX.

## Parameters

The following table describes the parameters of the WIN.DBMSGBOX command:

Parameter	Description
<i>NAME</i>	The name of the dialog box that the message box be attached to.
<i>TEXT</i>	The text to display in the message box.
<i>TITLE</i>	The title for the message box. If you do not specify, default is “User Message”.
<i>FLAGS</i>	Icon and Response buttons to display in the box. the Icon Flag and Response Button tables in section.)
<i>RESP</i>	The response from the user.

## Values for FLAGS

Use only one button and one icon flag

Value	Description
<i>MB_OK</i>	Displays an “OK” button.
<i>MB_OKCANCEL</i>	Displays “OK” and “Cancel” buttons.
<i>MB_ABORTRETRYIGNORE</i>	Displays “Abort”, “Retry”, and “Ignore” buttons.

Value	Description
<i>MB_YESNOCANCEL</i>	Displays “Yes”, “No”, and “Cancel” buttons.
<i>MB_YESNO</i>	Displays “Yes” and “No” buttons.
<i>MB_RETRYCANCEL</i>	Displays “Retry” and “Cancel” buttons.
<i>MB_DEFBUTTON1</i>	Makes the first button the default button.
<i>MB_DEFBUTTON2</i>	Makes the second button the default button.
<i>MB_DEFBUTTON3</i>	Makes the third button the default button.
<i>MB_ICONHAND</i>	Displays the stop sign icon.
<i>MB_ICONQUESTION</i>	Displays the question icon.
<i>MB_ICONEXCLAMATION</i>	Displays the exclamation warning icon.
<i>MB_ICONASTERISK</i>	Displays the asterisk information icon.

## Example

This example is part of the WIN.PRODMD2 program located in the WIN.PROGS directory.

---

```
*
IF ERRMSG = " " THEN
    CALL WIN.DBINPOK(DBXNAME, TRUE)
END ELSE
    DUM = " "
    CALL WIN.DBMSGBOX(DBXNAME, ERRMSG, "Product Maintenance",
"MB_ICONHAND|MB_OK", DUM)
    CALL WIN.DBINPOK(DBXNAME, FALSE)
END
```

---

## Related Subroutines

[WIN.MSGBOX](#), [WIN.DBATTACH](#)

## Related Script Commands

MessageBox

# WIN.DBMSIZE

## Syntax

**WIN.DBMSIZE** ( *DLGNAME*, *WIDTH*, *DEPTH*, *XSTEP*, *YSTEP* )

## Description

This subroutine defines the maximum size of a dialog box and determines the increment rate of a scroll bar in a large dialog box.

## Parameters

The following table describes the parameters of the WIN.DBMSIZE command:

Parameter	Description
<i>DLGNAME</i>	The name of the dialog box.
<i>WIDTH</i>	The maximum width of the dialog box. The allowable is 16383.
<i>DEPTH</i>	The maximum depth of the dialog box. The allowable is 16383.
<i>XSTEP</i>	Specifies the number of rows to scroll in each step.
<i>YSTEP</i>	Specifies the number of columns to scroll in scroll step.

## Example

Set maximum size of the CustMaint dialog box

---

```
CALL WIN.DBNEW(DBX, "CustMaint", "Customer Maintenance", 0, 0, 200,
100,
"WS_THICKFRAME | WS_CAPTION | WS_HSCROLL | WS_VSCROLL", "")
... add controls to dialog box
CALL WIN.DBLOAD(DBX, ERR);* Load customer maintenance dialog box
* Set up full page size to be 4 times that of initial view with
* arrows scrolling 1/10 of the size on each click
CALL WIN.DBMSIZE("CustMaint", 400, 200, 40, 20)
```

---

## **Related Subroutines**

[WIN.DBNEW](#)

## **Related Script Commands**

DialogBox MaxSize

# WIN.DBMVCTRL

## Syntax

WIN.DBMVCTRL ( *DLGNAME*, *NAME*, *X*, *Y*, *W*, *D*, *DIALOG.UNITS* )

## Description

This subroutine moves a control on a dialog box which has been shown.

## Parameters

The following table describes the parameters of the WIN.DBMVCTRL command:

Parameter	Description
<i>DLGNAME</i>	The name of the dialog box containing the control
<i>NAME</i>	Controls name
<i>X</i>	New column position. Set to "" to leave unchanged
<i>Y</i>	New row position. Set to "" to leave unchanged
<i>W</i>	New width. Set to "" to leave unchanged
<i>D</i>	New depth. Set to "" to leave unchanged
<i>DIALOG.UNITS</i>	0 - Dimensions are in pixels, 1 - dimensions are in dialog units

## Version

4.1 Original

# WIN.DBNEVENT

## Syntax

WIN.DBNEVENT ( *DBXNAME*, *CTRL*, *ETYPE* )

## Description

This subroutine returns the next event from a dialog box during validation of a field if the dialog box options are set to V1 or V2 with WIN.DBOPTION. If there is no event, the ETYPE and CTRL are set to "" (null).

## Parameters

The following table describes the parameters of the WIN.DBNEVENT command:

Parameter	Description
<i>DBXNAME</i>	The name of the dialog box to check.
<i>CTRL</i>	The returned control name for an event.
<i>ETYPE</i>	See the Event Types table below.

## Values for ETYPE

Value	Description
<i>D</i>	Default - The control that has the input focus. This event is produced edit text, list box, and combo box controls.
<i>V</i>	Validation - The control that has lost input focus. This event is produced edit text, list box, and combo box controls.
<i>C</i>	Button Click - Returns that one of the following has been clicked on: push button, radio button, check box, image or image button.
<i>L</i>	List box double click - Returns a list box double click or a combo box selection.

## Example

The following example is code from the WIN.PRODMD2 demonstration program.

---

```
* Use WIN.DBNEVENT to skip validation if cancel button is clicked.
NCTRL = ""
NTYPE = ""
CALL WIN.DBNEVENT(DBXNAME, NCTRL, NTYPE)
IF NCTRL = "Cancel" AND NTYPE = "C" THEN
* Cancel was clicked, all we need to do is skip the fields case
statement
* as the event will arrive in the main loop normally
PRINT "Cancel clicked, skipping validation"
END ELSE
```

---

## Related Subroutines

[WIN.DBEVENT](#), [WIN.DBFETCH](#)

## Related Script Commands

DialogNextEvent

# WIN.DBNEW

## Syntax

**WIN.DBNEW** ( *DBX*, *NAME*, *TITLE*, *X*, *Y*, *W*, *D*, *STYLE*, *OPTS* )

## Description

This subroutine creates a new dialog box. You must use this command to create a dialog box. This command starts the definition of a dialog box. You must then use controls within the box to create edit, text boxes, labels, etc. (See WIN.DBEDIT, WIN.DBTEXT, and WIN.DBLABEL.) Then load the new dialog box using WIN.DBLOAD, show the box with WIN.DBSHOW, and monitor events in a loop using WIN.DBEVENT. To close the dialog box, use WIN.DBEND, then delete the box from memory with WIN.DBDEL.

For step-by-step instructions on how to create dialog boxes, see "Appendix A - Using Host Subroutines".

## Parameters

The following table describes the parameters of the WIN.DBNEW command:

Parameter	Description
<i>DBX</i>	This is the wIntegrate "handle" for a dialog box.
<i>NAME</i>	The name of the new dialog box. This name is how the dialog is referenced in other dialog box calls.
<i>TITLE</i>	The title for the new dialog box. The title appears in the title bar the dialog box, and can be the same as the name.
<i>X</i>	Specifies the dialog box column position.
<i>Y</i>	Specifies the dialog box row position.
<i>W</i>	Defines the dialog box width.
<i>D</i>	Defines the dialog box depth



Parameter	Description
<i>STYLE</i>	The style for the dialog box. Extended styles can be specified by adding a second field to this variable. See the Script Reference for style and extended style names.
<i>OPTS</i>	Options for the dialog box

## Fields for OPTS

The following fields may be set in the options

Field	Description
<i>1.1</i>	Font size. If OPTS<1> = "" this default to 8.
<i>1.2</i>	Font name. If OPTS<1> = "" this defaults to "helv"
<i>2</i>	Set to 1 to map characters according to the emulation

## Example

The program example shown here is from the WIN.DBDEMO demonstration program.

```
CALL WIN.DBNEW(DBX, DLG.NAME, "Demonstration of Dialog
boxes", 10, 10, 200, 160,
'', '')
*
CALL WIN.DBLABEL(DBX, "Label 1", 4, 4, '', '', '')
CALL WIN.DBEDIT(DBX, "E1", 40, 4, 40, '', '')
CALL WIN.DBTEXT(DBX, "T1", 4, 24, 80, '', 'C')
CALL WIN.DBCHECK(DBX, "C1", "Check one", 4, 36, 60, '')
.
.
.
CALL WIN.DBLOAD(DBX, ERR)
IF ERR THEN
  ERR.MSG = 'Loading with WIN.DBLOAD'
  * Just delete dialog and try again if we get already exist error
  IF ERR = 3 THEN
    CALL WIN.DBDEL(DLG.NAME)
    CALL WIN.DBLOAD(DBX, ERR)
  END
END
END
```

```
*
IF ERR = 0 THEN
    CALL WIN.DBSHOW(DLG.NAME, 1, ERR)
    IF ERR THEN
        CALL WIN.DBDEL(DLG.NAME)
        ERR.MSG = 'Displaying with WIN.DBSHOW'
    END
END
END
```

---

### A simple test to show a dialog box in the taskbar

---

```
AM=CHAR(254)
DBX=''
CALL WIN.DBNEW(DBX, "APPWIND", "App Window", 10,10, 200,100,
"":AM:"WS_EX_APPWINDOW", "")
CALL WIN.DBLABEL(DBX, "This dialog box shows in the task bar",
10,10,120,14,"")
CALL WIN.DBLOAD(DBX, ERR)
IF ERR = 0 THEN
    CALL WIN.DBSHOW("APPWIND",1,ERR)
END
END
```

---

## Related Subroutines

[WIN.DBEDIT](#), [WIN.DBTEXT](#), [WIN.DBLABEL](#), [WIN.DBLOAD](#), [WIN.DBSHOW](#),  
[WIN.DBEVENT](#), [WIN.DBEND](#), [WIN.DBDEL](#)

## Related Script Commands

DialogBox Create, DialogBox Caption, DialogBox Style, DialogBox Font,  
DialogBox ExStyle

## Version

5.1.3 Support for Extended styles

# WIN.DBOPTION

## Syntax

WIN.DBOPTION ( *DBX*, *OPTS* )

## Description

This subroutine adds the options for a dialog to a dialog box. See the Parameters table for available options.

The "V" options lock the host program until WIN.DBINPOK with the "True" option is called. When using V1 and V2 only the following event is saved until the WIN.DBINPOK. subroutine is called. WIN.DBINPOK also enables the dialog box if option V2 disabled it.

## Parameters

The following table describes the parameters of the WIN.DBOPTION command:

Parameter	Description
<i>DBX</i>	This is the wIntegrate "handle" for a dialog box.
<i>OPTS</i>	See the options table below

## Values for OPTS

Combine the following in a single string

Value	Description
<i>V0</i>	Runs the Validate script with no more processing.
<i>V1</i>	Runs the Validate script, no message processing
<i>V2</i>	As V1 but disable further input.
<i>L</i>	No effect. Retained for backwards compatability

Value	Description
<i>B</i>	Make font bold and resize dialog to match a dialog create with the 16 bit version

## Example

This example is a part of the WIN.PRODMD2 example program.

---

```
* Set disable dialog while waiting for host option  
CALL WIN.DBOPTION(DBX, "V2")
```

---

## Related Subroutines

[WIN.DBINPOK](#), [WIN.DBNEW](#)

## Related Script Commands

DialogBox Option

# WIN.DBPANEL

## Syntax

WIN.DBPANEL ( *DBX*, *PANEL*, *POSITION* )

## Description

This subroutine adds a divider that splits the dialog box into different areas that are used to automatically position controls when the dialog box is resized.

When a dialog box is resized the panels determine the position and size of the controls as follows:

Controls to the left of the left panel are not moved

Controls above the top panel are not moved

Controls to the right of the right panel are moved so they remain the same distance from the right hand edge of the dialog box.

Controls below the bottom panel are moved so they remain the same distance from the bottom edge of the dialog box

Controls in the area between all the panels are resized so that their right edge remains the same distance from the right hand side of the dialog box and their bottom edge remains the same distance from the bottom of the dialog box

When any panels are created the minimum size of the dialog box is restricted to so that the controls will not be moved over each other and that a resized control will not go smaller than 4x4 pixels. To override this use the WIN.DBMNSIZE subroutine, but note that panels calculate what controls to move by the current position of the panel so if the minimum size is made too small it is possible to shrink the dialog box to include a control in a panel that wasn't there before.

The unit of measure for the position parameter is in dialog box units. This is calculated as 1/4 of the average width and 1/8 of the average depth of the dialog box font.

## Parameters

The following table describes the parameters of the WIN.DBPANEL command:

Parameter	Description
<i>DBX</i>	This is the "handle" for a dialog box as it is constructed
<i>PANEL</i>	The panel to add. One of "LEFT", "RIGHT", "TOP" or "BOTTOM".
<i>POSITION</i>	The position of the panel in dialog units

## Related Subroutines

[WIN.DBMNSIZE](#)

## Related Script Commands

DialogBox Panel

## Version

4.2.1 Original

# WIN.DBPOST

## Syntax

**WIN.DBPOST** ( *DLG*, *CTRLS*, *ETYPE* )

## Description

This command is used in conjunction with WIN.DBFETCH. WIN.DBPOST is used to put an event in the queue again after an event has been checked by WIN.DBFETCH.

## Parameters

The following table describes the parameters of the WIN.DBPOST command:

Parameter	Description
<i>DLG</i>	The dialog box name
<i>CTRLS</i>	The control name.
<i>ETYPE</i>	The event type. See WIN.DBEVENT for details.

## Example

This example is taken from the WIN.CUSTMD demonstration program.

```
This example is taken from the WIN.CUSTMD demonstration program.
* Validation events
* These events are often immediately followed by a default for the
* Next field or a button click. So we must use WIN.DBFETCH to check
* the next event and WIN.DBPOST to re-queue it (if required).
*
300 CALL WIN.DBFETCH(NEXT.DLG, NEXT.CTRL, NEXT.ETYPE)
*
* If next event is click on cancel don't bother with validation
IF NEXT.ETYPE = "C" AND NEXT.CTRL = "Cancel" THEN
PRINT "Ignoring last event as it is followed by:-"
PRINT "Event ":NEXT.ETYPE:" from ":NEXT.CTRL
FLD = NEXT.CTRL
GOSUB 340
END ELSE
*
* Field Validations
BEGIN CASE
```

```
CASE FLD = "CustRef"; GOSUB 1050; * Validate customer number
CASE FLD = "Name"; GOSUB 1150; * Validate Name
CASE FLD = "Street"; CALL WIN.DBGET(DBX.NAME, FLD, CUST.STREET)
CASE FLD = "City" ; CALL WIN.DBGET(DBX.NAME, FLD, CUST.CITY)
CASE FLD = "State"; CALL WIN.DBGET(DBX.NAME, FLD, CUST.STATE)
CASE FLD = "Zip"; CALL WIN.DBGET(DBX.NAME, FLD, CUST.ZIP)
CASE FLD = "Tel";
CALL WIN.DBGET(DBX.NAME, FLD, CUST.TEL)
CASE FLD = "Fax"; CALL WIN.DBGET(DBX.NAME, FLD, CUST.FAX)
CASE FLD = "Salesman"; CALL WIN.DBGET(DBX.NAME, FLD, CUST.SALESMAN)
CASE FLD = "Comment"; CALL WIN.DBGET(DBX.NAME, FLD, CUST.COMMENT)
CASE FLD = "Discount"; GOSUB 1250; * Validate discount
END CASE
IF NEXT.ETYPE # "" THEN CALL WIN.DBPOST(NEXT.DLG, NEXT.CTRL,
NEXT.ETYPE)
END
```

---

## Related Subroutines

[WIN.DBFETCH](#), [WIN.DBEVENT](#)



# WIN.DBPRGRES

## Syntax

**WIN.DBPRGRES** ( *DBX*, *NAME*, *X*, *Y*, *W*, *D*, *STYLE* )

## Description

This subroutine adds a progress control to a dialog box. The progress control displays a thick bar that is increased in length to show the progress of some action.

Use WIN.DBEVENT to set up the events to be returned by the control. The initial properties of a control can be set with WIN.DBINIPRP. After the dialog box containing the control has been shown properties can be set and retrieved using WIN.DBSETPRP and WIN.DBGETPRP. The methods of the control can be run at this time using WIN.DBMETHOD.

For a full list of properties, methods and events see the Client scripting reference.

## Parameters

The following table describes the parameters of the WIN.DBPRGRES command:

Parameter	Description
<i>DBX</i>	This is the wIntegrate "handle" for a dialog box.
<i>NAME</i>	Controls name
<i>X</i>	Column position
<i>Y</i>	Row position
<i>W</i>	Width
<i>D</i>	Depth
<i>STYLE</i>	See the Client Scripting Addendum

## Related Subroutines

[WIN.DBINIPRP](#), [WIN.DBEVENTS](#), [WIN.DBSETPRP](#), [WIN.DBGETPRP](#),  
[WIN.DBMETHOD](#)

## **Related Script Commands**

DialogBox Progress

## **Version**

4.0 Original

# WIN.DB RADIO

## Syntax

**WIN.DB RADIO** ( *DBX*, *NAME*, *TEXT*, *X*, *Y*, *W*, *D*, *GROUP* )

## Description

This subroutine adds a radio button to a dialog box. A radio button is an inset circle that can be clicked on or off. When a radio button is on, it displays a black circle in the center.

Use a group box (WIN.DB GROUP) to group several radio buttons to choose one of several options.

Note Check boxes are similar to radio buttons, but cannot be grouped. Use check boxes if you want the user to choose one option.

## Parameters

The following table describes the parameters of the WIN.DB RADIO command:

Parameter	Description
<i>DBX</i>	This is the wIntegrate "handle" for a dialog box.
<i>NAME</i>	The name you give the radio button.
<i>TEXT</i>	The text to display for the button.
<i>X</i>	Specifies the radio button column position.
<i>Y</i>	Specifies the radio button row position.
<i>W</i>	Defines the radio button width.
<i>D</i>	Defines the radio button depth
<i>GROUP</i>	Designates which button is first radio button in a group. Set the first radio button in a group to 1, and set all others in the group to 0. If one button in a group is selected, wIntegrate automatically turns off other radio buttons in a group.

## Example

This example is part of the WIN.DBDEMO demonstration program. This first line of the example shows the creation of a group box, and the next three lines place radio buttons within the group box. Note that the radio button “R1” has a 1 as the GROUP parameter to designate it as the first in the group of three buttons.

---

```
CALL WIN.DBGROUP(DBX,"G1","Group box",4,50,60,52)
CALL WIN.DBRADIO(DBX,"R1","Radio One",6,58,50,'',1)
CALL WIN.DBRADIO(DBX,"R2","Radio Two",6,72,50,'',0)
CALL WIN.DBRADIO(DBX,"R3","Radio Three",6,88,50,'',0)
```

---

## Related Subroutines

[WIN.DBNEW](#), [WIN.DBLOAD](#), [WIN.DBCAPT](#), [WIN.DBEVENT](#),  
[WIN.DBGROUP](#)

## Related Script Commands

DialogBox RadioButton, DialogBox ControlCommand

# WIN.DBRECT

## Syntax

**WIN.DBRECT** ( *DLGNAME*, *NAME*, *OPT*, *LEFT*, *TOP*, *RIGHT*, *BOTTOM* )

## Description

This subroutine returns the current position of a displayed dialog box or control.

## Parameters

The following table describes the parameters of the WIN.DBRECT command:

Parameter	Description
<i>DLGNAME</i>	The name of the dialog box
<i>NAME</i>	Controls name. Set to "" to return details on the dialog box
<i>OPT</i>	The information to return: See table below
<i>LEFT</i>	Returned left position of the dialog box or control
<i>TOP</i>	Returned top position of the dialog box or control
<i>RIGHT</i>	Returned right position of the dialog box or control
<i>BOTTOM</i>	Returned bottom position of the dialog box or control

## Values for OPT

Information to return

Value	Description
1	Client rectangle. This is the position of the internal dimensions of the dialog or control.
2	Absolute position of dialog box or control. This is the position on the desktop of the control

## **Version**

4.1 Original

# WIN.DBSCROLL

## Syntax

**WIN.DBSCROLL** ( *DBX*, *NAME*, *X*, *Y*, *W*, *D*, *STYLE*, *MIN*, *MAX* )

## Description

This subroutine adds a scroll bar to a dialog box.

## Parameters

The following table describes the parameters of the WIN.DBSCROLL command:

Parameter	Description
<i>DBX</i>	This is the wIntegrate "handle" for a dialog box.
<i>NAME</i>	The name you give to the scroll bar.
<i>X</i>	Specifies the scroll bar column position.
<i>Y</i>	Specifies the scroll bar row position.
<i>W</i>	Defines the scroll bar width. The default value is 12.
<i>D</i>	Defines the scroll bar depth. The default value is 12.
<i>STYLE</i>	Scroll bar style. See the scroll bar control in the Script Reference
<i>MIN</i>	The minimum value to scroll. The default value is 0.
<i>MAX</i>	The maximum value for the scroll bar. the default is 100

## Example

The following example is part of the WIN.DBDEMO demonstration program.

---

The following example is part of the WIN.DBDEMO demonstration program.  
`CALL WIN.DBSCROLL(DBX,"S1", 4,116,60,'','WS_TABSTOP','','')`

---

## Related Subroutines

[WIN.DBNEW](#), [WIN.DBLOAD](#), [WIN.DBCAPT](#), [WIN.DBEVENT](#)

## Related Script Commands

DialogBox ScrollBar



# WIN.DBSELECT

## Syntax

**WIN.DBSELECT** ( *DLGNAME*, *NAME*, *TEXT* )

## Description

This subroutine selects one of several options in a list or combo box dialogbox field.

## Parameters

The following table describes the parameters of the WIN.DBSELECT command:

Parameter	Description
<i>DLGNAME</i>	The name of the dialog box shown.
<i>NAME</i>	The name of the control.
<i>TEXT</i>	The text of the line to select.

## Example

This example is a part of the WIN.DBDEMO demonstration program.

---

This example is a part of the WIN.DBDEMO demonstration program.

```
IF ERR = 0 THEN
CALL WIN.DBSET(DLG.NAME,"T1","Centered Text")
CALL WIN.DBSET(DLG.NAME,"R1",1)
R.LIST = "Option 1"
R.LIST<-1>="Option 2"
R.LIST<-1>="Option 3"
CALL WIN.DBSET(DLG.NAME,"L1", R.LIST)
CALL WIN.DBSELECT(DLG.NAME,"L1", "Option 2")
CALL WIN.DBSET(DLG.NAME,"CB1",R.LIST)
CALL WIN.DBSELECT(DLG.NAME,"CB1", "Option 3")
END
```

Here WIN.DBSELECT selects Option 2 in the list box and Option 3 in the combo box

---

## Related Subroutines

[WIN.DBSET](#), [WIN.DBLIST](#), [WIN.DBCOMBO](#), [WIN.DBGET](#), [WIN.DBEDGET](#),  
[WIN.DBEDSET](#)

## Related Script Commands

DialogBox Select

# WIN.DBSET

## Syntax

**WIN.DBSET** ( *DLGNAME*, *NAME*, *VALUE* )

## Description

This subroutine sets dialog box controls that were previously created. You need this subroutine in conjunction with many of the dialog box routines like WIN.DBEDIT, WIN.DBTEXT, WIN.DBCOMBO, etc.

This sets the default property for the control. See the Script Reference for details.

For multi-line edit controls, use the WIN.DBEDSET routine.

## Parameters

The following table describes the parameters of the WIN.DBSET command:

Parameter	Description
<i>DLGNAME</i>	The name of the active dialog box.
<i>NAME</i>	The name of the control to set.
<i>VALUE</i>	The value depends on what type of control .

## Example

This example is part of the WIN.DBDEMO demonstration program.

---

```
* The name and position of the text control are created with
WIN.DBTEXT,
* then the text is set using WIN.DBSET.
CALL WIN.DBNEW(DBX, DLG.NAME, "Demonstration of Dialog boxes",
10,10,200,160, ' ', '')
*
CALL WIN.DBLABEL(DBX, "Label 1", 4, 4, ' ', ' ', ' ')
CALL WIN.DBEDIT(DBX, "E1", 40, 4, 40, ' ', ' ')
CALL WIN.DBTEXT(DBX, "T1", 4, 24, 80, ' ', 'C')
CALL WIN.DBCHECK(DBX, "C1", "Check one", 4, 36, 60, ' ')
.
.
.
```

```
IF ERR = 0 THEN
  CALL WIN.DBSET(DLG.NAME, "T1", "Centered Text")
  CALL WIN.DBSET(DLG.NAME, "R1", 1)
  R.LIST = "Option 1"
  R.LIST<-1>="Option 2"
  R.LIST<-1>="Option 3"
  CALL WIN.DBSET(DLG.NAME, "L1", R.LIST)
  CALL WIN.DBSELECT(DLG.NAME, "L1", "Option 2")
  CALL WIN.DBSET(DLG.NAME, "CB1", R.LIST)
  CALL WIN.DBSELECT(DLG.NAME, "CB1", "Option 3")
END
```

---

## Related Subroutines

[WIN.DBGET](#), [WIN.DBEDSET](#), [WIN.DBSETPRP](#), [WIN.DBINIPRP](#)

# WIN.DBSETCOL

## Syntax

**WIN.DBSETCOL** ( *DBX*, *CTRLNAME*, *FORECOL*, *BACKCOL* )

## Description

This subroutine sets the foreground and the background color of a specific control within a dialog box. If the parameter *CTRLNAME* is None, it affects all controls named None.

## Parameters

The following table describes the parameters of the WIN.DBSETCOL command:

Parameter	Description
<i>DBX</i>	Handle for current dialog box.
<i>CTRLNAME</i>	Name of the control to change.
<i>FORECOL</i>	Foreground color.
<i>BACKCOL</i>	Background color.

## Example

The following example sets the text to blue in a control called Text1.

---

```
The second line sets the text to white and the background color to  
blue in a control called Event. See the demonstration program  
WIN.DBDEMO2.  
CALL WIN.DBSETCOL(DBX, "Text1", "Blue", "")  
CALL WIN.DBSETCOL(DBX, "Event", "White", "Blue")
```

---

# WIN.DBSETFNT

## Syntax

**WIN.DBSETFNT** ( *DBX*, *CTRLNAME*, *FONTNAME*, *PTSIZE*, *STYLE* )

## Description

This subroutine sets the font for a specific control within a dialogbox. See the demonstration program WIN.DBDEMO2.

## Parameters

The following table describes the parameters of the WIN.DBSETFNT command:

Parameter	Description
<i>DBX</i>	Handle for creating the dialog box
<i>CTRLNAME</i>	Name of the control to set the font for
<i>FONTNAME</i>	The name of the font to use
<i>PTSIZE</i>	Size of the font in points
<i>STYLE</i>	Styles separated by " "

## Example

Set the fonts in a dialog

---

```
CALL WIN.DBSETFNT(DBX, "Text1", "Times New Roman", 18,  
"FONT_Bold|FONT_Italic")  
CALL WIN.DBSETFNT(DBX, "Event", "Courier New", 8, "")
```

---

# WIN.DBSETMNU

## Syntax

**WIN.DBSETMNU** ( *DBX*, *MENUNAME* )

## Description

This subroutine sets a previously created menu for use in a dialog box.

Note A dialog box which has a menu cannot have the DS\_MODALFRAME style.  
See WIN.DBNEW.

## Parameters

The following table describes the parameters of the WIN.DBSETMNU command:

Parameter	Description
<i>DBX</i>	This is the wIntegrate "handle" for a dialog box.
<i>MENUNAME</i>	The name of menu.

## Example

Add a menu to a dialog box

---

```
CALL WIN.MENULOAD("DDMenuBar", MENUBAR)
CALL WIN.DBNEW(DBX, DLG.NAME, "Dialog box menus, fonts and colors",
10,10,200,160, " ",OPTS)
CALL WIN.DBSETMNU(DBX, "DDMenuBar")
```

---

# WIN.DBSETPRP

## Syntax

**WIN.DBSETPRP** ( *DLG.NAME*, *NAME*, *PROPERTY*, *VALUE* )

## Description

This subroutine sets the value for a controls property after it has been loaded to thePC with WIN.DBLOAD.

For a full list of properties for each control see the Client scripting reference.

## Parameters

The following table describes the parameters of the WIN.DBSETPRP command:

Parameter	Description
<i>DLG.NAME</i>	The name of the loaded dialog box
<i>NAME</i>	Controls name
<i>PROPERTY</i>	Name of the property to set
<i>VALUE</i>	New value of the property

## Related Subroutines

[WIN.DBGETPRP](#), [WIN.DBINIPRP](#), [WIN.DBMETHOD](#)

## Version

4.0.1 Original



# WIN.DBSHOW

## Syntax

WIN.DBSHOW ( *DLGNAME*, *MODAL*, *ERR* )

## Description

This subroutine displays the dialog box on the PC screen. The dialog must first be created with WIN.DBNEW, and loaded with WIN.DBLOAD.

## Parameters

The following table describes the parameters of the WIN.DBSHOW command:

Parameter	Description
<i>DLGNAME</i>	The name of the dialog box to display.
<i>MODAL</i>	Specifies the dialog box as modal or modeless. 1 = modal 0 = modeless. If the dialog is modal the session window is disabled until the user closes the dialog box.
<i>ERR</i>	This parameter returns error codes. See below for details.

## Values for ERR

The following error codes are returned.

Value	Description
1	The call is missing a dialog box name e.g. DLGNAME="".
2	The dialog box is already shown. This means that it has been displayed and may not have a WIN.DBEND call to close it.
3	The dialog box is not loaded. This means that the dialog may be missing a WIN.DBLOAD call.

Value	Description
4	Unable to create the dialog box window.

## Example

This example is part of the WIN.DBDEMO demonstration program. Remember that you must use

---

```
Remember that you must use WIN.DBSHOW to display a dialog box.  
CALL WIN.DBSHOW(DLG.NAME, 1, ERR)  
IF ERR THEN  
CALL WIN.DBDEL(DLG.NAME)  
ERR.MSG = 'Displaying with WIN.DBSHOW'
```

---

## Related Subroutines

[WIN.DBLOAD](#), [WIN.DBEND](#), [WIN.DBNEW](#)

## Related Script Commands

DialogBox Show, DialogBox Window

# WIN.DBSHOWPU

## Syntax

**WIN.DBSHOWPU** *DLGNAME*, *PARENT*, *ALIGNCTRL*, *ALIGNFLAG*

## Description

This command shows a dialog box as a window and automatically aligns it to the specified control.

If the window can not be seen fully as the control is too near to an edge of the screen it is repositioned so both it and the original control can be seen.

## Parameters

The following table describes the parameters of the WIN.DBSHOWPU command:

Parameter	Description
<i>DLGNAME</i>	Name of the dialog box to show
<i>PARENT</i>	The name of the dialog box this window is being popped up from
<i>ALIGNCTRL</i>	The name of the control with which to align the window
<i>ALIGNFLAG</i>	The alignment option. 0 - left align, 1 - right align

## Example

Popup the previously defined calculator dialog

---

```
* Right aligning the dialog to the second icon in AlansUtils toolbar  
CALL WIN.DBSHOWPU("Calculator","AlansUtils","icon2",1,ERR)
```

---

## Related Subroutines

[WIN.DBSHOW](#), [WIN.DBATTACH](#)

## **Related Script Commands**

DialogBox PopupWindow

## **Version**

4.2.1 Original

# WIN.DBSTATUS

## Syntax

**WIN.DBSTATUS** ( *NAME*, *LOADED*, *SHOWN* )

## Description

This subroutine verifies whether a dialog box is loaded to the PC and if it is currently shown.

## Parameters

The following table describes the parameters of the WIN.DBSTATUS command:

Parameter	Description
<i>NAME</i>	The name of the dialog box.
<i>LOADED</i>	Returns the dialog load status. 0 = dialog is not loaded 1 = dialog is loaded
<i>SHOWN</i>	Returns the dialog shown status. 0 = dialog is not shown 1 = dialog is shown

## Example

This example checks to see if the dialog box needs to be loaded and shows the dialog box if it is not already displayed.

---

```
* DemoDlg definition is in DBX
CALL WIN.DBSTATUS ("DemoDlg", LOADED, SHOWN)
IF NOT (LOADED) THEN CALL WIN.DBLOAD (DBX, ERR)
IF ERR = 0 AND NOT (SHOWN) THEN
    CALL WIN.DBSHOW("MemoDlg", 1, ERR)
END
```

---

## Related Subroutines

[WIN.DBLOAD](#), [WIN.DBSHOW](#)

## **Related Script Commands**

IsShown, IsDialog

# WIN.DBTAB

## Syntax

**WIN.DBTAB** ( *DBX*, *NAME*, *X*, *Y*, *W*, *D*, *STYLE* )

## Description

This subroutine adds a tab control to a dialog box. The tab control show tabs across the dialog box.

Use WIN.DBEVENT to set up the events to be returned by the control. The initial properties of a control can be set with WIN.DBINIPRP. After the dialog box containing the control has been shown properties can be set and retrieved using WIN.DBSETPRP and WIN.DBGETPRP. The methods of the control can be run at this time using WIN.DBMETHOD.

For a full list of properties, methods and events see the Client scripting reference.

## Parameters

The following table describes the parameters of the WIN.DBTAB command:

Parameter	Description
<i>DBX</i>	This is the wIntegrate "handle" for a dialog box.
<i>NAME</i>	Controls name
<i>X</i>	Column position
<i>Y</i>	Row position
<i>W</i>	Width
<i>D</i>	Depth
<i>STYLE</i>	See the Client Scripting Reference

## Related Subroutines

[WIN.DBINIPRP](#), [WIN.DBEVENTS](#), [WIN.DBSETPRP](#), [WIN.DBGETPRP](#),  
[WIN.DBMETHOD](#)

## Related Script Commands

DialogBox TabControl

## Version

4.0 Original



# WIN.DBTABS

## Syntax

**WIN.DBTABS** ( *DLGNAME*, *NAME*, *TABS* )

## Description

This subroutine sets the location of the tab stops in a list box control.

## Parameters

The following table describes the parameters of the WIN.DBTABS command:

Parameter	Description
<i>DLGNAME</i>	The name of the dialog box.
<i>NAME</i>	The name of the list box where tabs will be set.
<i>TABS</i>	A string containing comma-separated tab positions. The tab positions are measured in dialog units.

## Example

Set up tabs at 20 characters and 60 characters

---

```
CALL WIN.DBTABS(DLGNAME, "CustList", "20,60")
REC<1,1> = "ID1"
REC<1,2> = "Company1"
REC<1,3> = "Phone rate"
.
.
.
CALL WIN.DBSET(DLGNAME, "CustList", REC)
```

---

## Related Subroutines

[WIN.DBLIST](#), [WIN.DBSHOW](#)

## Related Script Commands

DialogBox SetTabs

# WIN.DBTEXT

## Syntax

**WIN.DBTEXT** ( *DBX*, *NAME*, *X*, *Y*, *W*, *D*, *OPTIONS* )

## Description

This subroutine creates a text control in a dialog box. You can set and change the text of this control with the WIN.DBSET subroutine.

When you need a label that does not change, for instance to display text as a label for a group, edit, or list box, use WIN.DBLABEL.

## Parameters

The following table describes the parameters of the WIN.DBTEXT command:

Parameter	Description
<i>DBX</i>	This is the wIntegrate "handle" for a dialog box.
<i>NAME</i>	The name of the text control
<i>X</i>	Column position
<i>Y</i>	Row position
<i>W</i>	Width. The default is the length of the name multiplied by 4
<i>D</i>	Depth. The default depth is 12
<i>OPTIONS</i>	See the following options table

## Values for DBX

Value	Description
<i>1</i>	Text justification. This can be one of the following values:
<i>2</i>	L = left
<i>3</i>	R = right

## Fields for OPTIONS

The options are specified in a dynamic array of the following fields:

Field	Description
1	Text justification. This can be one of the following values: L = Left, R = Right, C = Center
2	Fixed number of dialog units to add to the width.
3	Put the label in a sunken rectangle if set to 1

## Example

Set up a normal and sunken text control

```
SINK.TEXT = ""
SINK.TEXT<3> = 1; * Sunken text option for text and labels
CALL WIN.DBTEXT(DBX, "Text1", 5,22,78,12,"")
CALL WIN.DBINIPRP(DBX,"Text1","", "This is normal text")
CALL WIN.DBTEXT(DBX, "Text2", 141,22,78,12, SINK.TEXT)
CALL WIN.DBINIPRP(DBX,"Text2","", "This is sunken text")
```

## Related Subroutines

[WIN.DBLABEL](#)

## Related Script Commands

DialogBox LText, DialogBox CText, DialogBox RText

## Version

4.1.1 Added Sunken and extra width option

# WIN.DBTRACK

## Syntax

**WIN.DBTRACK** ( *DBX*, *NAME*, *X*, *Y*, *W*, *D*, *STYLE* )

## Description

This subroutine adds a trackbar control to a dialog box.

Use WIN.DBEVENT to set up the events to be returned by the control. The initial properties of a control can be set with WIN.DBINIPRP. After the dialog box containing the control has been shown properties can be set and retrieved using WIN.DBSETPRP and WIN.DBGETPRP. The methods of the control can be run at this time using WIN.DBMETHOD.

For a full list of properties, methods and events see the Client scripting reference.

## Parameters

The following table describes the parameters of the WIN.DBTRACK command:

Parameter	Description
<i>DBX</i>	This is the wIntegrate "handle" for a dialog box.
<i>NAME</i>	Controls name
<i>X</i>	Column position
<i>Y</i>	Row position
<i>W</i>	Width
<i>D</i>	Depth
<i>STYLE</i>	See the Client Scripting Reference

## Related Subroutines

[WIN.DBINIPRP](#), [WIN.DBEVENTS](#), [WIN.DBSETPRP](#), [WIN.DBGETPRP](#),  
[WIN.DBMETHOD](#)

## **Related Script Commands**

DialogBox Trackbar

## **Version**

4.0.1 Original

# WIN.DBTREEVW

## Syntax

**WIN.DBTREEVW** ( *DBX*, *NAME*, *X*, *Y*, *W*, *D*, *STYLE* )

## Description

This subroutine creates a tree structure control for displaying items. A tree view control is used to display hierarchical data. Each line of the control can be expanded to show further lines.

Images can be assigned to each line, its state can be changed, and a number can be associated with it.

Each line of the control has its own unique item ID. Using this item ID it is possible to modify or retrieve the current text, images, state, and data for the line.

Use WIN.DBEVENT to set up the events to be returned by the control. The initial properties of a control can be set with WIN.DBINIPRP. After the dialog box containing the control has been shown properties can be set and retrieved using WIN.DBSETPRP and WIN.DBGETPRP. The methods of the control can be run at this time using WIN.DBMETHOD.

For a full list of properties, methods and events see the Client scripting reference.

## Parameters

The following table describes the parameters of the WIN.DBTREEVW command:

Parameter	Description
<i>DBX</i>	This is the wIntegrate "handle" for a dialog box.
<i>NAME</i>	Controls name
<i>X</i>	Column position
<i>Y</i>	Row position
<i>W</i>	Width
<i>D</i>	Depth

Parameter	Description
<i>STYLE</i>	See the Client Scripting Reference

## Related Subroutines

[WIN.DBINIPRP](#), [WIN.DBEVENTS](#), [WIN.DBSETPRP](#), [WIN.DBGETPRP](#),  
[WIN.DBMETHOD](#)

## Related Script Commands

DialogBox TreeView

## Version

4.0 Original

# WIN.DBUNIT

## Syntax

WIN.DBUNIT ( *DLGNAME*, *CNV*, *VALUE* )

## Description

This subroutine is used to convert from dialog units to pixels.

## Parameters

The following table describes the parameters of the WIN.DBUNIT command:

Parameter	Description
<i>DLGNAME</i>	The name of the dialog box
<i>CNV</i>	The conversion to apply. See below
<i>VALUE</i>	Values to be converted.

## Values for CNV

Value	Description
<i>0</i>	Convert column pixels to column dialog units.
<i>1</i>	Convert row pixels to row dialog units.
<i>2</i>	Convert multiple column pixels to column dialog units.
<i>3</i>	Convert multiple row pixels to row dialog units.
<i>4</i>	Convert column dialog units to column pixels.
<i>5</i>	Convert row dialog units to row pixels.
<i>6</i>	Convert multiple column dialog units to column pixels



Value	Description
7	Convert multiple row dialog units to row pixels.

## Related Script Commands

DialogUnit

## Version

4.0.2 Original

# WIN.DBUPDOWN

## Syntax

**WIN.DBUPDOWN** ( *DBX*, *NAME*, *X*, *Y*, *W*, *D*, *STYLE* )

## Description

This subroutine adds a spin box (an edit box with up and down arrows) to a dialog box.

Use WIN.DBEVENT to set up the events to be returned by the control. The initial properties of a control can be set with WIN.DBINIPRP. After the dialog box containing the control has been shown properties can be set and retrieved using WIN.DBSETPRP and WIN.DBGETPRP. The methods of the control can be run at this time using WIN.DBMETHOD.

For a full list of properties, methods and events see the Client scripting reference.

## Parameters

The following table describes the parameters of the WIN.DBUPDOWN command:

Parameter	Description
<i>DBX</i>	This is the wIntegrate "handle" for a dialog box.
<i>NAME</i>	Controls name
<i>X</i>	Column position
<i>Y</i>	Row position
<i>W</i>	Width
<i>D</i>	Depth
<i>STYLE</i>	See the Client Scripting Reference.

## Related Subroutines

[WIN.DBINIPRP](#), [WIN.DBEVENTS](#), [WIN.DBSETPRP](#), [WIN.DBGETPRP](#),  
[WIN.DBMETHOD](#)

## **Related Script Commands**

DialogBox UpDown

## **Version**

4.1 Original

# WIN.DDECLOSE

## Syntax

WIN.DDECLOSE ( *NAME* )

## Description

This subroutine stops a DDE conversation with another Windows application. Start a DDE conversation with WIN.DDEOPEN.

## Parameters

The following table describes the parameters of the WIN.DDECLOSE command:

Parameter	Description
<i>NAME</i>	The name of the DDE conversation that you specified in WIN.DDEOPEN.

## Example

This example is a part of the WIN.DDEDEMO demonstration program.

```
* Excel DDE demonstration
200 CALL WIN.COLOR("Yellow","Blue")
CALL WIN.TWOPEN("ddedemo", "Excel Dynamic Data Exchange demo",10, 5,
70, 16,1)
PRINT "To find out the topics Excel supports:-"
PRINT "First open a link to the system topic as follows:-"
PRINT ' CALL WIN.DDEOPEN("EXCEL_LINK","excel","System", OK)'
CALL WIN.DDEOPEN("EXCEL_LINK","excel","System", OK)
.
.
.
CALL WIN.DDEREQ("EXCEL_LINK", "Topics", TOPICS)
PRINT
VALUE = TOPICS; S = CHAR(9); R = ","; GOSUB 500; * Replace tabs
PRINT "The topics known by Excel are"
PRINT VALUE
PRINT
PRINT 'Finally we close the link with'
PRINT ' CALL WIN.DDECLOSE("EXCEL_LINK")'
CALL WIN.DDECLOSE("EXCEL_LINK")
```

## **Related Subroutines**

[WIN.DDECLOSE](#), [WIN.DDEREQ](#), [WIN.DDEEXEC](#), [WIN.DDEPOKE](#)

## **Related Script Commands**

DDE Terminate

# WIN.DDEEXEC

## Syntax

WIN.DDEEXEC ( *NAME*, *MACRO* )

## Description

This subroutine executes a macro or command in another application using DDE. The format of the *MACRO* parameter depends on the requirements of the other application.

## Parameters

The following table describes the parameters of the WIN.DDEEXEC command:

Parameter	Description
<i>NAME</i>	The name of the current DDE conversation you specified in WIN.DDEOPEN.
<i>MACRO</i>	The command to run in the other Windows application.

## Example

This example is a part of the WIN.DDEDEMO demonstration program.

```
This example is a part of the WIN.DDEDEMO demonstration program.
*
PRINT @(-1):"Now to run a few Excel macros to tidy the data up"
PRINT
PRINT ' CALL WIN.DDEEXEC("EXCEL_SHEET",' : '' : '
[SELECT("R2C2:R8C3","R2C2")]': '')"
PRINT ' CALL WIN.DDEEXEC("EXCEL_SHEET", "[COLUMN.WIDTH(1,,3)]")'
CALL WIN.DDEEXEC("EXCEL_SHEET", '[SELECT("R2C2:R8C3","R2C2")]')
CALL WIN.DDEEXEC("EXCEL_SHEET", "[COLUMN.WIDTH(1,,3)]")
*
PRINT "These two Excel commands select our data then size it to for
the best fit"
*
PRINT
PRINT 'Finally we close our link'
PRINT ' CALL WIN.DDECLOSE("EXCEL_SHEET")'
CALL WIN.DDECLOSE("EXCEL_SHEET")
```

END

END

---

## **Related Subroutines**

[WIN.DDEOPEN](#), [WIN.DDECLOSE](#), [WIN.DDEREQ](#), [WIN.DDEPOKE](#),  
[WIN.DDETIME](#)

## **Related Script Commands**

DDE Execute

# WIN.DDEEXEC2

## Syntax

**WIN.DDEEXEC2** ( *NAME*, *MACRO*, *RESP* )

## Description

This subroutine executes a macro or command in another application using DDE. The format of the *MACRO* parameter depends on the requirements of the other application.

## Parameters

The following table describes the parameters of the WIN.DDEEXEC2 command:

Parameter	Description
<i>NAME</i>	The name of the current DDE conversation that you specified in WIN.DDEOPEN.
<i>MACRO</i>	The command to run in the other Windows application.
<i>RESP</i>	Variable to receive the returned status of the command. 0 no error, otherwise see the description of DDE Execute in the client scripting reference.

## Example

Simple test of status returned by WIN.DDEEXEC2

---

```
*
XL = "XL"
CALL WIN.DDEOPEN(XL, "Excel", "Sheet1", OK)
IF NOT(OK) THEN PRINT "Excel sheet 1 not started"; STOP
*
* Try the beep command that should exist in Excel
CALL WIN.DDEEXEC2(XL, "[BEEP()]", RESP)
PRINT "BEEP() macro command ":
IF RESP = 0 THEN
PRINT "successful"
END ELSE
PRINT "Failed, error code ": RESP
END
```



```
*
* Try a made up command that should fail as it's not part
* of Excels macro language
CALL WIN.DDEEXEC2(XL,"[MADEUP()]",RESP)
PRINT "MADEUP() macro command ":
IF RESP = 0 THEN
PRINT "successful"
END ELSE
PRINT "Failed, error code ":RESP
END
```

---

## **Related Subroutines**

[WIN.DDEEXEC](#)

## **Related Script Commands**

DDE Execute

## **Version**

4.1.1 Original

# WIN.DDEOPEN

## Syntax

**WIN.DDEOPEN** ( *NAME*, *SERVER*, *TOPIC*, *OK* )

## Description

This subroutine starts a DDE conversation between wIntegrate and another Windows application that supports DDE. The parameters are defined by the other application.

## Parameters

The following table describes the parameters of the WIN.DDEOPEN command:

Parameter	Description
<i>NAME</i>	The identifier or name that you assign to this DDE conversation.
<i>SERVER</i>	The remote application's DDE name.
<i>TOPIC</i>	The remote application topic name. This is dependent the application.
<i>OK</i>	The number returned if the link is successful. 0 = the link is not started 1 = the link is started.

## Example

This example is a part of the WIN.DDEDEMO demonstration program.

```
* Excel DDE demonstration
200 CALL WIN.COLOR("Yellow","Blue")
CALL WIN.TWOPEN("ddedemo", "Excel Dynamic Data Exchange demo",10, 5,
70, 16,1)
PRINT "To find out the topics Excel supports:-"
PRINT "First open a link to the system topic as follows:-"
PRINT ' CALL WIN.DDEOPEN("EXCEL_LINK","excel","System", OK)'
CALL WIN.DDEOPEN("EXCEL_LINK","excel","System", OK)
.
.
.
CALL WIN.DDEREQ("EXCEL_LINK", "Topics", TOPICS)
PRINT
VALUE = TOPICS; S = CHAR(9); R = ","; GOSUB 500; * Replace tabs
```

```
PRINT "The topics known by Excel are"  
PRINT VALUE  
PRINT  
PRINT 'Finally we close the link with'  
PRINT ' CALL WIN.DDECLOSE("EXCEL_LINK")'  
CALL WIN.DDECLOSE("EXCEL_LINK")
```

---

## **Related Subroutines**

[WIN.DDECLOSE](#), [WIN.DDEEXEC](#), [WIN.DDEPOKE](#), [WIN.DDEREQ](#),  
[WIN.DDETIME](#), [WIN.DDEEXEC2](#)

## **Related Script Commands**

DDE Initiate

# WIN.DDEPOKE

## Syntax

**WIN.DDEPOKE** ( *NAME*, *ITEM*, *VALUE* )

## Description

This subroutine sends data from wIntegrate to another Windows application using Dynamic Data Exchange. The other application sets the requirements for the format of the *ITEM* and *VALUE* parameters.

## Parameters

The following table describes the parameters of the WIN.DDEPOKE command:

Parameter	Description
<i>NAME</i>	The name of the DDE conversation specified in WIN.DDEOPEN.
<i>ITEM</i>	The remote application item.
<i>VALUE</i>	The new value of the item.

## Example

This example is a part of the WIN.DDEDEMO demonstration program.

```
INPUT DUM:
*
PRINT @(-1):"First we open a link to the spread sheet 'Sheet1'"
PRINT ' CALL WIN.DDEOPEN("EXCEL_SHEET","EXCEL","SHEET1",OK)'
CALL WIN.DDEOPEN("EXCEL_SHEET","EXCEL","SHEET1",OK)
IF NOT(OK) THEN
CALL WIN.COLOR("LightRed","")
PRINT "We could not open the link so we will exit the demonstration."
END ELSE
PRINT "We then add the titles to the spreadsheet"
PRINT ' CALL WIN.DDEPOKE("EXCEL_SHEET", "R2C2", "Fruit")'
PRINT ' CALL WIN.DDEPOKE("EXCEL_SHEET", "R2C3", "Quantity")'
CALL WIN.DDEPOKE("EXCEL_SHEET", "R2C2", "Fruit")
CALL WIN.DDEPOKE("EXCEL_SHEET", "R2C3", "Quantity")
RQM
*
PRINT "and some data in exactly the same way"
```

```
R.DATA = "Apple"; R.DATA<1,2> = 20
R.DATA<2,1> = "Banana"; R.DATA<2,2> = 30
R.DATA<3,1> = "Pear"; R.DATA<3,2> = 10
FOR J = 1 TO 3
CALL WIN.DDEPOKE("EXCEL_SHEET", "R":(J+3):"C2",R.DATA<J,1>)
CALL WIN.DDEPOKE("EXCEL_SHEET", "R":(J+3):"C3", R.DATA<J,2>)
NEXT J
RQM
```

---

## **Related Subroutines**

[WIN.DDEOPEN](#), [WIN.DDECLOSE](#), [WIN.DDEEXEC](#), [WIN.DDEREQ](#),  
[WIN.DDETIME](#)

## **Related Script Commands**

DDE Poke

# WIN.DDEREQ

## Syntax

**WIN.DDEREQ** ( *NAME*, *ITEM*, *VALUE* )

## Description

This subroutine gets data from another Windows application using DDE.

## Parameters

The following table describes the parameters of the WIN.DDEREQ command:

Parameter	Description
<i>NAME</i>	The name of the DDE conversation specified WIN.DDEOPEN.
<i>ITEM</i>	The remote application item. This is dependent the remote application.
<i>VALUE</i>	The new value of the item. This is dependent the remote application.

## Example

This example is a part of the WIN.DDEDEMO demonstration program.

```
This example is a part of the WIN.DDEDEMO demonstration program.
IF OK THEN
PRINT
PRINT "Press <CR> to continue ":
INPUT DUM:
PRINT @(-1):"Link opened to Excel system topic"
PRINT
PRINT "Ask for the Topics items data"
PRINT ' CALL WIN.DDEREQ("EXCEL_LINK", "Topics", TOPICS)'
TOPICS = ''
CALL WIN.DDEREQ("EXCEL_LINK", "Topics", TOPICS)
PRINT
VALUE = TOPICS; S = CHAR(9); R = ","; GOSUB 500; * Replace tabs
PRINT "The topics known by Excel are"
PRINT VALUE
PRINT
PRINT 'Finally we close the link with'
```

```
PRINT ' CALL WIN.DDECLOSE ( "EXCEL_LINK" ) '  
CALL WIN.DDECLOSE ( "EXCEL_LINK" )
```

---

## **Related Subroutines**

[WIN.DDEOPEN](#), [WIN.DDECLOSE](#), [WIN.DDEEXEC](#), [WIN.DDEPOKE](#),  
[WIN.DDETIME](#)

## **Related Script Commands**

DDERequest

# WIN.DDETIME

## Syntax

**WIN.DDETIME** ( *NAME*, *TIMEOUT* )

## Description

This subroutine sets the timeout for the DDE link. A timeout is the length of time wIntegrate waits for a command to execute before generating an error. If WIN.DDETIME is not set, then 10 seconds is the default timeout.

## Parameters

The following table describes the parameters of the WIN.DDETIME command:

Parameter	Description
<i>NAME</i>	The name of the DDE conversation specified in WIN.DDEOPEN.
<i>TIMEOUT</i>	The number of seconds for the timeout.

## Example

This example line will increase the default timeout from 10 to 30 seconds.

---

This example line will increase the default timeout from 10 to 30 seconds.

```
CALL WIN.DDETIME( "EXCEL_LINK", 30 )
```

---

## Related Subroutines

[WIN.DDEOPEN](#), [WIN.DDECLOSE](#), [WIN.DDEEXEC](#), [WIN.DDEPOKE](#),  
[WIN.DDEREQ](#), [WIN.DDEEXEC2](#)

## Related Script Commands

DDE Timeout



# WIN.DISPLAY

## Syntax

**WIN.DISPLAY** ( *TURN.ON* )

## Description

This subroutine turns the screen display on or off. When the screen display is turned off nothing is displayed, but the screen is in memory so that when it is turned on the entire screen display reappears.

## Parameters

The following table describes the parameters of the WIN.DISPLAY command:

Parameter	Description
<i>TURN.ON</i>	1 "ON" Turns the screen display on.

## Example

Turn off the display while executing Report

---

```
CALL WIN.DISPLAY(0) ;* Turn screen display off
EXECUTE "LIST STUDENT LNAME BY LNAME"
CALL WIN.DISPLAY(1) ;* Turn screen display back on
```

---

## Related Subroutines

[WIN.SCREEN](#)

## Related Script Commands

Display DisplayOn

# WIN.DRARC

## Syntax

**WIN.DRARC** ( *X1, Y1, X2, Y2, SA, EA, WIN.DRARC* )

## Description

This subroutine draws the arc of an ellipse within a rectangle defined by the parameters X1,Y1 to X2,Y2. If X1 and Y1 are null (""), then the top left of the rectangle is the current graphic cursor position.

## Parameters

The following table describes the parameters of the WIN.DRARC command:

Parameter	Description
<i>X1</i>	The x position of the top left of the rectangle.
<i>Y1</i>	The y position of the top left of the rectangle.
<i>X2</i>	The x position of the bottom right of the rectangle.
<i>Y2</i>	The y position of the bottom right of the rectangle.
<i>SA</i>	The degree of the start angle.
<i>EA</i>	The degree of the end angle.
<i>WIN.DRARC</i>	Parameters

## Example

This example is part of the WIN.DRDEMO demonstration program.

```
* Arcs etc
1400 GOSUB 5100; * Erase graphic
X = XTL
Y = YTL + 2
*
PRINT @(X,Y):"Arcs":
CALL WIN.DRARC((X+12)*10,Y*10-15,(X+24)*10,Y*10+25,0,45)*
Y = Y + 4
PRINT @(X,Y):"Chords":
CALL WIN.DRCHORD((X+12)*10,Y*10-15,(X+24)*10,Y*10+25,10,115)
*
```

```
Y = Y + 4
PRINT @(X,Y):"Pie segments":
CALL WIN.DRPIE((X+12)*10,Y*10-15,(X+24)*10,Y*10+25,0,45)
*
MSG = "The arcs, chords and pie segments are drawn with:-"
MSG<-1> = " WIN.DRARC(X1,Y1,X2,Y2,SA,EA),
WIN.DRCHORD(X1,Y1,X2,Y2,SA,EA) "
MSG<-1> = "and WIN.DRPIE(X1,Y1,X2,Y2,SA,EA) "
*
```

---

## **Related Subroutines**

[WIN.DRELL](#), [WIN.DRCHORD](#), [WIN.DRPIE](#), [WIN.DRPEN](#), [WIN.DRMOVE](#),  
[WIN.DRRECT](#), [WIN.DRTEXT](#), [WIN.DRPOLY](#)

## **Related Script Commands**

Draw Arc, Draw Brush, Draw Chord, Draw Ellipse, Draw Text, Draw Rect, Draw  
Pie, Draw Pen, Draw Polygon

# WIN.DRBRUSH

## Syntax

WIN.DRBRUSH ( *COLOR*, *STYLE* )

## Description

This subroutine changes the color and style of the brush used in other draw routines. Use WIN.DRBRUSH to set the color and pattern for shapes.

## Parameters

The following table describes the parameters of the WIN.DRBRUSH command:

Parameter	Description
<i>COLOR</i>	See the Color Table in this section.
<i>STYLE</i>	See the Style Table in this section.

## Values for COLOR

Value	Description
<i>RGB</i>	Red, Green and Blue (RGB) are combined in varying amounts for different colors. Minimum amounts of RGB create black; maximum amounts create white.
<i>Color name</i>	A color name, e.g., “Blue”, “Magenta”, “LightGrey”. For a complete list of the color names, see the Color Table under WIN.COLOR in this chapter.

## Values for STYLE

Use the following style names in quotes

Value	Description
<i>Solid</i>	Specifies a solid color.

Value	Description
<i>Null</i>	Specifies no color and ignores the COLOR argument.
<i>Horizontal</i>	Specifies horizontal lines.
<i>Vertical</i>	Specifies vertical lines.
<i>FDiagonal</i>	Specifies diagonal slashes from left to right.
<i>BDiagonal</i>	Specifies diagonal slashes from right to left.
<i>Cross</i>	Specifies cross hatching.
<i>DiagCross</i>	Specifies diagonal cross hatching.

## Example

This example is part of the WIN.DRDEMO demonstration program.

---

```
* Draw white back ground for drawings
*
STATE=' '
CALL WIN.SSTATE(STATE, " ")
CALL WIN.COLOR("BLACK","WHITE")
XTL = 21;YTL = 4; XBR=59; YBR = 14
CALL WIN.BOX(XTL-1,YTL-1,XBR+1,YBR+1,0)
CALL WIN.DRPEN("Black","Solid"," ")
CALL WIN.DRBRUSH(" ","Null")
```

---

## Related Subroutines

[WIN.DRPEN](#)

## Related Script Commands

Draw Brush

# WIN.DRCHORD

## Syntax

**WIN.DRCHORD** ( *X1*, *Y1*, *X2*, *Y2*, *SA*, *EA* )

## Description

This subroutine draws the chord of an ellipse within a rectangle defined by the parameters *X1*,*Y1* to *X2*,*Y2*.

If *X1* and *Y1* are null (""), then the top left of the rectangle is the current graphic cursor position.

Choose the color and style setting of the chord with **WIN.DRPEN**, and fill the interior color and style with **WIN.DRBRUSH**.

## Parameters

The following table describes the parameters of the **WIN.DRCHORD** command:

Parameter	Description
<i>X1</i>	The x position of the top left of the rectangle.
<i>Y1</i>	The y position of the top left of the rectangle.
<i>X2</i>	The x position of the bottom right of the rectangle.
<i>Y2</i>	The y position of the bottom right of the rectangle.
<i>SA</i>	The degree of the start angle.
<i>EA</i>	The degree of the end angle.

## Example

This example is part of the **WIN.DRDEMO** demonstration program.

```
* Arcs etc
1400 GOSUB 5100; * Erase graphic
X = XTL
Y = YTL + 2
*
PRINT @(X,Y) : "Arcs" :
```

```
CALL WIN.DRARC((X+12)*10,Y*10-15,(X+24)*10,Y*10+25,0,45)*  
Y = Y + 4  
PRINT @(X,Y): "Chords":  
CALL WIN.DRCHORD((X+12)*10,Y*10-15,(X+24)*10,Y*10+25,10,115)
```

---

## **Related Subroutines**

[WIN.DRELL](#), [WIN.DRARC](#), [WIN.DRPIE](#), [WIN.DRPEN](#), [WIN.DRMOVE](#),  
[WIN.DRRECT](#), [WIN.DRTEXT](#), [WIN.DRPOLY](#), [WIN.DRBRUSH](#)

## **Related Script Commands**

Draw Arc, Draw Brush, Draw Chord, Draw Ellipse, Draw Text, Draw Rect, Draw Pie, Draw Pen, Draw Polygon

# WIN.DRELL

## Syntax

**WIN.DRELL** ( *X1*, *Y1*, *X2*, *Y2*, *SA*, *EA* )

## Description

This subroutine draws an ellipse within a rectangle defined by the parameters X1,Y1 to X2,Y2. If X1 and Y1 are null (""), then the top left of the rectangle is the current graphic cursor position. Choose the color and style setting of the ellipse with WIN.DRPEN, and fill the interior color and style with WIN.DRBRUSH.

## Parameters

The following table describes the parameters of the WIN.DRELL command:

Parameter	Description
<i>X1</i>	The x position of the top left of the rectangle.
<i>Y1</i>	The y position of the top left of the rectangle.
<i>X2</i>	The x position of the bottom right of the rectangle.
<i>Y2</i>	The y position of the bottom right of the rectangle.
<i>SA</i>	The degree of the start angle.
<i>EA</i>	The degree of the end angle.

## Example

This example is part of the WIN.DRDEMO demonstration program.

```

Y = Y + 4
PRINT @(X,Y):"Ellipses":
CALL WIN.DRELL((X+12)*10,Y*10-3,XBR*10,Y*10+13)
*
MSG = "The lines/shapes are created with:-"
MSG<-1>=" CALL WIN.DRLINE(X1,Y1,X2,Y2)"
MSG=MSG:", CALL WIN.DRRECT(X1,Y1,X2,Y2)"
MSG<-1>=" CALL WIN.DRPOLY(R.POINTS)"
MSG=MSG:" and CALL WIN.DRELL(X1,Y1,X2,Y2)"
GOSUB 5000; * Display info
RETURN

```



\*

---

## **Related Subroutines**

[WIN.DRCHORD](#), [WIN.DRARC](#), [WIN.DRPIE](#), [WIN.DRPEN](#), [WIN.DRMOVE](#),  
[WIN.DRRECT](#), [WIN.DRTEXT](#), [WIN.DRPOLY](#), [WIN.DRBRUSH](#)

## **Related Script Commands**

Draw Arc, Draw Brush, Draw Chord, Draw Ellipse, Draw Text, Draw Rect, Draw  
Pie, Draw Pen, Draw Polygon

# WIN.DRERASE

## Syntax

**WIN.DRERASE** ( *X1*, *Y1*, *X2*, *Y2* )

## Description

This subroutine erases objects drawn by any of the draw routines: WIN.DRARC, WIN.DRBRUSH, WIN.DRCHORD, WIN.DRELL, WIN.DRFONT, WIN.DRLINE, WIN.DRPEN, WIN.DRPIE, WIN.DRPOLY, WIN.DRRECT, WIN.DRTEXT. It also erases images drawn by the WIN.IMAGE subroutine. Use this subroutine when you want to clear the screen of images or drawings.

Note: To erase all objects on an 80 X 24 size screen, use this line: CALL WIN.DRERASE(0,0, 799, 239)

## Parameters

The following table describes the parameters of the WIN.DRERASE command:

Parameter	Description
<i>X1</i>	The x position of the top left of the rectangle.
<i>Y1</i>	The y position of the top left of the rectangle.
<i>X2</i>	The x position of the bottom right of the rectangle.
<i>Y2</i>	The y position of the bottom right of the rectangle.

## Example

This example is part of the WIN.DRDEMO demonstration program.

```
This example is part of the WIN.DRDEMO demonstration program.
* Draw white back ground for drawings
*
STATE=' '
CALL WIN.SSTATE(STATE, " ")
CALL WIN.COLOR("BLACK","WHITE")
XTL = 21;YTL = 4; XBR=59; YBR = 14
CALL WIN.BOX(XTL-1,YTL-1,XBR+1,YBR+1,0)
CALL WIN.DRPEN("Black","Solid","")
CALL WIN.DRBRUSH("","Null")
```

```
*
LOOP
PRINT @(40,22):" ":(40,22):
INPUT DUM:
*
UNTIL DUM = "*" DO
IF NUM(DUM) THEN
BEGIN CASE
CASE DUM = PEN.STYLES; GOSUB 1100
CASE DUM = BRUSH.STYLES; GOSUB 1200
CASE DUM = SHAPES; GOSUB 1300
CASE DUM = ARCS; GOSUB 1400
CASE DUM = TEXT; GOSUB 1500
END CASE
END
REPEAT
*
CALL WIN.SSTATE(" ",STATE)
*
PRINT C22:"Press <CR> to continue ":
INPUT DUM:
*
CALL WIN.DRERASE(XTL*10,YTL*10,XBR*10+9,YBR*10+9)
*
STOP
```

---

## Related Subroutines

[WIN.DRARC](#), [WIN.DRBRUSH](#), [WIN.DRCHORD](#), [WIN.DRELL](#), [WIN.DRFONT](#),  
[WIN.DRLINE](#), [WIN.DRPEN](#), [WIN.DRPIE](#), [WIN.DRPOLY](#), [WIN.DRRECT](#),  
[WIN.DRTEXT](#), [WIN.IMAGE](#)

## Related Script Commands

Draw Erase

# WIN.DRFONT

## Syntax

**WIN.DRFONT** ( *NAME*, *WIDTH*, *DEPTH*, *WEIGHT*, *STYLE* )

## Description

This subroutine sets the font for calls to WIN.DRTEXT. The width and depth are 10 units per character. This means that a width of 20 is twice as wide as the current terminal font.

## Parameters

The following table describes the parameters of the WIN.DRFONT command:

Parameter	Description
<i>NAME</i>	Specifies the name of the font, e.g., Helvetica, Times New Roman, etc.
<i>WIDTH</i>	The width of the font in graphic units. These are logical units allowing pro-portional On an 80 x 24 terminal there are 800 x 240 graphic units.
<i>DEPTH</i>	The width of the font in graphic units. These are logical units allowing pro-portional On an 80 x 24 terminal there are 800 x 240 graphic units.
<i>WEIGHT</i>	The weight of the font. The weight options are: Light Normal Heavy Bold
<i>STYLE</i>	The available font styles are: Italic, Underline, Bold

## Example

This example is part of the WIN.DRDEMO demonstration program.

---

```
CALL WIN.DRFONT("Arial",10,10,"","")
CALL WIN.DRTEXT(X*10,Y*10,"Text can be overlayed in any font")
*
Y = Y + 2
CALL WIN.DRFONT("Times New Roman",20,20,"","")
CALL WIN.DRTEXT(X*10,Y*10,"At any size")
```

```
*
Y = Y + 2
CALL WIN.DRFONT("Courier New",20,20,"","italic")
CALL WIN.DRTEXT(X*10,Y*10,"Italics")
*
Y = Y + 2
CALL WIN.DRFONT("Wingdings",30,30,"","")
CALL WIN.DRTEXT(X*10,Y*10,"ACEGKNZ")
*
Y = Y + 4
CALL WIN.DRFONT("IBSFont",10,10,"","")
CALL WIN.DRTEXT(X*10,Y*10, "Fixed or variable pitch")
```

---

## **Related Subroutines**

[WIN.DRTEXT](#), [WIN.DRPEN](#), [WIN.DRBRUSH](#)

## **Related Script Commands**

Draw Font

# WIN.DRLINE

## Syntax

**WIN.DRLINE** ( *X1*, *Y1*, *X2*, *Y2* )

## Description

This subroutine draws a line from X1,Y1 to X2,Y2. If X1 and Y1 are null (""), then the top left of the rectangle is the current graphic cursor position. Choose the color and style setting of the line with WIN.DRPEN.

## Parameters

The following table describes the parameters of the WIN.DRLINE command:

Parameter	Description
<i>X1</i>	The x position of the top left of the rectangle.
<i>Y1</i>	The y position of the top left of the rectangle.
<i>X2</i>	The x position of the bottom right of the rectangle.
<i>Y2</i>	The y position of the bottom right of the rectangle.

## Example

This example is part of the WIN.DRDEMO demonstration program.

---

```
CALL WIN.DRPEN( "Red",STYLES<J>,W)
CALL WIN.DRLINE((X+12)*10,Y*10+5, XBR*10, Y*10+5)
```

---

## Related Subroutines

[WIN.DRELL](#), [WIN.DRARC](#), [WIN.DRPIE](#), [WIN.DRPEN](#), [WIN.DRMOVE](#),  
[WIN.DRRECT](#), [WIN.DRTEXT](#), [WIN.DRPOLY](#), [WIN.DRBRUSH](#)

## Related Script Commands

Draw Line, Draw Arc, Draw Brush, Draw Chord, Draw Ellipse, Draw Text, Draw Rect, Draw Pie, Draw Pen, Draw Polygon

# WIN.DRMOVE

## Syntax

WIN.DRMOVE ( X, Y )

## Description

This subroutine moves the graphics cursor to a new position specified by X and Y, but does not draw anything. Move the cursor when you want to use another draw subroutine, starting at a specific point on the screen.

## Parameters

The following table describes the parameters of the WIN.DRMOVE command:

Parameter	Description
X	New X co-ordinate of graphics cursor
Y	New Y co-ordinate of graphics cursor

## Example

Move to 100,200

---

```
CALL WIN.DRMOVE ( 100 , 200 )
```

---

# WIN.DRPEN

## Syntax

WIN.DRPEN ( *COLOR*, *STYLE*, *WIDTH* )

## Description

This subroutine changes the color, style and width of the pen used in drawing. Use WIN.DRPEN to set the color for text, lines and outlines of shapes.

## Parameters

The following table describes the parameters of the WIN.DRPEN command:

Parameter	Description
<i>COLOR</i>	See the Color Table in this section.
<i>STYLE</i>	See the Style Table in this section.
<i>WIDTH</i>	Sets the width of the pen. This parameter is available when you use the "Solid" argument for This can be a number from 1 to 100.

## Values for COLOR

Value	Description
<i>RGB</i>	Red, Green and Blue (RGB) are combined in varying amounts for different colors. Minimum amounts of RGB create black; maximum amounts create white.
<i>Color names</i>	A color name, e.g., "Blue", "Magenta", "LightGrey". For a complete list of the color names, see the Color Table under WIN.COLOR in this chapter.



## Values for STYLE

Value	Description
<i>Solid</i>	Specifies a solid color.
<i>Dash</i>	Specifies a dashed line. Specify the color of the area outside the dashes with WIN.DRBRUSH.
<i>Dot</i>	Specifies a dotted line. Specify the color of the area outside the dots with WIN.DRBRUSH.
<i>DashDot</i>	Specifies a line of alternate dashes and dots. Specify the color of the area outside the dashes/dots with WIN.DRBRUSH.
<i>DashDotDot</i>	Specifies a line of dashes and two dots. Specify the color of the area outside the dashes/dots with WIN.DRBRUSH.
<i>Null</i>	Specifies no color, and ignores the COL argument.
<i>InsideFrame</i>	Specifies a line drawn inside the edge of a polygon.

## Example

This example is part of the WIN.DRDEMO demonstration program.

---

```
* Draw white back ground for drawings
*
STATE=' '
CALL WIN.SSTATE(STATE, " ")
CALL WIN.COLOR("BLACK","WHITE")
XTL = 21;YTL = 4; XBR=59; YBR = 14
CALL WIN.BOX(XTL-1,YTL-1,XBR+1,YBR+1,0)
CALL WIN.DRPEN("Black","Solid"," ")
CALL WIN.DRBRUSH(" ","Null")
```

---

## Related Subroutines

[WIN.DRBRUSH](#)

## Related Script Commands

Draw Brush

# WIN.DRPIE

## Syntax

**WIN.DRPIE** ( *X1, Y1, X2, Y2, SA, EA* )

## Description

This subroutine draws a pie segment of an ellipse within a rectangle defined by the parameters X1,Y1 to X2,Y2. If X1 and Y1 are null (""), then the top left of the rectangle is the current graphic cursor position. Choose the color and style setting of the chord with WIN.DRPEN, and fill the interior color and style with WIN.DRBRUSH.

## Parameters

The following table describes the parameters of the WIN.DRPIE command:

Parameter	Description
<i>X1</i>	The x position of the top left of the rectangle.
<i>Y1</i>	The y position of the top left of the rectangle.
<i>X2</i>	The x position of the bottom right of the rectangle.
<i>Y2</i>	The y position of the bottom right of the rectangle.
<i>SA</i>	The degree of the start angle.
<i>EA</i>	The degree of the end angle.

## Example

This example is part of the WIN.DRDEMO demonstration program.

```
* Arcs etc
1400 GOSUB 5100; * Erase graphic
X = XTL
Y = YTL + 2
*
PRINT @(X,Y):"Arcs":
CALL WIN.DRARC((X+12)*10,Y*10-15,(X+24)*10,Y*10+25,0,45)*
Y = Y + 4
PRINT @(X,Y):"Chords":
CALL WIN.DRCHORD((X+12)*10,Y*10-15,(X+24)*10,Y*10+25,10,115)
```

```
*
Y = Y + 4
PRINT @(X,Y):"Pie segments":
CALL WIN.DRPIE((X+12)*10,Y*10-15,(X+24)*10,Y*10+25,0,45)
*
MSG = "The arcs, chords and pie segments are drawn with:-"
MSG<-1> = " WIN.DRARC(X1,Y1,X2,Y2,SA,EA),
WIN.DRCHORD(X1,Y1,X2,Y2,SA,EA) "
MSG<-1> = "and WIN.DRPIE(X1,Y1,X2,Y2,SA,EA) "
*
```

---

## Related Subroutines

[WIN.DRELL](#), [WIN.DRCHORD](#), [WIN.DRPEN](#), [WIN.DRMOVE](#), [WIN.DRRECT](#),  
[WIN.DRTEXT](#), [WIN.DRPOLY](#)

## Related Script Commands

Draw Arc, Draw Brush, Draw Chord, Draw Ellipse, Draw Text, Draw Rect, Draw Pie, Draw Pen

# WIN.DRPOLY

## Syntax

WIN.DRPOLY ( *POINTS* )

## Description

This subroutine draws a polygon by joining the vertices specified in points. Set the outline color and style with WIN.DRPEN, and fill the interior color and style with WIN.DRBRUSH.

## Parameters

The following table describes the parameters of the WIN.DRPOLY command:

Parameter	Description
<i>POINTS</i>	Multi-field dynamic array of vertices of the polygon. The x-coordinate is the first mul-tivalue and the y-coordinate is the second multivalue. A polygon may have from 3 to 5 vertices. The coordinates are based on 10 per character on the terminal screen, with the top left at 0,0.

## Example

This example is part of the WIN.DRDEMO demonstration program.

---

```
*
Y = Y + 4
PRINT @(X,Y): "Polygons":
POINTS= ' '
W = INT((XBR-X-12)/3)-2
X1 = X + 12
X1 = X1 * 10
Y1 = Y * 10 + 5
W = W * 5
D = 18
*
FOR J = 3 TO 5
  A = 0
  INC = INT(360/J)
  FOR K = 1 TO J
```

```
POINTS<K,1> = X1 + INT(W * COS(A))
POINTS<K,2> = Y1 + INT(D * SIN(A))
A = A + INC
NEXT K
CALL WIN.DRPOLY(POINTS)
X1 = X1 + W * 2 + 10
NEXT J
*
```

---

## Related Subroutines

[WIN.DRRECT](#), [WIN.DRPEN](#), [WIN.DRBRUSH](#)

## Related Script Commands

Draw Polygon

# WIN.DRRECT

## Syntax

**WIN.DRRECT** ( *X1*, *Y1*, *X2*, *Y2* )

## Description

This subroutine draws a rectangle defined by the parameters X1,Y1 to X2,Y2. If X1 and Y1 are null (""), then the top left of the rectangle is the current graphic cursor position. Choose the color and style setting of the rectangle with WIN.DRPEN, and fill the interior color and style with WIN.DRBRUSH.

## Parameters

The following table describes the parameters of the WIN.DRRECT command:

Parameter	Description
<i>X1</i>	The x position of the top left of the rectangle.
<i>Y1</i>	The y position of the top left of the rectangle.
<i>X2</i>	The x position of the bottom right of the rectangle.
<i>Y2</i>	The y position of the bottom right of the rectangle.

## Example

This example is part of the WIN.DRDEMO demonstration program.

```
Y = Y + 2
PRINT @(X,Y):"Rectangles":
CALL WIN.DRRECT((X+12)*10,Y*10-3,XBR*10,Y*10+13)
```

## Related Subroutines

[WIN.DRELL](#), [WIN.DRCHORD](#), [WIN.DRPEN](#), [WIN.DRMOVE](#), [WIN.DRTEXT](#),  
[WIN.DRPOLY](#)

## **Related Script Commands**

Draw Arc, Draw Brush, Draw Chord, Draw Ellipse, Draw Text, Draw Rect, Draw Pie



# WIN.DRTEXT

## Syntax

**WIN.DRTEXT** ( *XI*, *YI*, *TEXT* )

## Description

This subroutine draws text within a rectangle on the screen. Use this in conjunction with WIN.DRPEN to set the color and WIN.DRFONT to set the font. Fill in the rectangle in a color set with WIN.DRBRUSH.

## Parameters

The following table describes the parameters of the WIN.DRTEXT command:

Parameter	Description
<i>XI</i>	The column position of the text.
<i>YI</i>	The row position of the text.
<i>TEXT</i>	The text to display.

## Example

This example is part of the WIN.DRDEMO demonstration program.

```
CALL WIN.DRFONT("Arial",10,10,"","")
CALL WIN.DRTEXT(X*10,Y*10,"Text can be overlayed in any font")
*
Y = Y + 2
CALL WIN.DRFONT("Times New Roman",20,20,"","")
CALL WIN.DRTEXT(X*10,Y*10,"At any size")
*
Y = Y + 2
CALL WIN.DRFONT("Courier New",20,20,"","italic")
CALL WIN.DRTEXT(X*10,Y*10,"Italics")
*
Y = Y + 2
CALL WIN.DRFONT("Wingdings",30,30,"","")
CALL WIN.DRTEXT(X*10,Y*10,"ACEGKNZ")
*
Y = Y + 4
CALL WIN.DRFONT("IBSFont",10,10,"","")
CALL WIN.DRTEXT(X*10,Y*10, "Fixed or variable pitch")
```

```
*  
MSG = "The font and its size, weight and style is first chosen with:-"  
MSG<-1> = " CALL WIN.DRFONT(NAME,WIDTH,DEPTH,WEIGHT,STYLE)"  
MSG<-1> = "and then drawn with:-"  
MSG<-1> = " CALL WIN.DRTEXT(X,Y,TEXT)"  
*
```

---

## Related Subroutines

[WIN.DRPEN](#), [WIN.DRFONT](#), [WIN.DRBRUSH](#)

## Related Script Commands

Draw Text

# WIN.EDIT

## Syntax

WIN.EDIT *FILE*, *ITEM*

## Description

Run this program at ECL to run the wIntegrate editor to edit a host program item.

## Parameters

The following table describes the parameters of the WIN.EDIT command:

Parameter	Description
<i>FILE</i>	The name of the host file that contains the item to be edited
<i>ITEM</i>	Host item to edit

# WIN.EFFECT

## Syntax

WIN.EFFECT ( *NAME* )

## Description

This subroutine sets the effect of the text that follows.

## Parameters

The following table describes the parameters of the WIN.EFFECT command:

Parameter	Description
<i>NAME</i>	The name of the effect to use for the text. See the Effects list below for available effects. Any of the effects can be used individually, or concatenated as a string, e.g. DimReverse, BoldUnderline.

## Values for NAME

Use one or more names concatenated from the table

Value	Description
<i>Normal</i>	Normal intensity
<i>Dim</i>	Low intensity
<i>Reverse</i>	Foreground and background colors are reversed
<i>Underline</i>	Underlined
<i>Flash</i>	Blinking text
<i>Bold</i>	High intensity
<i>Secret</i>	Invisible

## Example

This example is part of the WIN.CEDEMO demonstration program.

---

```
* Dummy customer maintenance program
200 CALL WIN.EFFECT("NORMAL")
PRINT @(0,3):@(-3)
PRINT @(30,3):"Customer maintenance":
PRINT @(5,5):"Id:":
PRINT @(5,7):"1. Name:":
PRINT @(5,10):"2. Address:":
PRINT @(5,15):"3. Contact:"
*
PRINT @(25,5):
CALL WIN.EFFECT("REVERSE")
PRINT " ":
CALL WIN.EFFECT("NORMAL")
PRINT @(25,7):
CALL WIN.EFFECT("REVERSE")
PRINT " ":
CALL WIN.EFFECT("NORMAL")
FOR J = 10 TO 13
PRINT @(25,J):
CALL WIN.EFFECT("REVERSE")
PRINT " ":
CALL WIN.EFFECT("NORMAL")
NEXT J
PRINT @(25,15):
CALL WIN.EFFECT("REVERSE")
PRINT " ":
CALL WIN.EFFECT("NORMAL")
PRINT @(52,7):"This area has some useful":
PRINT @(52,8):"or important static text":
PRINT @(52,9):"diplayed in it.":
RETURN
```

---

## Related Subroutines

[WIN.COLOR](#)

## Related Script Commands

Color

# WIN.EFILL

## Syntax

**WIN.EFILL** ( *NAME*, *LEFT*, *TOP*, *RIGHT*, *BOTTOM* )

## Description

This subroutine fills an area of the screen with an effect.

## Parameters

The following table describes the parameters of the WIN.EFILL command:

Parameter	Description
<i>NAME</i>	The name of the effect. See the Effects list below.
<i>LEFT</i>	The left column of the area to fill.
<i>TOP</i>	The top of the area to fill.
<i>RIGHT</i>	The right column of the area to fill.
<i>BOTTOM</i>	The bottom column of the area to fill.

## Values for NAME

Value	Description
<i>Normal</i>	Normal intensity
<i>Dim</i>	Low intensity
<i>Reverse</i>	Foreground and background colors are reversed
<i>Underline</i>	Underlined
<i>Flash</i>	Blinking text
<i>Bold</i>	High intensity
<i>Secret</i>	Invisible

## Example

This example is part of the WIN.CEDEMO demonstration program.

---

```
* Emphasise prompt text
260 TEXT = "Emphasis for the title and field text can be added by
setting up"
TEXT = TEXT : " Bold to be a raised style and then paint the bold
effect"
TEXT = TEXT : " on the drawn screen."
GOSUB 2000; * Print text
GOSUB 1000; * Continue prompt
CALL WIN.SETEFFECT("Bold","Black","LightGray","Raised",0)
* Have to temporarily put scrollregion back to whole screen
CALL WIN.COMSUB("Screen ScrollRegion")
CALL WIN.EFILL("Bold", 30,3,49,3)
CALL WIN.EFILL("Bold", 5,5,15,5)
CALL WIN.EFILL("Bold", 5,7,15,7)
CALL WIN.EFILL("Bold", 5,10,15,10)
CALL WIN.EFILL("Bold", 5,15,15,15)
```

---

## Related Subroutines

[WIN.EFFECT](#)

## Related Script Commands

Display EffectFill

# WIN.EI

## Syntax

**WIN.EI** ( *X*, *Y*, *LEN*, *NLINES*, *NDISPLAY*, *OPTS*, *VALID*, *EXITS*, *VALUE*, *ESC* )

## Description

This subroutine allows local editing on the host and returns the value.

You may finish the input in one of these ways:

Entering a carriage return - the default method.

Entering the Esc key (in this case the value is not updated)

Entering this combination: Ctrl-F Shift-End.

Defining an exit key in the EXITS parameter.

The WIN.EI2 subroutine can be used for more control of the input.

## Parameters

The following table describes the parameters of the WIN.EI command:

Parameter	Description
<i>X</i>	The column position for the input.
<i>Y</i>	The row position for the input.
<i>LEN</i>	The length of the input.
<i>NLINES</i>	The number of lines of the input.
<i>NDISPLAY</i>	The number of lines to display at one time.
<i>OPTS</i>	See the Options table below.
<i>VALID</i>	See the Validations table below.
<i>EXITS</i>	Specifies the characters to exit the routine. Set to "" a carriage return exit.



Parameter	Description
<i>VALUE</i>	The default and returned values.
<i>ESC</i>	This variable is set to 1 if the input was exited with the escape key otherwise it is set to 0.

## Values for OPTS

Options are used in conjunction with the validation parameter. The options for the WIN.EI subroutine can be one or more of the option letters, concatenated and inside of quotes, e.g., "AM":

Value	Description
<i>M</i>	Makes the validation parameter mandatory.
<i>U</i>	Converts input into upper case.
<i>B</i>	Beeps when an error occurs.
<i>D</i>	Shows the error on a dialog box.
<i>S</i>	Displays the error on the status bar.
<i>A</i>	Causes an automatic carriage return. The field is automatically finished when the last character is typed.
<i>V</i>	Validates on exit by exit_chars.
<i>P(char)</i>	Pad the background the a character.

## Values for VALID

Validation checks the input against the specified pattern. During input, validation checks each character for specified patterns. For instance, if a field must have three alpha characters, a dash, and two numeric characters, the validation pattern would look like: "3A`-'2N". You can also specify minimum or maximum occurrences of a pattern by numbers in parentheses (min(max)).

Value	Description
<i>N</i>	A numeric character.
<i>A</i>	An alpha character.

Value	Description
"x"	Any character.
"text"	Text inside of quotes. To designate a choice of characters or text, separated the choices by a vertical bar ( ). Example: "'EXIT' ' ' 'QUIT'" Example: "IN   '*'"

## Example

This example is the entire WIN.EIDEMO demonstration program. To understand how this subrou-tine

To understand how this subroutine works, enter WIN.EIDEMO from your database prompt. The program demonstrates how WIN.EI creates edit input fields with mandatory input parameters.

```
* WIN.EIDEMO
* Demonstration of the Edit Input command using the WIN.EI subroutine
* Compile for: GENERIC AP MD ME PI PR SQ UD UL UP UV UC IN
* Copyright (c) 1991-93. Impact Business Systems
*
DIM R.TEST(7)
MAT R.TEST = " "
*
PROMPT ' '
SCREEN = @(-1):"WIN.EIDEMO"
SCREEN = SCREEN : @(20,0):"Demonstration of WIN.EI subroutine"
SCREEN = SCREEN : @(0,2):"1. Any entry"
SCREEN = SCREEN : @(0,4):"2. Mandatory Entry"
SCREEN = SCREEN : @(0,6): "3. Pure Numeric"
SCREEN = SCREEN : @(0,8): "4. 1 to 3 Alpha '-' 2 digits"
SCREEN = SCREEN : @(0,10):"5. Numeric, 'EXIT' or 'HI'"
SCREEN = SCREEN : @(0,12):"6. Text Entry"
*
PRINT SCREEN:
C22 = @(0,22):@(-4)
ESC = ' '
*
LOOP
PRINT C22:"Press number (without <CR>) or * to end":
VALUE = ' '
EXIT = "*"
CALL WIN.EI(40,22,1,1,1, "AM","N", EXIT, VALUE, ESC)
IF ESC OR EXIT # " " THEN VALUE = '*'
UNTIL VALUE = '*' DO
```

```
BEGIN CASE
CASE VALUE = 1; GOSUB 100; * Test 1
CASE VALUE = 2; GOSUB 200; * Test 2
CASE VALUE = 3; GOSUB 300; * Test 4
CASE VALUE = 4; GOSUB 400; * Test 4
CASE VALUE = 5; GOSUB 500; * Test 5
CASE VALUE = 6; GOSUB 600; * Test 6
END CASE
REPEAT
*
STOP
*
100 CALL WIN.EI(30,2,10,"","","",""," ", R.TEST(1), ESC)
RETURN
*
200 CALL WIN.EI(30,4,15, "","","M",""," ", R.TEST(2), ESC)
RETURN
*
300 CALL WIN.EI(30,6,8,"","","","ON", " ", R.TEST(3), ESC)
RETURN
*
400 CALL WIN.EI(30,8,25,"","","","U","1-3A'-'2N", " ", R.TEST(4), ESC)
RETURN
*
500 CALL WIN.EI(30,10,15,"","","","U","1NON|'EXIT'|'HI'", " ", R.TEST(5),
ESC)
RETURN
*
600 CALL WIN.EI(30,12,25,12,3,"","","",R.TEST(6), ESC)
RETURN
*
END
```

---

## Related Subroutines

[WIN.LI](#), [WIN.EI2](#)

## Related Script Commands

EditInput

# WIN.EI2

## Syntax

**WIN.EI2** ( *X*, *Y*, *LEN*, *NLINES*, *PARAMS*, *VALUE*, *EXITKEY*, *STATE* )

## Description

The WIN.EI2 host subroutine is used input data from a user and allow them simple editing of the input before it is sent back to the host.

If the Edit Input Keys parameter is set in Setup Preferences then the Soft Edit keys are used for navigation in the input.

You may finish the input in one of these ways:

Entering a carriage return (on the last line of input for multiple line inputs)

Entering the Esc character. In this case the variable will not be updated unless CHAR(27) is set to be an exit key.

Entering this combination: Ctrl-F Shift-End.

Entering a character defined in the exit chars field of the parameter.

Pressing a key defined in the exit\_keys field of the parameters

The WIN.EI2 subroutine is an enhanced version of the WIN.EI subroutine.

## Parameters

The following table describes the parameters of the WIN.EI2 command:

Parameter	Description
<i>X</i>	The column position for the input.
<i>Y</i>	The row position for the input
<i>LEN</i>	The length of the input.
<i>NLINES</i>	The number of lines of the input.

Parameter	Description
<i>PARAMS</i>	A dynamic array specifying additional parameters for the input. See below for details. If set to "" the default values below are used.
<i>VALUE</i>	The value to be input. On entry to the routine this is the initial value to be used.
<i>EXITKEY</i>	The key value or name used to exit the routine. This will be "" if the field was exited with return, CHAR(27) if the escape key was used or the name of value of the exit key specified in the PARAMS argument. If the return value is CHAR(27) the variable will not have been updated.
<i>STATE</i>	This is a dynamic array containing the state of the input on exit. Passing this value back to the input routine next time the field is edited will restore the editing position.

## Fields for PARAMS

The first 4 fields of the array are unused.

Field	Description
5	The number of columns to display. Default is the LEN parameter
6	The number of lines to display. Default is the NLINES parameter
7	Options. See the OPTS parameter of WIN.EI
8	Validation. See the VALIDS parameter of WIN.EI
9	Exit chars. Additional characters that will cause the input to exit.
10	Exits keys. A value-mark separated list of the keys that cause the input to exit. The key names can be found by looking at the names displayed in the Setup Keyboard menu option. e.g. "Escape":@VM:"F1"

## Example

See the WIN.EI2DEMO host program

## Related Subroutines

[WIN.EI](#)

## Related Script Commands

EditInput2

## Version

5.1.2 Original version

# WIN.EVAL

## Syntax

**WIN.EVAL** ( *EXPRESSION*, *RESULT* )

## Description

Evaluates a script expression and returns the result

## Parameters

The following table describes the parameters of the WIN.EVAL command:

Parameter	Description
<i>EXPRESSION</i>	The script expression to be evaluated
<i>RESULT</i>	Variable to hold the result

# WIN.EXPORT

## Syntax

**WIN.EXPORT** ( *PCFILE*, *FILE*, *ITEMS*, *FIELDS*, *OPTS*, *STATUS* )

## Description

This subroutine exports a PC file to the host computer. It calls the RunExportFile menu command, so to understand this subroutine, you can look at the Run Export dialog box.

## Parameters

The following table describes the parameters of the WIN.EXPORT command:

Parameter	Description
<i>PCFILE</i>	The full path name of the PC file to export
<i>FILE</i>	The name of the host file to hold the export file contents
<i>ITEMS</i>	The item names to use on the exported file.
<i>FIELDS</i>	The field names of the fields to export
<i>OPTS</i>	Additional modifications for other file transfer options. See the table below
<i>STATUS</i>	The returned status. This returns are "OK", "Cancel" or "Error"

## Fields for OPTS

Field	Description
<i>I</i>	Format - A file format for the DOS file. Valid formats are ASC, CRD, CSV, DBF, FIX, MRG, RAW, WKS, WK1, and XLS. The default is the extension in the FILE parameter.



Field	Description
2	Overwrite - Specifies options of overwriting an existing file. This can be "Yes", "No", or "Combine". The default is "Yes".
3	Multiple disks - True to export from multiple floppy disks
6	Translate - This option is available to change characters as they are exported, according to the Translation. This can be "None", "Ascii" or "All". The default is "Ascii".
7	Translation - The default is "255,\r\n\f\r\n,\254,\r\n". If you want to use a format other than ASCII, you must go into the Export dialog box and delete the Translation definition.
8	AutoExit - Exits the Export monitor at the end of the transfer. The default is "True".
9	Inform - Notify the user when export is finished. The default is "False".
10	Timeout - The maximum number of seconds to wait for a response from the host. The default is "5".
11	Retries - The maximum number of retries. The default is "3".
12	Use formatting information - Set to 1 to use the formatting information from the host dictionary items in the export.

## Example

Export Orders spread sheet to NEW.ORDERS host file

---

```
STATUS = ""
OPTS = ""
CALL WIN.EXPORT("c:\excel\orders.xls", "NEW.ORDERS", "#", "*", OPTS,
STATUS)
```

---

## Related Subroutines

[WIN.IMPORT](#), [WIN.SPOOL](#), [WIN.TRANSFER](#)

## Related Script Commands

Dialog RunImportFile, Set

## Version

4.1 New options OPTS<12>

# WIN.FKEY

## Syntax

**WIN.FKEY** ( *KEYS*, *DEFS* )

## Description

This subroutine programs function keys, as in wIntegrate's Setup Keyboard. If the *KEYS* parameter is "" (null), WIN.FKEY assumes that *DEFS* contains definitions starting with the F1 key.

## Parameters

The following table describes the parameters of the WIN.FKEY command:

Parameter	Description
<i>KEYS</i>	A multi-field dynamic array that matches <i>DEFS</i> .
<i>DEFS</i>	Key definitions that can contain function key or names. See the table below.

## Values for *KEYS*

The following key codes can be used

Value	Description
<i>1-12</i>	F1 to F12
<i>13-24</i>	Shift F1 to Shift F12
<i>25-36</i>	Control F1 to Control F12
<i>37-48</i>	Control Shift F1 to Control Shift F12

## Example

Program keys F1, F3 and Shift\_F2

```
KEYS = "";DEFS = ""  
KEYS<1> = 1;DEFS<1> = "Key 1"  
KEYS<2> = 3;DEFS<2> = "Key 3"  
KEYS<3> = 14;DEFS<3> = "Keys Shift F2"
```

```
CALL WIN.FKEY(KEYS,DEFS)
* Set shift F1 to run wIntegrate Help
CALL WIN.FKEY("Shift_F1", "\mInvoke HelpIndex")
```

---

# WIN.FSCRIPT

## Syntax

**WIN.FSCRIPT** ( *SCRIPT* )

## Description

Sends a script to the PC to be executed.

This is similar to the WIN.HSCRIPT routine but uses a different technique to transfer the script to the PC. In most cases this routine is faster than WIN.HSCRIPT.

## Parameters

The following table describes the parameters of the WIN.FSCRIPT command:

Parameter	Description
<i>SCRIPT</i>	The text of the script to be run

## Related Subroutines

[WIN.HSCRIPT](#)

# WIN.FTPCLOSE

## Syntax

WIN.FTPCLOSE

## Description

This subroutine closes a file which was previously opened by WIN.FTPOPEN.

## Parameters

None

## Related Subroutines

[WIN.FTPOPEN](#)

## Version

4.1 Original

# WIN.FTPCON

## Syntax

WIN.FTPCON ( *SERVERNAME*, *USERNAME*, *PASSWORD*, *RESP* )

## Description

This command connects wIntegrate to a remote FTP server. You can only have one FTP connection per session.

## Parameters

The following table describes the parameters of the WIN.FTPCON command:

Parameter	Description
<i>SERVERNAME</i>	The name of IP address of the FTP server
<i>USERNAME</i>	The user name on the server. Set this and password to "" to log on as an anonymous user (where your username is "anonymous" and your password is your email address)
<i>PASSWORD</i>	The password required for this user to logon to the FTP server
<i>RESP</i>	"" if logon was successful, otherwise an error message

## Version

4.1 Original version

# WIN.FTPDEL

## Syntax

WIN.FTPDEL ( *FILENAME* )

## Description

This command deletes a file on an FTP server. Use WIN.FTPRMDIR to delete a directory.

## Parameters

The following table describes the parameters of the WIN.FTPDEL command:

Parameter	Description
<i>FILENAME</i>	The name of the file to delete from the FTP server

## Version

4.1 Original version



# WIN.FTPDIR

## Syntax

WIN.FTPDIR ( *DIRNAME*, *EXISTS* )

## Description

This command check is a directory exists on a FTP server.

## Parameters

The following table describes the parameters of the WIN.FTPDIR command:

Parameter	Description
<i>DIRNAME</i>	The name of the directory to check
<i>EXISTS</i>	Set to 1 if directory exist, otherwise 0

## Version

4.1 Original version

# WIN.FTPDISC

## Syntax

WIN.FTPDISC

## Description

This subroutine disconnects the FTP session.

## Parameters

None

## Related Subroutines

[WIN.FTPCON](#)

## Version

4.0.4 Original

# WIN.FTPFILE

## Syntax

**WIN.FTPFILE** ( *FILENAME*, *EXISTS* )

## Description

This command checks if a file exists on a FTP server.

## Parameters

The following table describes the parameters of the WIN.FTPFILE command:

Parameter	Description
<i>FILENAME</i>	The name of the file to check
<i>EXISTS</i>	Set to 1 if file exists, otherwise 0

## Version

4.1 Original version

# WIN.FTPGET

## Syntax

WIN.FTPGET ( *REMOTE.FILE*, *PC.FILE*, *TYPE*, *RESP* )

## Description

This command copies a file from the FTP server to the local PC.

## Parameters

The following table describes the parameters of the WIN.FTPGET command:

Parameter	Description
<i>REMOTE.FILE</i>	The name of the file on the FTP server
<i>PC.FILE</i>	The name of the file on the PC.
<i>TYPE</i>	The type of file to retrieve: 0 - ASCII text file (the default). Select the this option only if the file is a readable ASCII file. This ensures the end-of-line characters are converted appropriately between local and remote file systems.1 - Binary file.
<i>RESP</i>	"" if the file transferred ok, otherwise an error message.

## Version

4.1 Original version

# WIN.FTPGETDR

## Syntax

WIN.FTPGETDR ( *DIRECTORY* )

## Description

This command returns the name of the current directory on the FTP server.

## Parameters

The following table describes the parameters of the WIN.FTPGETDR command:

Parameter	Description
<i>DIRECTORY</i>	Returned directory name

## Version

4.1 Original version

# WIN.FTPINFO

## Syntax

WIN.FTPINFO ( *FILENAME*, *INFO* )

## Description

This command returns information about the specified file on the FTP server. It seems that not all the information returned by this function can be relied upon (the information returned depends on the remote FTP site).

## Parameters

The following table describes the parameters of the WIN.FTPINFO command:

Parameter	Description
<i>FILENAME</i>	The name of the file to retrieve information on
<i>INFO</i>	A text string with the information seperated by spaces in the order: length,date,time,attributes

## Version

4.1 Original version

# WIN.FTPLIST

## Syntax

**WIN.FTPLIST** ( *FILESPEC*, *OPTS*, *RESP* )

## Description

This command returns a list of the files and/or subdirectories on an FTP server matching the criteria that you specify in a dynamic array.

## Parameters

The following table describes the parameters of the WIN.FTPLIST command:

Parameter	Description
<i>FILESPEC</i>	The files to return. This can be a wild card expression to match files
<i>OPTS</i>	Specifies other matching options. See the following Options table for more information.
<i>RESP</i>	A dynamic array of the files/directories which match the specification

## Values for FILESPEC

Value	Description
<i>0</i>	Returns files only.
<i>1</i>	Returns files and directories.
<i>2</i>	Returns files, and includes directories in square brackets
<i>3</i>	Returns files, and includes directories prefixed with "&gt;"
<i>4</i>	Returns only subdirectories.
<i>32</i>	Do not include current directory.

## Values for OPTS

The options field specifies which files or subdirectories to match. The default option is 0. The options 4 and 32 can be used alone or added to other option numbers.

Value	Description
<i>1</i>	Returns files and directories.
<i>2</i>	Returns files, and includes directories in square brackets
<i>3</i>	Returns files, and includes directories prefixed with ">"
<i>4</i>	Returns only subdirectories.
<i>32</i>	Do not include current directory.

## Version

4.1 Original version



# WIN.FTPMKDIR

## Syntax

WIN.FTPMKDIR ( *DIRNAME* )

## Description

This command creates a directory on the FTP server.

## Parameters

The following table describes the parameters of the WIN.FTPMKDIR command:

Parameter	Description
<i>DIRNAME</i>	The name of the directory to create

## Version

4.1 Original version

# WIN.FTPOPEN

## Syntax

WIN.FTPOPEN ( *FILENAME*, *OPTS*, *RESP* )

## Description

This command opens a file on the FTP server for reading and writing. Only one file can be opened at a time and only calls to the WIN.FTPREAD and WIN.FTPWRITE FTP routines should be made until WIN.FTPCLOSE is called. The file must be closed when finished with with WIN.FTPCLOSE

## Parameters

The following table describes the parameters of the WIN.FTPOPEN command:

Parameter	Description
<i>FILENAME</i>	The name of the file to open
<i>OPTS</i>	Specifies the file and operation type. You cannot open a remote file for both reading and writing, so the options must reflect which operation is intended. See the table below
<i>RESP</i>	1 if file could be opened, 0 otherwise.

## Values for OPTS

Value	Description
1	Read Binary.
2	Write ASCII text. Select this option only if you are reading or writing readable ASCII text - this ensures the end-of-line characters are converted appropriately between local and remote file systems.
3	Write Binary.

## **Version**

4.0.4 Original version

# WIN.FTPPOS

## Syntax

WIN.FTPPOS ( *POSITION* )

## Description

This command moves the position in the currently open file to the specified location.

## Parameters

The following table describes the parameters of the WIN.FTPPOS command:

Parameter	Description
<i>POSITION</i>	The new position in the file. 0 is the start of the file and -1 can be used for the end of the file.

## Version

4.1 Original version

# WIN.FTPPUT

## Syntax

WIN.FTPPUT ( *PC.FILE*, *REMOTE.FILE*, *TYPE*, *RESP* )

## Description

This command copies a file from the local PC to the FTP server.

## Parameters

The following table describes the parameters of the WIN.FTPPUT command:

Parameter	Description
<i>PC.FILE</i>	The name of the file on the PC.
<i>REMOTE.FILE</i>	The name of the file on the FTP server
<i>TYPE</i>	The type of file to send:0 - ASCII text file (the default). Select the this option only if the file is a readable ASCII file. This ensures the end-of-line characters are converted appropriately between local and remote file systems.1 - Binary file.
<i>RESP</i>	"" if the file transferred ok, otherwise an error message.

## Version

4.1 Original version

# WIN.FTPREAD

## Syntax

**WIN.FTPREAD** ( *VALUE*, *MAX.BYTES*, *CONVERT* )

## Description

This command reads bytes from the currently open file on the FTP server. The file must have been opened for read access with WIN.FTPOPEN.

## Parameters

The following table describes the parameters of the WIN.FTPREAD command:

Parameter	Description
<i>VALUE</i>	Variable to receive the bytes read
<i>MAX.BYTES</i>	The maximum number of bytes to read. Set to "" for the default value of approximately 30K.
<i>CONVERT</i>	Data conversion to use on reply. See table below

## Values for CONVERT

Value	Description
""	no conversion
"TEXT"	convert cr/lf to field mark
"HEX"	return each byte in the file as a 2 digit hex number
"FT"	return in wIntegrate's file transfer format

## Version

4.1 Original version

# WIN.FTPRMDIR

## Syntax

WIN.FTPRMDIR ( *DIRNAME* )

## Description

This command removes (deletes) the specified directory from the FTP server.

## Parameters

The following table describes the parameters of the WIN.FTPRMDIR command:

Parameter	Description
<i>DIRNAME</i>	The name of the directory to delete

## Version

4.1 Original version

# WIN.FTPSCR

## Syntax

WIN.FTPSCR ( *SCRIPTNAME*, *OK* )

## Description

This command loads a script from the FTP server and runs it.

## Parameters

The following table describes the parameters of the WIN.FTPSCR command:

Parameter	Description
<i>SCRIPTNAME</i>	The file name of the script to run on the server
<i>OK</i>	1 if the script started ok, otherwise 0

## Version

4.1 Original version



# WIN.FTPSCRIPT

## Syntax

**WIN.FTPSCRIPT** ( *SCRIPTNAME*, *MAT FTP.OPTS*, *OK* )

## Description

This command transfers the script to the PC using FTP and then runs it. It requires a connection to be made to the FTP server with WIN.FTPCON and a temporary file on the host which can also be accessed through FTP.

## Parameters

The following table describes the parameters of the WIN.FTPSCRIPT command:

Parameter	Description
<i>SCRIPTNAME</i>	The file name of the script to run on the server
<i>MAT FTP.OPTS</i>	10 element array of additional information required for the transfer. See below
<i>OK</i>	1 if the script started ok, otherwise 0

## Fields for MAT FTP.OPTS

Field	Description
<i>1</i>	Handle to temporary file
<i>2</i>	Path to temporary file for the FTP server
<i>3-10</i>	Reserved. Set to ""

## Example

Run Excel using the "Run" script command

---

```
DIM OPTS(10)
MAT OPTS = ""
OPEN "", "TEMP" TO OPTS(1) ELSE STOP "No TEMP file"
OPTS(2) = "/usr/uv/account/TEMP";* Path to server
CALL WIN.FTPCON("server","username","password", RESP)
```

```
IF RESP = "" THEN  
CALL WIN.FTPSCRPT("Run 'excel.exe'", MAT_OPTS, OK)  
CALL WIN.FTPDISC  
END
```

---

## Related Subroutines

[WIN.FTPCON](#), [WIN.FTPSCR](#)

## Version

4.1 Original version

# WIN.FTPSETDR

## Syntax

WIN.FTPSETDR ( *DIRNAME* )

## Description

This command sets the current directory on the FTP server. This is useful as it allows the FTP routines to then refer to filenames and sub-directories without having to specify their full paths.

## Parameters

The following table describes the parameters of the WIN.FTPSETDR command:

Parameter	Description
<i>DIRNAME</i>	The directory name to be set as the current directory

## Version

4.1 Original version

# WIN.FTPSETV

## Syntax

WIN.FTPSETV ( *VARNAME*, *VALUE*, *MAT FTP.OPTS*, *OK* )

## Description

This command set the script variable or dialog box property with the specified value using FTP. It requires a connection to be made to the FTP server with WIN.FTPCON and a temporary file on the host which can also be accessed through FTP.

## Parameters

The following table describes the parameters of the WIN.FTPSETV command:

Parameter	Description
<i>VARNAME</i>	Name of the script variable to set. This can also be the name of a dialog box property.
<i>VALUE</i>	The value to set the variable
<i>MAT FTP.OPTS</i>	10 element array of additional information required for the transfer. See below
<i>OK</i>	1 if the variable was transferred ok, otherwise 0

## Fields for MAT FTP.OPTS

Field	Description
<i>1</i>	Handle to temporary file
<i>2</i>	Path to temporary file for the FTP server
<i>3-10</i>	Reserved. Set to ""

## Example

Put the contents of the record REC in the grid

---

```
*
DIM OPTS(10)
MAT OPTS = ""
OPEN "", "TEMP" TO OPTS(1) ELSE STOP "No TEMP file"
OPTS(2) = "/usr/uv/account/TEMP";* Path to server
*
GOSUB 100 ;* Create the dialog box with the Grid on it
GOSUB 200 ;* Set up the data for the grid in REC
*
CALL WIN.FTPCON("server","username","password", RESP)
IF RESP = "" THEN
* Set the grids data property to display the data
CALL WIN.FTPSETV("GridTest.Grid1.Data",REC, MAT OPTS, RESP)
IF RESP = 0 THEN ERR = 7
END
CALL WIN.FTPDISC
*
IF ERR THEN PRINT "UNABLE TO LOAD, ERROR = ":ERR;STOP
*
CALL WIN.DBSHOW(DLGNAME, 0, ERR)
IF ERR THEN PRINT "UNABLE TO SHOW, ERROR = ":ERR;STOP
```

---

## Version

### 4.1 Original version

# WIN.FTPWRITE

## Syntax

WIN.FTPWRITE ( *VALUE*, *CONVERT* )

## Description

This command writes bytes to the currently open file on the FTP server. The file must have been opened for write access with WIN.FTPOPEN.

## Parameters

The following table describes the parameters of the WIN.FTPWRITE command:

Parameter	Description
<i>VALUE</i>	Data to write to the ftp file
<i>CONVERT</i>	Data conversion to use on the <i>VALUE</i> before writing it to the server file.

## Values for CONVERT

Value	Description
<i>""</i>	no conversion
<i>"TEXT"</i>	convert cr/lf to field mark
<i>"HEX"</i>	return each byte in the file as a 2 digit hex number
<i>"FT"</i>	return in wIntegrate's file transfer format

## Related Subroutines

[WIN.FTPOPEN](#)

## **Version**

### 4.1 Original version

# WIN.FTTOASC

## Syntax

WIN.FTTOASC ( *VALUE* )

## Description

Converts a seven-bit File Transfer string to ASCII format. The file transfer format represents control codes and other special characters as one or two seven-bit values. The seven-bit data can be sent across any communications line.

## Parameters

The following table describes the parameters of the WIN.FTTOASC command:

Parameter	Description
<i>VALUE</i>	The string to convert.

## Related Subroutines

[WIN.ASCTOFT](#)



# WIN.GETDATA

## Syntax

**WIN.GETDATA** ( *VARNAME*, *VALUE* )

## Description

This subroutine returns the value or contents of a global variable. The variable can contain ASCII characters less than 32 and greater than 126, including tabs and carriage returns. If your variables contain only ASCII characters between 32 and 126, use the faster WIN.GETVAR subroutine.

## Parameters

The following table describes the parameters of the WIN.GETDATA command:

Parameter	Description
<i>VARNAME</i>	The name of the variable to return.
<i>VALUE</i>	The value of the variable specified in VARNAME.

## Example

Find and print the current value of UserName

---

```
Find and print the current value of UserName
CALL WIN.GETDATA("UserName", VALUE)
PRINT "The current UserName is ": VALUE
```

---

## Related Subroutines

[WIN.GETVAR](#), [WIN.SETDATA](#), [WIN.SETVAR](#), [WIN.DBGET](#), [WIN.DBSET](#),  
[WIN.SETLIST](#)

# WIN.GETLIST

## Syntax

**WIN.GETLIST** ( *VARNAME*, *VALUE* )

## Description

Since PC items generally have lines separated by carriage returns, and values are separated by tabs, this routine is provided to convert these characters to field marks (char 254) and value marks (char 253). This subroutine gets the value or contents of a global or local variable. The variable can contain ASCII characters less than 32 and greater than 126, including tabs and carriage returns. If your variables contain only ASCII characters between 32 and 126, use the faster WIN.GETVAR sub-routine.

## Parameters

The following table describes the parameters of the WIN.GETLIST command:

Parameter	Description
<i>VARNAME</i>	The name of the variable to return.
<i>VALUE</i>	The value of the variable specified in <i>VARNAME</i> .

## Example

Get the value of the PC item PCREC, convert to the host item format

---

```
CALL WIN.GETLIST( "PCREC" , HOSTREC )  
WRITE HOSTREC ON FILE, ID
```

---

## Related Subroutines

[WIN.GETDATA](#), [WIN.GETVAR](#), [WIN.SETDATA](#), [WIN.SETLIST](#), [WIN.SETVAR](#),  
[WIN.DBGET](#), [WIN.DBSET](#)

# WIN.GETPARAM

## Syntax

**WIN.GETPARAM** ( *NAME*, *VALUE* )

## Description

This function returns a wIntegrate menu option variable.

For a list of variables see the "Reference - Menu Options" section of the "Client Scripting Reference" manual.

## Parameters

The following table describes the parameters of the WIN.GETPARAM command:

Parameter	Description
<i>NAME</i>	The name of the parameter or variable to return.
<i>VALUE</i>	The value of the parameter specified in NAME.

## Example

This example looks for and returns the wIntegrate Title name.

```
VALUE = ""  
CALL WIN.GETPARAM("Title", VALUE)  
PRINT VALUE
```

## Related Subroutines

[WIN.SETPARAM](#)

## Related Script Commands

Get

# WIN.GETVAL

## Syntax

WIN.GETVAL ( *VAR* )

## Description

This subroutine retrieves a value sent to the host by the Host Send script command. A Host Send command must be run before this subroutine is called

## Parameters

The following table describes the parameters of the WIN.GETVAL command:

Parameter	Description
<i>VAR</i>	name of the variable to put the retrieved value in

## Example

Get the current application directory and version of wIntegrate

---

```
CALL WIN.COMSUB("Host Send AppDir;Host Send Version")
CALL WIN.GETVAL(WINTDIR)
CALL WIN.GETVAL(WINTVER)
```

---

## Version

4.0.1 Original

# WIN.GETVAR

## Syntax

**WIN.GETVAR** ( *VARNAME*, *VALUE* )

## Description

This routine returns the contents of a global variable. It is like WIN.GETDATA, but it can be used only if the variable has a simple value. This makes the transfer faster. No characters with ASCII values less than 32 or more than 126. This means there can be no carriage returns, tabs, or eight-bit characters. This routine is similar to WIN.GETDATA, but runs faster and sends less data along communication lines.

## Parameters

The following table describes the parameters of the WIN.GETVAR command:

Parameter	Description
<i>VARNAME</i>	The name of the variable to return.
<i>VALUE</i>	The value of the variable specified in <i>VARNAME</i> .

## Example

Find and print the current value of UserName

---

```
CALL WIN.GETVAR("UserName", VALUE)
PRINT "The current UserName is ": VALUE
```

---

## Related Subroutines

[WIN.GETDATA](#), [WIN.SETDATA](#), [WIN.SETVAR](#), [WIN.DBGET](#), [WIN.DBSET](#),  
[WIN.SETLIST](#)

# WIN.HELP

## Syntax

**WIN.HELP** *NAME/ID, HELPFILE*

## Description

This is a command line utility which looks up an entry in a Windows Help file.

This relies on command line parsing which is currently only implemented for UniVerse and UniData machine types.

## Parameters

The following table describes the parameters of the WIN.HELP command:

Parameter	Description
<i>NAME/ID</i>	The name of the topic to look up. If this numeric it is assumed to be a topic number
<i>HELPFILE</i>	The name of the windows help file. Set to "" for the wIntegrate help file

## Related Subroutines

[WIN.HELPID](#), [WIN.HELPNAME](#)

## Related Script Commands

Help TopicId, Help TopicName

## Version

4.1 Original version

# WIN.HELPID

## Syntax

**WIN.HELPID** ( *ID*, *HELPPFILE* )

## Description

This command shows the specified topic number in a Windows Help File.

## Parameters

The following table describes the parameters of the WIN.HELPID command:

Parameter	Description
<i>ID</i>	The numeric topic id
<i>HELPPFILE</i>	The name of the windows help file. Set to "" for the wIntegrate help file

## Related Subroutines

[WIN.HELPNAME](#)

## Related Script Commands

Help TopicId

## Version

4.1 Original version

# WIN.HELPPNAME

## Syntax

WIN.HELPPNAME ( *NAME*, *HELPPFILE* )

## Description

This command looks up the specified topic name in a Windows Help File.

## Parameters

The following table describes the parameters of the WIN.HELPPNAME command:

Parameter	Description
<i>NAME</i>	The name of the topic to look up
<i>HELPPFILE</i>	The name of the windows help file. Set to "" for the wIntegrate help file

## Related Subroutines

[WIN.HELPPID](#)

## Related Script Commands

Help TopicName

## Version

4.1 Original version



# WIN.HGLASS

## Syntax

**WIN.HGLASS** ( *MODE*, *FLAG* )

## Description

This subroutine switches on and off the hourglass cursor. Use this when running a process and you want to disable the cursor.

## Parameters

The following table describes the parameters of the WIN.HGLASS command:

Parameter	Description
<i>MODE</i>	Specifies local or global mode for the hourglass display. 0 = Local mode displays the hourglass only when the is over the wIntegrate window. 1 = Global mode displays the hourglass anywhere on screen.
<i>FLAG</i>	Switches the hourglass on and off. 0 = Hourglass off 1 = Hourglass on

## Example

This example turns on the hourglass in local mode while running a select statement

---

The WIN.HGLASS subroutine with both parameters as zero turns off the hourglass and enables the cursor.

\* Turn on the hourglass while running a select statement

```
CALL WIN.HGLASS(0,1)
```

```
EXECUTE "SSELECT COURSES"
```

```
CALL WIN.HGLASS(0,0)
```

---

# WIN.HOSTVER

## Syntax

WIN.HOSTVER

## Description

This subroutine allows you to check the Host programs Version and Machine Type. Enter WIN.HOSTVER at the UniData prompt.

## Parameters

None

## Example

Check the host version

---

```
:WIN.HOSTVER
Host Programs:
  Version:      4.2.1
  Machine Type: UV
  Parameter File: WIN.PROGS
```

---

# WIN.HOTSPOT

## Syntax

WIN.HOTSPOT ( *SPOTS* )

## Description

This subroutine creates hot spots on the screen. The hot spot executes a defined command in response to a mouse click. Hotspots overrides but does not change any definition in Setup Mouse. If two hotspots occupy the same area, the most recent will take precedence.

## Parameters

The following table describes the parameters of the WIN.HOTSPOT command:

Parameter	Description
<i>SPOTS</i>	The SPOTS parameter defines the position and command for the hot spot. If SPOTS is null, hotspots are removed.

## Fields for SPOTS

The hotspots are defined as one hotspot per field with the following values

Field	Description
<i>1.1</i>	The hot spot top left x-coordinate.
<i>1.2</i>	The hot spot top left y-coordinate.
<i>1.3</i>	The hot spot bottom right x-coordinate.
<i>1.4</i>	The hot spot bottom right y-coordinate.
<i>1.5</i>	The hot spot command. The default is to send down the field number of the hot spot.

## Example

The following example is taken from the WIN.HSDEMO demonstration program.

---

```
*
AM=CHAR(254)
VM=CHAR(253)
R.SPOTS=''
R.COLS=''
* Set up array of areas for hot spots
R.SPOTS<-1>=0:VM:13:VM:79:VM:20;R.COLS<-1>="Blue"
R.SPOTS<-1>=15:VM:15:VM:39:VM:16;R.COLS<-1>="LightRed"
R.SPOTS<-1>=40:VM:15:VM:65:VM:16;R.COLS<-1>="Yellow"
R.SPOTS<-1>=15:VM:17:VM:39:VM:18;R.COLS<-1>="LightCyan"
R.SPOTS<-1>=40:VM:17:VM:65:VM:18;R.COLS<-1>="LightGreen"
*
* Cover the remaining area so we can tell people whats going on
R.SPOTS<-1>=0:VM:0:VM:79:VM:12:VM:"DESC"
R.SPOTS<-1>=0:VM:21:VM:79:VM:23:VM:"PROMPT"
R.SPOTS<-1>=58:VM:23:VM:78:VM:23:VM:"*"
.
.
.
* Define hotspots
CALL WIN.HOTSPOT(R.SPOTS)
C22=@(0,22):@(-4)
```

---

# WIN.HOTSPOT2

## Syntax

WIN.HOTSPOT2 ( *SPOTS* )

## Description

This subroutine creates hot spots on the screen. The hot spot executes a defined command in response to a mouse click. Hotspots overrides but does not change any definition in Setup Mouse.

If two hotspots occupy the same area, the most recent will take precedence.

Use this instead of the WIN.HOTSPOT routine if you want to specify a double click response on the hotspot or to set individual styles for each hotspot.

## Parameters

The following table describes the parameters of the WIN.HOTSPOT2 command:

Parameter	Description
<i>SPOTS</i>	The SPOTS parameter is a dynamic array of all the hotspots to define. Each sub-value in each field defines the position, command and styles for the hot spot. If SPOTS is null, all hotspots are removed.

## Fields for SPOTS

The following specifies the multivalues to set for each hot spot

Field	Description
<i>1</i>	The left column of the hotspot
<i>2</i>	The top row of the hotspot
<i>3</i>	The right column of the hotspot
<i>4</i>	The bottom row of the hotspot

Field	Description
5	The value to return to the host on a single click. Use "" for the position of the hotspot in the SPOTS dynamic array. When sent to the host this value is appended with a <CR>.
6	The value to return from the host on a double click. No value is sent if this field is not set
7	The style of the hotspot: 0 - Hotspot is transparent. The default is 0. 1 - Hotspot is raised with a grey/white border. 2 - Hotspot is sunk, with a grey/white border. 3 - Hotspot looks like a text box with a 2 pixel grey/white and black border. 64 - overrides the screen color. Use it by itself or add it to other flag numbers.
8	Foreground color (see WIN.COLOR for how to specify this). Default "Black"
9	Background color (see WIN.COLOR for how to specify this). Default "White"

## Version

### 4.1 Original

# WIN.HSCRIPT

## Syntax

WIN.HSCRIPT ( *SCRIPT* )

## Description

This subroutine allows you to define a script, then transfers it to the PC as a single script. Use this subroutine when you want to run PC scripts from the host. This subroutine assumes the scripts contain only ASCII characters 32 to 126. If you include ASCII characters outside of these parameters, use the WIN.HSCRIPTC subroutine.

## Parameters

The following table describes the parameters of the WIN.HSCRIPT command:

Parameter	Description
<i>SCRIPT</i>	The script or scripts to transfer to the PC.

## Example

This example sends a script to the PC to show a message

It brings up a message box depending whether the time is before or after noon.

\* Create a message box dependent on time of day

```
SCRIPT = ''
```

```
SCRIPT<-1> = 'T = Oconv(Time()),"MT")'
```

```
SCRIPT<-1> = 'If Field(T,":",1) > 12 Then'
```

```
SCRIPT<-1> = '  MessageBox "Good Afternoon"'
```

```
SCRIPT<-1> = 'Else'
```

```
SCRIPT<-1> = '  MessageBox "Good Morning"'
```

```
SCRIPT<-1> = 'EndIf'
```

```
CALL WIN.HSCRIPT(SCRIPT)
```

```
END
```

## Related Subroutines

[WIN.COMSUB](#), [WIN.HSCRIPTC](#), [WIN.FSCRIPT](#)

# WIN.HSCRIPTC

## Syntax

WIN.HSCRIPTC ( *SCRIPT* )

## Description

This subroutine allows you to define a script, then transfers the script to the PC as a single script. Use this subroutine when you want to use PC scripts on the host. This subroutine allows any ASCII character to be in the *SCRIPT*, and for this reason runs more slowly than WIN.HSCRIPT.

## Parameters

The following table describes the parameters of the WIN.HSCRIPTC command:

Parameter	Description
<i>SCRIPT</i>	The script or scripts to transfer to the PC.

## Example

This example sends a script to the PC and brings up a message box

It brings up a message box depending whether the time is before or after noon.

\* Create a message box dependent on time of day

```
SCRIPT = ''
```

```
SCRIPT<-1> = 'T = Oconv(Time(),"MT")'
```

```
SCRIPT<-1> = 'If Field(T,":",1) > 12 Then'
```

```
SCRIPT<-1> = '  MessageBox "Good Afternoon"'
```

```
SCRIPT<-1> = 'Else'
```

```
SCRIPT<-1> = '  MessageBox "Good Morning"'
```

```
SCRIPT<-1> = 'EndIf'
```

```
CALL WIN.HSCRIPTC(SCRIPT)
```

```
END
```

## Related Subroutines

[WIN.COMSUB](#), [WIN.HSCRIPT](#), [WIN.FSCRIPT](#)



# WIN.ILADD

## Syntax

**WIN.ILADD** ( *NAME*, *FILENAME*, *MASK*, *FLAGS*, *RESP* )

## Description

This command add the image in the specified bitmap file to the named image list.

## Parameters

The following table describes the parameters of the WIN.ILADD command:

Parameter	Description
<i>NAME</i>	Image List name
<i>FILENAME</i>	Filename of bitmap file to add
<i>MASK</i>	Color to use as mask (use "" for no mask.)
<i>FLAGS</i>	Can be one of the following:0x0001 - Load image as a monochrome image.0x0020 - Map first pixel to Window background.0x0040 - Use system default size.0x1000 - Map button edge colors.
<i>RESP</i>	Index number where the image was added to the image list or -1 if unsuccessful

## Version

4.0.3 Original version

# WIN.ILCOUNT

## Syntax

WIN.ILCOUNT ( *NAME*, *RESP* )

## Description

This command returns the number of images within the named image list.

## Parameters

The following table describes the parameters of the WIN.ILCOUNT command:

Parameter	Description
<i>NAME</i>	Image List name
<i>RESP</i>	Returned number of images in the image list

## Version

4.0.3 Original version

# WIN.ILDELETE

## Syntax

WIN.ILDELETE ( *NAME* )

## Description

This routine deletes the image list.

## Parameters

The following table describes the parameters of the WIN.ILDELETE command:

Parameter	Description
<i>NAME</i>	Image List name

## Version

4.0.3 Original version

# WIN.ILICON

## Syntax

WIN.ILICON ( *NAME*, *FILENAME*, *RESP* )

## Description

This command add the image in the specified icon file to the named image list.

## Parameters

The following table describes the parameters of the WIN.ILICON command:

Parameter	Description
<i>NAME</i>	Image List name
<i>FILENAME</i>	Filename of icon file to add
<i>RESP</i>	Index number where the image was added to the image list or -1 if unsuccessful

## Version

4.0.3 Original version

# WIN.ILINFO

## Syntax

**WIN.ILINFO** ( *NAME*, *IMAGE.NO*, *PLANES*, *BPP* )

## Description

This routine returns information on the pixel layout of an image within the image list.

## Parameters

The following table describes the parameters of the WIN.ILINFO command:

Parameter	Description
<i>NAME</i>	Image List name
<i>IMAGE.NO</i>	The index of the image to return the values of
<i>PLANES</i>	Returned number of planes in the image
<i>BPP</i>	Returned number of bits per pixel in the image

## Version

4.0.3 Original version

# WIN.ILISIZE

## Syntax

WIN.ILISIZE ( *NAME*, *WIDTH*, *DEPTH* )

## Description

This routine returns the width and depth of the images stored in the specified image list.

## Parameters

The following table describes the parameters of the WIN.ILISIZE command:

Parameter	Description
<i>NAME</i>	Image List name
<i>WIDTH</i>	Returned width of each image
<i>DEPTH</i>	Returned depth of each image

## Version

4.0.3 Original version

# WIN.ILLOAD

## Syntax

**WIN.ILLOAD** ( *NAME*, *FILENAME*, *OPTS*, *RESP* )

## Description

This routine loads an image list from a file. The file can contain multiple images.

## Parameters

The following table describes the parameters of the WIN.ILLOAD command:

Parameter	Description
<i>NAME</i>	Image List name
<i>FILENAME</i>	Filename of bitmap file to add
<i>OPTS</i>	Options for this load. See Table below
<i>RESP</i>	Set to 1 if load was sucessful, otherwise 0

## Fields for OPTS

Field	Description
<i>1</i>	Width of each image in the list (default 0)
<i>2</i>	Maximum number of images the list can contain (default is number that are loaded)
<i>3</i>	Color for transparency mask. Default is no mask
<i>4</i>	Type of image file: 0 - bitmap, 1-icon, 2-cursor. Default is 0
<i>5</i>	Flags: Can be one of the following: 0x0001- Load image as a monochrome image. 0x0020 - Map first pixel to Window background. 0x0040 - Use system default size. 0x1000 - Map button edge colors.

## Version

4.0.3 Original version



# WIN.ILNEW

## Syntax

WIN.ILNEW ( *NAME*, *WIDTH*, *DEPTH*, *OPTS*, *RESP* )

## Description

This routine loads an image list from a file. The file can contain multiple images.

## Parameters

The following table describes the parameters of the WIN.ILNEW command:

Parameter	Description
<i>NAME</i>	Image List name
<i>WIDTH</i>	Width of each image in the list
<i>DEPTH</i>	Depth of the image list
<i>OPTS</i>	Options for this load. See Table below
<i>RESP</i>	Set to 1 if load was sucessful, otherwise 0

## Fields for OPTS

Field	Description
<i>1</i>	Color depth of the image mask (default 0, 4 for new drivers DDB for old) 4,8,16,24,32 (or 254 for Device dependent bitmap).
<i>2</i>	Color for transparency mask. Default is no mask
<i>3</i>	Initial size of image list. Default 4
<i>4</i>	Maximum size of image list. Default 0

## Version

4.0.3 Original version

# WIN.ILREMOVE

## Syntax

WIN.ILREMOVE ( *NAME*, *INDEX* )

## Description

This routine remove an image from an image list.

## Parameters

The following table describes the parameters of the WIN.ILREMOVE command:

Parameter	Description
<i>NAME</i>	Image List name
<i>INDEX</i>	Index of the image to remove

## Version

4.0.3 Original version

# WIN.IMAGE

## Syntax

**WIN.IMAGE** ( *FILENAME*, *LEFT*, *TOP*, *RIGHT*, *BOTTOM*, *CLIP* )

## Description

This subroutine displays an image on the wIntegrate screen. You must specify the full path of the file name.

## Parameters

The following table describes the parameters of the WIN.IMAGE command:

Parameter	Description
<i>FILENAME</i>	The name of the PC file where the image is stored.
<i>LEFT</i>	Position of the left side of the image.
<i>TOP</i>	Position of the top of the image.
<i>RIGHT</i>	Position of the right side of the image.
<i>BOTTOM</i>	Position of the bottom of the image.
<i>CLIP</i>	Sets whether to clip or scale the image. 0 - scales the image 1 - clips the image.

## Example

This example is taken from the WIN.IMDEMO demonstration program.

---

```
*
CALL WIN.IMAGE( "IMAGE\COMPUTER.WMF", 40,5,70,17, 0)
*
PRINT @(0,18): "The image above was displayed by :-"
PRINT ' CALL WIN.IMAGE( "IMAGE\COMPUTER.WMF", 40,4,70,18,0) '
END
```

---

## Related Subroutines

[WIN.IMOPEN](#), [WIN.IMCLOSE](#), [WIN.IMCHANGE](#)

## **Related Script Commands**

Draw Image, FileExist

# WIN.IMCHANGE

## Syntax

WIN.IMCHANGE ( *IMAGE.NO*, *FILENAME* )

## Description

This subroutine changes an image that was opened by WIN.IMOPEN. Specify the full path of the file unless the file is stored in the wintegrate directory. You must use WIN.IMCLOSE to close the PC image file.

## Parameters

The following table describes the parameters of the WIN.IMCHANGE command:

Parameter	Description
<i>IMAGE.NO</i>	The image number that you specified in WIN.IMOPEN.
<i>FILENAME</i>	The full path of the PC image file.

## Example

This subroutine is a simple demonstration of the image window subroutines.

---

```
* This demonstrates how to open, change, and close image windows
PRINT "To display an image in a window, use WIN.IMOPEN"
CALL WIN.IMOPEN(1,"image\normal.bmp",0,0,8,8)
PRINT "Close the window, then press Enter to continue"
INPUT DUM:
* Change the image
PRINT "Then change the image with WIN.IMCHANGE"
CALL WIN.IMCHANGE(1,"image\pressed.bmp")
* Close the image file
PRINT "Close the window, then press Enter to continue"
INPUT DUM:
PRINT "Then close the image file with WIN.IMCLOSE"
CALL WIN.IMCLOSE(1)
END
```

---

## **Related Subroutines**

[WIN.IMOPEN](#), [WIN.IMCLOSE](#), [WIN.IMAGE](#)

## **Related Script Commands**

DialogBox Graphic

# WIN.IMCLOSE

## Syntax

WIN.IMCLOSE ( *IMAGE.NO* )

## Description

This subroutine closes a PC image file that was opened with the WIN.IMOPEN subroutine.

## Parameters

The following table describes the parameters of the WIN.IMCLOSE command:

Parameter	Description
<i>IMAGE.NO</i>	The number of the image specified in WIN.IMOPEN.

## Example

This subroutine is a simple demonstration of the image window subroutines.

```
* This demonstrates how to open, change, and close image windows
PRINT "To display an image in a window, use WIN.IMOPEN"
CALL WIN.IMOPEN(1,"image\normal.bmp",0,0,8,8)
PRINT "Close the window, then press Enter to continue"
INPUT DUM:
* Change the image
PRINT "Then change the image with WIN.IMCHANGE"
CALL WIN.IMCHANGE(1,"image\pressed.bmp")
* Close the image file
PRINT "Close the window, then press Enter to continue"
INPUT DUM:
PRINT "Then close the image file with WIN.IMCLOSE"
CALL WIN.IMCLOSE(1)
END
```

## Related Subroutines

[WIN.IMOPEN](#), [WIN.IMCHANGE](#), [WIN.IMAGE](#)



## **Related Script Commands**

DialogBox Graphic

# WIN.IMOPEN

## Syntax

**WIN.IMOPEN** ( *IMAGE.NO*, *FILENAME*, *LEFT*, *TOP*, *RIGHT*, *BOTTOM* )

## Description

This subroutine opens a window to contain a PC image file on the wIntegrate screen. Specify the full path of the file unless the file is stored in the wIntegrate directory. You must use WIN.IMCLOSE to close the PC image file. The window size and position are relative to the size of the Windows system font being used.

## Parameters

The following table describes the parameters of the WIN.IMOPEN command:

Parameter	Description
<i>IMAGE.NO</i>	A number that you specify for this image.
<i>FILENAME</i>	The name of the PC file where the image is stored.
<i>LEFT</i>	Position of the left side of the image.
<i>TOP</i>	Position of the top of the image.
<i>RIGHT</i>	Position of the right side of the image.
<i>BOTTOM</i>	Position of the bottom of the image.

## Example

This subroutine is a simple demonstration of the image window subroutines.

```
* This demonstrates how to open, change, and close image windows
PRINT "To display an image in a window, use WIN.IMOPEN"
CALL WIN.IMOPEN(1,"image\normal.bmp",0,0,8,8)
PRINT "Close the window, then press Enter to continue"
INPUT DUM:
* Change the image
PRINT "Then change the image with WIN.IMCHANGE"
CALL WIN.IMCHANGE(1,"image\pressed.bmp")
* Close the image file
PRINT "Close the window, then press Enter to continue"
INPUT DUM:
```

```
PRINT "Then close the image file with WIN.IMCLOSE"  
CALL WIN.IMCLOSE(1)  
END
```

---

## **Related Subroutines**

[WIN.IMCHANGE](#), [WIN.IMCLOSE](#), [WIN.IMAGE](#)

## **Related Script Commands**

DialogBox Graphic

# WIN.IMPORT

## Syntax

**WIN.IMPORT** ( *PCFILE*, *FILE*, *ITEMS*, *FIELDS*, *OPTS*, *STATUS* )

## Description

This subroutine imports data from the host computer to a PC file. It calls the RunImportFile menu command, so to understand this subroutine, you can look at the Run Import File menu option.

## Parameters

The following table describes the parameters of the WIN.IMPORT command:

Parameter	Description
<i>PCFILE</i>	The full path name of the PC file where the data will be saved.
<i>FILE</i>	The name of the host file to export.
<i>ITEMS</i>	The items or records in the host file to export.
<i>FIELDS</i>	The fields or attributes of the records to export.
<i>OPTS</i>	Additional modifications for other file transfer options
<i>STATUS</i>	The returned status. This returns "OK", "Cancel" or "Error".

## Fields for OPTS

Field	Description
<i>I</i>	Format - A file format for the DOS file. Valid formats are ASC, CRD, CSV, DBF, FIX, MRG, RAW, WKS, WK1, and XLS. The default is the extension in the FILE parameter.

Field	Description
2	Overwrite - Specifies options of overwriting an existing file. This can be "Yes", "No", or "Append". The default is "Yes".
3	Mode - Mode for import "Normal", "Capture" or "Reformat"
4	Suppress Id 0 Suppress the item id from the import. 0 don't suppress id, 1 suppress id.
5	Field descriptions - Import field descriptions as first record of import
6	Translate - This option is available to change characters as they are exported, according to the Translation. This can be "None", "Ascii" or "All". The default is "Ascii".
7	Translation - The default is "255,\r\n\f\r\n,\254,\r\n". If you want to use a format other than ASCII, you must go into the Export dialog box and delete the Translation definition.
8	AutoExit - Exits the Export monitor at the end of the transfer. The default is "True".
9	Inform - Notify the user when export is finished. The default is "False".
10	Timeout - The maximum number of seconds to wait for a response from the host. The default is "5".
11	Retries - The maximum number of retries. The default is "3".
13	Numeric conversion - Multi-valued field with the following values:
13.1	- Number conversion on 1 (or 0 to turn it off)
13.2	- Separator character
13.3	- Currency symbol
13.4	- Decimal symbol

Field	Description
<i>14</i>	Multi-Value options - Multi-valued field with the followings values:
<i>14.1</i>	- Explode multi-values, 1 for on, 0 for off
<i>14.2</i>	- Repeat multi-values, 1 for on, 0 for off
<i>15</i>	Dictionary options - Multi-valued field with the followings values:
<i>15.1</i>	- Use formatting information
<i>15.2</i>	- Left-justified fields are text
<i>15.3</i>	- Right-justified fields are numeric

## Example

This example imports the item IMPORT.TEXT from WIN.PROGS to the PC file C:\imptext.txt.

---

```
CALL WIN.IMPORT("c:\imptext.txt", "WIN.PROGS", "IMPORT.TEXT", "", "",  
STATUS)
```

---

The next example imports a spreadsheet to Excel with no item ID's and notifies the user when done

---

```
OPTS = ""  
OPTS<3> = "Capture"  
OPTS<4> = " True"  
OPTS<9> = "True"  
CALL WIN.IMPORT("c:\excel\orders.xls", "ORDER", "SSELECT ORDER BY  
CUST.NAME",  
"CUST.NAME QUANTITY VALUE", OPTS, STATUS)
```

---

## Related Subroutines

[WIN.EXPORT](#), [WIN.SPOOL](#)

## Related Script Commands

Dialog RunImportFile

## **Version**

4.1 New options OPTS<15>

# WIN.INFOBOX

## Syntax

**WIN.INFOBOX** ( *TEXT*, *FLAGS* )

## Description

This subroutine displays text in an information box. This box always has a header of "Host Information". If the info box is displayed and you call WIN.INFOBOX again and change the text, the existing box displays the new text. To close the info box, call WIN.INFOBOX and list the text as null.

## Parameters

The following table describes the parameters of the WIN.INFOBOX command:

Parameter	Description
<i>TEXT</i>	The text to display in the info box.
<i>FLAGS</i>	The position of the box on the screen. T = top B = bottom C = center

## Example

\* Display an info box while running report

---

```
* Display an info box while running report
CALL WIN.INFOBOX("Selecting data for the Student report","C")
EXECUTE "LIST STUDENT FNAME MAJOR BY MAJOR"
CALL WIN.INFOBOX("","")
RETURN
END
```

---



# WIN.INPBOX

## Syntax

**WIN.INPBOX** ( *TEXT*, *TITLE*, *LEN*, *RESP* )

## Description

This subroutine displays a dialog box and waits for a response.

## Parameters

The following table describes the parameters of the WIN.INPBOX command:

Parameter	Description
<i>TEXT</i>	The text that appears in the dialog box.
<i>TITLE</i>	The caption for the dialog box.
<i>LEN</i>	The length of the input box.
<i>RESP</i>	The response to wait for.

## Example

This example program prompts the user for a response:

---

```
This example program prompts the user for a response:
* Display an input box for the login name
RESP=" "
CALL WIN.INPBOX("Enter your login name","Login",10,RESP)
PRINT "Hello ":RESP, "welcome to UniData"
END
```

---

# WIN.INVOKE

## Syntax

**WIN.INVOKE** ( *MENU.DLG* )

## Description

This subroutine invokes a wIntegrate menu command.

For a full list of menu commands see the "Reference - Menu Options" section of the "Client Scripting Reference".

## Parameters

The following table describes the parameters of the WIN.INVOKE command:

Parameter	Description
<i>MENU.DLG</i>	The menu command to run. You may also use abbreviations for the commands.

## Example

The following example copies data to the clipboard using Edit copy special

---

```
CALL WIN.INVOKE( "EditCopySpecial" )
```

---

## Related Subroutines

[WIN.SHOW](#)

## Related Script Commands

Invoke

# WIN.LI

## Syntax

**WIN.LI** ( *VALUE*, *LEN*, *ESC* )

## Description

This subroutine calls the Edit Input routine. It edits an area of the screen from the current cursor position to the position specified in *LEN*, and returns the result. You can use this subroutine to edit host data, as you can using the WIN.EI subroutine, but this routine requires fewer parameters.

## Parameters

The following table describes the parameters of the WIN.LI command:

Parameter	Description
<i>VALUE</i>	The text returned.
<i>LEN</i>	The length of the edit area.
<i>ESC</i>	Returns 0 if Esc is not pressed, 1 if Esc is pressed.

## Related Subroutines

[WIN.EI](#)

## Related Script Commands

EditInput, Cursor

# WIN.LICINFO

## Syntax

WIN.LICINFO ( *INFO* )

## Description

This routine returns the license information for this copy of wIntegrate

## Parameters

The following table describes the parameters of the WIN.LICINFO command:

Parameter	Description
<i>INFO</i>	Variable to set with a dynamic array of the license information

## Fields for INFO

Fields returned in INFO

Field	Description
<i>1</i>	Registered to name
<i>2</i>	Organization
<i>3</i>	Serial Number
<i>4</i>	Maximum number of users

## Version

4.2.1 Original

# WIN.LOOKUP

## Syntax

**WIN.LOOKUP** ( *R.TABLE*, *R.LIST*, *REF* )

## Description

The WIN.LOOKUP subroutine displays a popup dialog box from which the user can make a selection. The R.TABLE parameter details how the dialog box looks, and R.LIST is the list of options to choose from.

## Parameters

The following table describes the parameters of the WIN.LOOKUP command:

Parameter	Description
<i>R.TABLE</i>	Specifies how the dialog box looks, such as caption, column and width, etc. See the following table for R.TABLE options.
<i>R.LIST</i>	This is the list of options listed in the dialog.
<i>REF</i>	The REF returned is the first multivalue of the line or field from R.LIST.

## Fields for R.TABLE

Field	Description
<i>1</i>	The caption for the dialog box.
<i>2</i>	Column headings within the dialog box.
<i>3</i>	Column sizes in the dialog box.
<i>4</i>	Number of Lines to show in the list box.
<i>5</i>	The start position to display the dialog or null (“”).
<i>6</i>	Parent dialog for lookup box.
<i>7</i>	Displays the hourglass cursor while building the lookup dialog box.

Field	Description
8	Map character depending on emulation if set to 1.

## Example

This example is part of the WIN.LUDEMO demonstration program.

---

This example is part of the WIN.LUDEMO demonstration program.

```
* Set up lookup details
```

```
R.TABLE = "Choose a fruit" ;* Lookup title
```

```
R.TABLE<2,1> = "Code"; R.TABLE<3,1> = 5 ;* Column titles and sizes
```

```
R.TABLE<2,2> = "Fruit"; R.TABLE<3,2> = 12
```

```
R.TABLE<4> = 5 ;* Number of lines to show
```

```
R.TABLE<5,1> = 50; R.TABLE<5,2> = 8 ;* Start position of dialog
```

```
*
```

```
* Set up data
```

```
R.LIST = ''
```

```
R.LIST<1,1> = 'AP'; R.LIST<1,2> = 'Apple'
```

```
R.LIST<2,1> = 'BN'; R.LIST<2,2> = 'Banana'
```

```
R.LIST<3,1> = 'KW'; R.LIST<3,2> = 'Kiwi'
```

```
R.LIST<4,1> = 'OR'; R.LIST<4,2> = 'Orange'
```

```
R.LIST<5,1> = 'PR'; R.LIST<5,2> = 'Pear'
```

```
*
```

```
REF = ''
```

```
CALL WIN.LOOKUP(R.TABLE, R.LIST, REF)
```

---

## Related Script Commands

DialogBox commands

# WIN.MENUATT

## Syntax

**WIN.MENUATT** ( *MENU*, *SUBMENU*, *TITLE*, *POS* )

## Description

This subroutine attaches a submenu to an existing menu. Use this when you need to attach a menu to a menu bar as a submenu. You may want to create submenus at the same time you create the menu bar because the menu tree is created with WIN.MENULOAD. Any attached menu is activated by the WIN.MENUIN subroutine.

## Parameters

The following table describes the parameters of the WIN.MENUATT command:

Parameter	Description
<i>MENU</i>	Specifies the menu that the submenu will to. Default is the "MainMenu".
<i>SUBMENU</i>	The name of the submenu to attach.
<i>TITLE</i>	The text on the main menu that the submenu to.
<i>POS</i>	The position on the main menu where the will be attached.

## Example

The following example is part of the WIN.MENUDEMO demonstration program.

```
* Attaching/Detaching in action
1900 IF MENU.SHOW = 1 THEN
CALL WIN.MENUATT( ' ', "Demo", "&Demo", "Help" )
CALL WIN.MSTATE( "Demo", 9, 1, 1 )
MENU.SHOW = 2
PRINT C23: "Select menu option from the Demo menu on the main menu
bar":
END ELSE
CALL WIN.MENUDET( ' ', "Demo" )
CALL WIN.MSTATE( "Demo", 9, 0, 1 )
MENU.SHOW= 1
```

```
PRINT C23:"Select the menu option from the Popup menu":  
END  
RETURN
```

---

## **Related Subroutines**

[WIN.MENUDEL](#), [WIN.MENUDET](#), [WIN.MENUIN](#), [WIN.MENULOAD](#)

## **Related Script Commands**

Menu Attach, Menu Enable



# WIN.MENUDEL

## Syntax

**WIN.MENUDEL** ( *NAME* )

## Description

This subroutine deletes a menu from the computer's memory. Use this when you no longer need to show a menu. If you want to delete a submenu and not the main menu, you must use WIN.MENUDET to detach it.

For the demonstration program WIN.MENUDEMO, a new submenu is attached to the main wIntegrate menu, then deleted using WIN.MENUDEL.

## Parameters

The following table describes the parameters of the WIN.MENUDEL command:

Parameter	Description
<i>NAME</i>	The name of the menu to delete

## Example

The following example is part of the WIN.MENUDEMO demonstration program.

```
*
IF MENU.SHOW = 2 THEN CALL WIN.MENUDET( ' ', "Demo" )
CALL WIN.MENUDEL( "Demo" )
CALL WIN.MENUDEL( "DemoSub" )
*
PRINT C23:
PRINT C22:"End of Demonstration, Press <CR> to continue":
INPUT DUM:
*
STOP
```

---

## Related Subroutines

[WIN.MENUATT](#), [WIN.MENUDET](#), [WIN.MENUIN](#), [WIN.MENULOAD](#)

## **Related Script Commands**

Menu Delete

# WIN.MENUDET

## Syntax

**WIN.MENUDET** ( *MENU*, *SUBMENU* )

## Description

This subroutine detaches a submenu from a menu.

## Parameters

The following table describes the parameters of the WIN.MENUDET command:

Parameter	Description
<i>MENU</i>	The name of the menu from which to detach submenu. The default is MainMenu.
<i>SUBMENU</i>	The name of the submenu to detach.

## Example

The following example is part of the WIN.MENUDEMO demonstration program.

---

```
* Attaching/Detaching in action
1900 IF MENU.SHOW = 1 THEN
    CALL WIN.MENUATT( ' ', "Demo", "&Demo", "Help" )
    CALL WIN.MSTATE( "Demo", 9, 1, 1 )
    MENU.SHOW = 2
    PRINT C23: "Select menu option from the Demo menu on the main menu
bar":
END ELSE
    CALL WIN.MENUDET( ' ', "Demo" )
    CALL WIN.MSTATE( "Demo", 9, 0, 1 )
    MENU.SHOW = 1
    PRINT C23: "Select the menu option from the Popup menu":
END
RETURN
*
```

---

## **Related Subroutines**

[WIN.MENUATT](#), [WIN.MENUDEL](#), [WIN.MENUIN](#), [WIN.MENULOAD](#)

## **Related Script Commands**

Menu Detach, IsOnMenu

# WIN.MENUIN

## Syntax

**WIN.MENUIN** ( *MENU*, *SUBMENU*, *RMENU*, *OPT* )

## Description

This subroutine enables a submenu, then waits for input. When it receives input, it is disabled.

## Parameters

The following table describes the parameters of the WIN.MENUIN command:

Parameter	Description
<i>MENU</i>	The name of the main menu the submenu is attached Default is MainMenu.
<i>SUBMENU</i>	The name of the submenu to enable.
<i>RMENU</i>	Returns the name the menu was given when it was created.
<i>OPT</i>	Returns the option number of the menu item, counting the top.

## Example

The following example is part of the WIN.MENUDEMO demonstration program.

```
*
IF MENU.SHOW = 1 THEN
  CALL WIN.POPUPIN( "Demo" , 45 , 10 , MENU , OPT )
END ELSE
  CALL WIN.MENUIN( "" , "Demo" , MENU , OPT )
END
*
PRINT @(0,18):@(-4):"Response, Menu = ':MENU:'", Option = ':OPT:':
* BEGIN CASE
CASE MENU = ''
  IF OPT = 'ESC' THEN
    PRINT CMSG:'This means no option has been selected from the
menu':
  END ELSE
```

```
        PRINT CMSG:'This is keyboard input of ':OPT:
CASE MENU = "Demo"
        PRINT CMSG:"This means the top level menu ":MENU:", had option
":OPT:' selected':
```

---

## Related Subroutines

[WIN.MENUATT](#), [WIN.MENUDEL](#), [WIN.MENUDET](#), [WIN.MENULOAD](#)

# WIN.MENULOAD

## Syntax

**WIN.MENULOAD** ( *NAME*, *DEF* )

## Description

This subroutine loads a menu to the PC before it is attached or shown on the main menu bar. DEF is a record that contains one field for each menu option. Each field consist of up to three multivalues (DEF).

## Parameters

The following table describes the parameters of the WIN.MENULOAD command:

Parameter	Description
<i>NAME</i>	The name of the menu to load.
<i>DEF</i>	The menu definition. See the DEF table below.

## Fields for DEF

Field	Description
1	This is the text for the menu. Use dash (-) — horizontal separator. Use greater-than sign to lead to a submenu defined in DEF Value 2. (See the example in this section.)
2	Text to send to the host when a menu option is selected. This refers to a submenu if the text starts with a ">". A null value (""), defines it for use with WIN.MENUIN and WIN.POPUPIN.
3	This is text to display as "help" in the status bar when an option is highlighted. This text can be only one line.

## Example

The following example is part of the WIN.MENUDEMO demonstration program.

---

```
* First define submenu
SUBDEF = ' '
SUBDEF<1>="&Creating"
SUBDEF<2>="&Attaching"
SUBDEF<3>="&Displaying"
CALL WIN.MENULOAD( "DemoSub" ,SUBDEF)
*
* Define Main menu
DEF = ' '
DEF<-1>='&Format of DEF':VM:VM:"Description of DEF argument"
DEF<-1>='&DEF used for this menu':VM:VM:"Definition used for this
menu"
DEF<-1>='&Popup arguments':VM:VM:"Arguments to WIN.POPUPIN"
DEF<-1>='>&Submenus':VM: "DemoSub"
DEF<-1>='Se&parators'
DEF<-1>='Setting menu &item state'
DEF<-1>='-'
DEF<-1>='Attaching to &MainMenu':VM:VM:"Details on attaching to main
menu"
DEF<-1>='&Attached to Main':VM:VM:"Attach/Unattach from main menu"
DEF<-1>='-'
DEF<-1>='E&xit Demo':VM:VM:"FIN the menu demonstration"
NO.OPTS = DCOUNT(DEF,AM)
*
PRINT C22:"Press <CR> to continue with demonstration":
INPUT DUM:
PRINT C22:
*
CALL WIN.MENULOAD( "Demo" ,DEF)
```

---

## Related Subroutines

[WIN.MENUATT](#), [WIN.MENUDEL](#), [WIN.MENUDET](#), [WIN.MENUIN](#),  
[WIN.MSTATE](#)



# WIN.MLADDR

## Syntax

**WIN.MLADDR** ( *MAIL.TO*, *MAIL.CC*, *MAIL.BCC*, *OPTS*, *DELIM* )

## Description

This routine displays a mail system dialog box allowing the user to address a message by selecting address book addresses.

If any of the variables *MAIL.TO*, *MAIL.CC* or *MAIL.BCC* contain names when this command is called, those names will appear pre-selected in the dialog box. Specified recipient names can be full names, partially unresolved names or full email addresses, eg. "Mary Doe", "Mary", "marydoe@widgets.com", respectively. Specified multiple recipients can be separated with a semi-colon, field mark (CHAR(254)) or value mark (CHAR(253)).

If, for any reason, the addressing was unsuccessful, the contents of the variables *MAIL.TO*, *MAIL.CC* or *MAIL.BCC* will be unchanged.

## Parameters

The following table describes the parameters of the WIN.MLADDR command:

Parameter	Description
<i>MAIL.TO</i>	the name of the variable that will receive the names of the primary recipients
<i>MAIL.CC</i>	the name of the variable that will receive the names of the copy recipients
<i>MAIL.BCC</i>	the name of the variable that will receive the names of the blind copy recipients
<i>OPTS</i>	See below
<i>DELIM</i>	Delimiter to separate the returned names.0 = semi-colon (";"), 1 = Field Mark (CHAR(254)), 2 = Value mark (CHAR(253)).

## Values for OPTS

Value	Description
4	Disable Mail user interaction

## Example

Allow user to select other recipients before sending message

---

```
SEND.TO = "Demi "  
SEND.TO<-1> = "Jodie "  
SEND.CC= " "  
SEND.BCC= " "  
CALL WIN.MLADDR(SEND.TO, SEND.CC, SEND.BCC, 0, 1)  
PRINT "You want to send to ":DCOUNT(SEND.TO, CHAR(254)):" people"
```

---

## Version

4.0.1 Original

# WIN.MLAVAIL

## Syntax

WIN.MLAVAIL ( *RESP* )

## Description

This function checks if the mail system and a default user profile are available. It is a convenient way of checking that the mail system is available before using any of the other mail commands or functions.

## Parameters

The following table describes the parameters of the WIN.MLAVAIL command:

Parameter	Description
<i>RESP</i>	"" if the mail system is available, otherwise error text with the reason for failure

## Example

Simple program to check if email is available

---

```
RESP = ""
CALL WIN.MLAVAIL(RESP)
IF RESP = "" THEN
    PRINT "Email is available on this machine"
END ELSE
    PRINT "Email is NOT available on this machine"
    PRINT RESP
END
```

---

## Version

4.2.1 Original

# WIN.MLDELETE

## Syntax

WIN.MLDELETE ( *ID*, *OPTS* )

## Description

This subroutine deletes a mail message.

## Parameters

The following table describes the parameters of the WIN.MLDELETE command:

Parameter	Description
<i>ID</i>	the ID of the message to delete. This ID will have been retrieved using the WIN.MLNEXT or WIN.MLFIND functions.
<i>OPTS</i>	See options below

## Values for OPTS

Value	Description
0	(default)
4	Disable Mail user interaction

## Example

Delete all messages from Sabrina

```
IDS = ""
CALL WIN.MLFIND("Sabrina", 16, IDS)
IF IDS # "" Then
  CNT = COUNT(IDS, CHAR(254))+1
  FOR J = 1 TO CNT
    CALL WIN.MLDELETE(IDS<J>, 0)
  NEXT J
END
```

---

## **Version**

4.0.1 Original

# WIN.MLFIND

## Syntax

WIN.MLFIND ( *PATTERN*, *OPTS*, *IDS* )

## Description

This function finds all available mail messages matching specified criteria. Once the messages have been found they can be read or deleted, using WIN.MLREAD or WIN.MLDELETE , respectively.

The original order option returns messages in the order they are stored in the mail system. This may or may not be chronological order (which is used by default). The reason for this option is that some mail systems may not support chronological order, in which case this function will fail with an error.

Note that if, for example, Match Sender and Match Subject options are selected, the function will return the ID of the first message that contains the specified pattern in either the sender OR the subject.

## Parameters

The following table describes the parameters of the WIN.MLFIND command:

Parameter	Description
<i>PATTERN</i>	The text to search for in a message. The text will be searched for in the location specified in theOptionsparameter. If PATTERN is not supplied (or no search location is specified in the OPTS parameter), the function returns the next message meeting the search criteria specified in the OPTS parameter but without performing any pattern matching. Pattern matching is not case-sensitive.
<i>OPTS</i>	See options below
<i>IDS</i>	Variable to return the list of ids in, field mark (CHAR(254)) separated.

## Values for OPTS

Value	Description
0	Search all messages (default)
1	Search unread messages only
2	Original order
4	Disable Mail user interaction
8	Reverse order of results
16	Match sender
32	Match subject
64	Match attachment
128	Match time

## Example

Count the number of unread messages on the subject of Dilbert

---

```
IDS=""  
CALL WIN.MLFIND("Dilbert", 33, IDS)  
PRINT DCOUNT(IDS, CHAR(254))
```

---

List the titles of all unread messages

---

```
IDS = ""  
CALL WIN.MLFIND("", 1, IDS)  
IF IDS # "" THEN  
  CNT = COUNT(IDS, CHAR(254)) + 1  
  MSG.HDR = ""  
  TEXT = ""  
  ATTS = ""  
  STATE= ""  
  SUBJECTS = ""  
  FOR J = 1 TO CNT  
    CALL WIN.MLREAD( IDS<J>, MSG.HDR, TEXT, ATTS, STATE, 0 )  
    PRINT MSG.HDR<4>  
  NEXT J  
ELSE  
  PRINT "No unread messages"  
END
```

---

## Version

4.0.1 Original



# WIN.MLLOOKUP

## Syntax

**WIN.MLLOOKUP** ( *NAME*, *ADDRESS*, *OPTS* )

## Description

This subroutine returns details about a mail recipient given a full or partial name.

*NAME* may contain a full name or a partially unresolved name, eg. "Mary Doe", "Mary", respectively. Multiple recipients maybe not be specified.

If, for any reason, details cannot be ascertained the returned *NAME* and *ADDRESS* variables will be empty.

If there is ambiguity over the partially resolved name specified, the mail system may display a dialog box allowing the user to select one of several possible full names.

## Parameters

The following table describes the parameters of the WIN.MLLOOKUP command:

Parameter	Description
<i>NAME</i>	the name of the variable that contains the full or partial name to look up. On return, it will receive the full name of the recipient. If the <i>NAME</i> variable is empty, the mail address book is displayed.
<i>ADDRESS</i>	the name of the variable that will receive the address of the recipient.
<i>OPTS</i>	See options below

## Values for OPTS

The options can be combined by addition

Value	Description
0	(default)

Value	Description
<i>1</i>	Display full details
<i>4</i>	Disable Mail user interaction

## Example

Display Sheila's full name and address

---

```
NAME = "Sheila"
ADDRESS = ""
CALL WIN.MLLOOKUP(NAME, ADDRESS, 0)
PRINT NAME
PRINT ADDRESS
```

---

Display Address Book

---

```
NAME = ""
ADDRESS = ""
CALL WIN.MLLOOKUP(NAME, ADDRESS, 0)
```

---

Display Sandra's full details

---

```
NAME = "Sandra"
ADDRESS = ""
CALL WIN.MLLOOKUP(NAME, ADDRESS, 1)
```

---

Check Sally's name before sending a message

---

```
NAME = "Sally"
ADDR = ""
CALL WIN.MLLOOKUP(NAME, ADDR, 0)
IF NAME # "" THEN
    CALL WIN.MLSEND(NAME, "", "", "Greetings!", "How about a drink
after work?", "", 0, RESP)
END
```

---

## Version

4.0.1 Original

# WIN.MLNEXT

## Syntax

**WIN.MLNEXT** ( *ID*, *Pattern*, *OPTS* )

## Description

This function finds the next available mail message matching specified criteria. It is intended to be used iteratively so that all mail messages may be enumerated. Once a message has been found it can be read or deleted, using WIN.MLREAD or WIN.MLDELETE , respectively.

The Original Order option returns messages in the order they are stored in the mail system. This may or may not be chronological order (which is used by default). The reason for this option is that some mail systems may not support chronological order, in which case this function will fail with an error.

Note that if, for example, Match Sender and Match Subject options are selected, the function will return the ID of the first message that contains the specified pattern in either the sender OR the subject.

## Parameters

The following table describes the parameters of the WIN.MLNEXT command:

Parameter	Description
<i>ID</i>	the ID of the message to start searching from. If it is not set (ie. "") then the first message matching the criteria specified by the Options parameter is read. On return, ID will contain the ID of the message found. If it is not set on return, there are no more matching messages. Message IDs are in an internal mail format and not intended for user display.

Parameter	Description
<i>Pattern</i>	the text to search for in a message. The text will be searched for in the location specified in the OPTS parameter. If PATTERN is not supplied (or no search location is specified in the OPTS parameter), the function returns the next message meeting the search criteria specified in the OPTS parameter but without performing any pattern matching. Pattern matching is not case-sensitive.
<i>OPTS</i>	See options below

## Values for OPTS

Value	Description
<i>0</i>	Search all messages (default)
<i>1</i>	Search unread messages only
<i>2</i>	Original order
<i>4</i>	Disable Mail user interaction
<i>16</i>	Match sender
<i>32</i>	Match subject
<i>64</i>	Match attachment
<i>128</i>	Match time

## Example

Search for, and display, the first unread message from Pamela

---

```
ID = ""
CALL WIN.MLNEXT(ID, "Pam", 17)
IF ID # "" THEN
  CALL WIN.MLREAD(ID, MSG.HDR, TEXT, ATTACHMENTS, STATE, 0)
  PRINT MSG.HDR<4>
  PRINT
  FOR J = 1 TO DCOUNT(TEXT, CHAR(254))
    PRINT TEXT&lt;J>
  NEXT
```

END

---

Delete for the fourth message of June 22nd

---

```
ID = ""
J = 1
LOOP
CALL WIN.MLFIND(ID, "/06/22", 128)
WHILE J < 4 AND ID # "" DO
J = J + 1
REPEAT
IF ID # "" THEN
    CALL WIN.MLDELETE(ID)
END
```

---

## **Version**

4.0.1 Original

# WIN.MLREAD

## Syntax

**WIN.MLREAD** ( *ID*, *MSG.HDR*, *TEXT*, *ATTACHMENTS*, *STATE*, *OPTS* )

## Description

This subroutine reads a mail message.

## Parameters

The following table describes the parameters of the WIN.MLREAD command:

Parameter	Description
<i>ID</i>	the ID of the message to read. This ID will have been retrieved using the WIN.MLNEXT or WIN.MLFIND functions.
<i>MSG.HDR</i>	the name of the variable that will receive a dynamic array of the message header (see below).
<i>TEXT</i>	the name of the variable that will receive the actual message text. cr's will be converted to field marks (CHAR(254)) and tabs to value marks (CHAR(253)).
<i>ATTACHMENTS</i>	the name of the variable that will receive the full path names of any attachment files.
<i>STATE</i>	the name of the variable that will receive the message state (see below).
<i>OPTS</i>	See options below

## Fields for MSG.HDR

Field	Description
<i>1</i>	From
<i>2</i>	To
<i>3</i>	CC

Field	Description
4	Subject
5	Date and Time

## Values for STATE

Value	Description
0	Read
1	Unread
2	File(s) attached

## Values for OPTS

Value	Description
0	(default)
1	Don't mark as read
2	Make message text first attachment
4	Disable Mail user interaction

## Example

Display subject and text from the first unread message

---

```
ID = ""
CALL WIN.MLFIND(ID, "", 1)
IF ID # "" THEN
  CALL WIN.MLREAD(ID, HDR, TEXT, ATTACHMENTS, STATE, 0)
  PRINT HDR<4>
  PRINT
  FOR J = 1 TO DCOUNT(TEXT, CHAR(254))
    PRINT TEXT<J>
  NEXT J
END
```

---

### Display attachment file from specific message

---

```
CALL WIN.MLFIND(ID, "Map of Treasure", 32)
IF ID # "" THEN
    CALL WIN.MLREAD(ID, HDR, TEXT, ATTACHMENT, STATE, 0)
    CALL WIN.COMSUB("Run ':'ATTACHMENT:''")
    * We really ought to delete the file after viewing it
END ELSE
    PRINT "Sorry, no treasure!"
END
```

---

## Version

4.0.1 Original



# WIN.MLSEND

## Syntax

**WIN.MLSEND** ( *MAIL.TO*, *MAIL.CC*, *MAIL.BCC*, *SUBJECT*, *TEXT*,  
*ATTACHMENTS*, *OPTS*, *RESP* )

## Description

This command sends a mail message.

Recipient names can be full names, partially unresolved names or full email addresses, eg. "Mary Doe", "Mary", "marydoe@widgets.com", respectively. Multiple recipients maybe specified by separating them with a semi-colon, field mark (CHAR(254)) or value mark (CHAR(253)).

Text can be split into multiple lines by either a field mark (CHAR(254)) or a value mark (CHAR(253)).

Multiple attachment files maybe specified by separating them with a semi-colon, field mark (CHAR(254)) or value mark (CHAR(253)).

## Parameters

The following table describes the parameters of the WIN.MLSEND command:

Parameter	Description
<i>MAIL.TO</i>	the names of the primary recipients. If this parameter contains valid names, the message will be sent without user interaction (unless an option is specified to prevent this).
<i>MAIL.CC</i>	the names of the copy recipients
<i>MAIL.BCC</i>	the names of the blind copy recipients
<i>SUBJECT</i>	the title or subject of the message
<i>TEXT</i>	the actual message text
<i>ATTACHMENTS</i>	the full path names of any attachment files
<i>OPTS</i>	See below

Parameter	Description
<i>RESP</i>	If an error occurs this parameter will contain text describing what went wrong. There will be NO indication of error if this parameter is not used.

## Values for OPTS

Value	Description
<i>0</i>	(default)
<i>1</i>	Review message before sending
<i>4</i>	Disable Mail user interaction (overrides 1)

## Example

### Tell grunts to attend a meeting

---

```
RESP=""  
CALL WIN.MLSEND("Bill;Ben", "", "", "Meeting at 10:30 today in my  
Office", "Attend or else!", "", 0, RESP)
```

---

### Ask managers to attend a meeting

---

```
SEND.TO = "Algernon"  
SEND.TO<-1> = "Emily"  
SEND.TO<-1> = "Nigel"  
SEND.CC = "Caterers"  
SUBJECT = "Meeting at 12:00 today at the Swimming Pool Bar"  
AM = CHAR(254)  
TEXT = "Dear Friends" : AM : AM  
TEXT = TEXT : "Would you mind attending this meeting?" : AM  
TEXT = TEXT : "Thank you so much." : AM : AM  
TEXT = TEXT : "Your friend, Uriah."  
RESP = ""  
CALL WIN.MLSEND(SEND.TO, SEND.CC, "", SUBJECT, TEXT, "c:\Temp\Country  
Club Map.gif", 1, RESP)  
IF RESP # "" THEN  
    PRINT RESP  
END
```

---

## **Related Script Commands**

Mail Send

## **Version**

4.0.1 Original version

# WIN.MOUSE

## Syntax

WIN.MOUSE ( *KEYS*, *DEFS* )

## Description

This subroutine programs mouse buttons.

## Parameters

The following table describes the parameters of the WIN.MOUSE command:

Parameter	Description
<i>KEYS</i>	Mouse buttons to define. A multi-field dynamic array matched to DEFS. The definitions can contain mouse button numbers or names. See the table below for the mouse numbers/names.
<i>DEFS</i>	Definitions for mouse buttons.

## Values for KEYS

Use the following numbers or names for the mouse buttons

Value	Description
<i>1</i>	MouseLeft
<i>2</i>	MouseRight
<i>3</i>	MouseBoth
<i>4</i>	Shift_MouseLeft
<i>5</i>	Shift_MouseRight
<i>6</i>	Shift_MouseBoth
<i>7</i>	Control_MouseLeft
<i>8</i>	Control_MouseRight
<i>9</i>	Control_MouseBoth

Value	Description
10	ControlShift_MouseLeft
11	ControlShift_MouseRight
12	ControlShift_MouseBoth

## Example

\* Program MouseLeft, MouseRight and MouseBoth

---

```
* Program MouseLeft, MouseRight and MouseBoth
KEYS = '';DEFS = ''
KEYS<1> = 1;DEFS<1> = 'Left Mouse button clicked'
KEYS<2> = 2;DEFS<2> = 'Right Mouse button clicked'
KEYS<3> = 3;DEFS<3> = 'Both mouse buttons clicked'
CALL WIN.MOUSE(KEYS,DEFS)
```

---

Assign the popup copymenu to the right mouse button

---

The copymenu script is in the application directory, otherwise the full path must be specified.

\* Assign copymenu to right mouse button

```
CALL WIN.MOUSE("MouseRight", "\mScript 'example\script\copymenu'")
```

---

## Related Subroutines

[WIN.MOUSEDEF](#), [WIN.MOUSEIN](#)

## Related Script Commands

Set MouseLeft, Set MouseRight, Set MouseBoth

# WIN.MOUSEDEF

## Syntax

WIN.MOUSEDEF

## Description

This is used when the host needs to know when and where mouse buttons are clicked. It programs the mouse buttons so they enter the x and y coordinates and a button number when the mouse is clicked on the wIntegrate screen. The necessary values are hard-coded within this routine, and overwrite the current settings. The format is recognized by the WIN.MOUSEIN command and returned. See the example for the WIN.MOUSEIN command.

## Parameters

None

## Related Subroutines

[WIN.MOUSEIN](#)

# WIN.MOUSEIN

## Syntax

**WIN.MOUSEIN** ( *X*, *Y*, *BUTTON* )

## Description

This subroutine waits until a mouse button is pressed and then splits the result into the position and a button type. For this routine to work, you must first call the WIN.MOUSEDEF routine, which programs the mouse buttons.

## Parameters

The following table describes the parameters of the WIN.MOUSEIN command:

Parameter	Description
<i>X</i>	The column number returned.
<i>Y</i>	The row number returned.
<i>BUTTON</i>	The button type returned. See the Button table.

## Values for BUTTON

The following button numbers are returned

Value	Description
<i>1</i>	Left
<i>2</i>	Right
<i>3</i>	Both
<i>+3</i>	Shift plus the above
<i>+6</i>	Control plus the above
<i>+9</i>	Control and shift plus the above

## Example

Wait for Control and mouse left to be clicked on line 23

---

```
*Set up the mouse
CALL WIN.MOUSEDEF
LOOP
CALL WIN.MOUSEIN(X.Y.BUTTON)
* End the loop when Control+MouseLeft button is clicked on line 23
UNTIL Y = 23 AND BUTTON = 7 DO
.
.
.
REPEAT
```

---

## Related Subroutines

[WIN.MOUSEDEF](#)



# WIN.MSGBOX

## Syntax

**WIN.MSGBOX** ( *TEXT*, *TITLE*, *FLAGS*, *RESP* )

## Description

This subroutine creates and displays a message box. You must specify the text, icon and response buttons to display.

## Parameters

The following table describes the parameters of the WIN.MSGBOX command:

Parameter	Description
<i>TEXT</i>	The text to display in the message box.
<i>TITLE</i>	The title for the message box. If you do not specify, the default is Message”.
<i>FLAGS</i>	Icon and Response buttons to display in the box. (See the table below) - If <i>FLAGS</i> <2> is set to 1, it will map characters depending emulation.
<i>RESP</i>	The response from the user.

## Values for FLAGS

Use one icon and one button flag together separated by a vertical bar (|).

Value	Description
<i>MB_OK</i>	Displays an “OK” button.
<i>MB_OKCANCEL</i>	Displays “OK” and “Cancel” buttons.
<i>MB_ABORTRETRYIGNORE</i>	Displays “Abort”, “Retry”, and “Ignore” buttons.
<i>MB_YESNOCANCEL</i>	Displays “Yes”, “No”, and “Cancel” buttons.
<i>MB_YESNO</i>	Displays “Yes” and “No” buttons.

Value	Description
<i>MB_RETRYCANCEL</i>	Displays “Retry” and “Cancel” buttons.
<i>MB_DEFBUTTON1</i>	Makes the first button the default button.
<i>MB_DEFBUTTON2</i>	Makes the second button the default button.
<i>MB_DEFBUTTON3</i>	Makes the third button the default button.
<i>MB_ICONHAND</i>	Displays the stop sign icon.
<i>MB_ICONQUESTION</i>	Displays the question icon.
<i>MB_ICONEXCLAMATION</i>	Displays the exclamation warning icon.
<i>MB_ICONASTERISK</i>	Displays the asterisk information icon.

## Example

This example demonstrates how to show two message boxes within a program. See the message

---

```
* Show a message box
CALL WIN.MSGBOX("Create report?", " ", "MB_OKCANCEL | MB_ICONHAND",
RESP)
IF RESP = "Yes" THEN GOSUB 100
100
* Record a report to file "recrept"
CALL WIN.RECON("recrept.txt", 1)
* Run a report
EXECUTE "LIST STUDENT LNAME MAJOR"
CALL WIN.STATLINE("Creating report...")
* Turn off recording
CALL WIN.RECOFF(1)
CALL WIN.STATLINE("")
CALL WIN.MSGBOX("Display the report?", "Student Report",
"MB_YESNOCANCEL | MB_ICONQUESTION ", RESP)
IF RESP = "Yes" THEN GOSUB 200
RETURN
200
* Display the report on the screen
CALL WIN.PLAYBACK("recrept.txt")
RETURN
END
```

---

## **Related Subroutines**

[WIN.INFOBOX](#), [WIN.INPBOX](#), [WIN.INFOBOX](#), [WIN.INPBOX](#)

## **Related Script Commands**

MessageBox

# WIN.MSTATE

## Syntax

WIN.MSTATE ( *NAME*, *OPTION*, *CHECK*, *ENABLE* )

## Description

This subroutine modifies an option on the menu. The *CHECK* parameter adds or deletes a check-mark next to the option. *ENABLE* makes a menu option available or makes it gray to disable it.

## Parameters

The following table describes the parameters of the WIN.MSTATE command:

Parameter	Description
<i>NAME</i>	The name of the menu.
<i>OPTION</i>	The option number.
<i>CHECK</i>	1 = Option is checked 0 = Option is not checked
<i>ENABLE</i>	1 = Enables the option 0 = Disables the option. (Makes it gray)

## Example

This example is part of the WIN.MENUDEMO demonstration program.

---

```
* Attaching/Detaching in action
1900 IF MENU.SHOW = 1 THEN
  CALL WIN.MENUATT( ' ', "Demo", "&Demo", "Help" )
  CALL WIN.MSTATE( "Demo", 9, 1, 1 )
  MENU.SHOW = 2
  PRINT C23: "Select menu option from the Demo menu on the main menu
bar": END ELSE
  CALL WIN.MENUDET( ' ', "Demo" )
  CALL WIN.MSTATE( "Demo", 9, 0, 1 )
  MENU.SHOW = 1
  PRINT C23: "Select the menu option from the Popup menu":
END
RETURN
```

---

## **Related Subroutines**

[WIN.MENUDEL](#), [WIN.MENUDET](#), [WIN.MENUATT](#)

# WIN.OBEXIST

## Syntax

WIN.OBEXIST ( *NAME*, *RESP* )

## Description

This subroutine returns TRUE if an object with the given name exists.

## Parameters

The following table describes the parameters of the WIN.OBEXIST command:

Parameter	Description
<i>NAME</i>	The name of the object.
<i>RESP</i>	1 - if object exist, 0 if it doesn't

## Version

4.0.3 Original

# WIN.OBGET

## Syntax

**WIN.OBGET** ( *NAME*, *FILENAME*, *PROGID*, *RESP* )

## Description

This subroutine gets an automation object from a file or retrieves an active object. Call WIN.OBREL to release the object when it has been finished with.

## Parameters

The following table describes the parameters of the WIN.OBGET command:

Parameter	Description
<i>NAME</i>	The name of the object.
<i>FILENAME</i>	Name of the file to get the object from. If omitted an active object with the <b>PROGID</b> specified is retrieved.
<i>PROGID</i>	The automation object or its class ID in {}. If omitted the default object for the specified filename is returned.
<i>RESP</i>	0 or the error code (see table in Object Get script command)

## Related Subroutines

[WIN.OBREL](#)

## Related Script Commands

Object Get

## Version

4.0.3 Original



# WIN.OBGETPRP

## Syntax

**WIN.OBGETPRP** ( *NAME*, *PROPERTY*, *VALUE* )

## Description

This subroutine gets the value for an objects property.

## Parameters

The following table describes the parameters of the WIN.OBGETPRP command:

Parameter	Description
<i>NAME</i>	Controls name
<i>PROPERTY</i>	Name of the property to set
<i>VALUE</i>	The value returned for the property

## Version

4.0.3 Original

# WIN.OBMETHOD

## Syntax

**WIN.OBMETHOD** ( *NAME*, *METHOD*, *ARG.FLAGS*, *MAT ARGS*, *RET.FLAG*, *RET.VALUE* )

## Description

This subroutine runs a method on an automation object.

## Parameters

The following table describes the parameters of the WIN.OBMETHOD command:

Parameter	Description
<i>NAME</i>	The objects name
<i>METHOD</i>	The name of the method to run
<i>ARG.FLAGS</i>	String describing the arguments. See below
<i>MAT ARGS</i>	20 element array to hold the arguments
<i>RET.FLAG</i>	Flag to specify if the method returns a value.0 - No return value1 - Returns a value2 - RET.VALUE is the name of an object to be attached to the return value
<i>RET.VALUE</i>	Value returned from the method if RET.FLAG is set to 1, or name of new object if RET.FLAG is set to 2.

## Values for ARG.FLAGS

Value	Description
<i>I</i>	Input argument. The value from the corresponding element of ARGS is used to run the method

Value	Description
<i>O</i>	Output argument. The value from the corresponding element of ARGS is set to the value returned from the method for this argument
<i>B</i>	Both ways. The value from the corresponding element of ARGS is used to run the method and the value of the argument is returned after the method has run
<i>N</i>	Input argument (name). Value is the name of a script variable.
<i>D</i>	Input argument (dispatch pointer). Value is the name of an existing object.

## Example

See host program WIN.OBDEMO

## Version

4.0.3 Original

# WIN.OBNEW

## Syntax

WIN.OBNEW ( *NAME*, *PROGID*, *RESP* )

## Description

This subroutine create a new automation object. Call WIN.OBREL to release the object when it has been finished with.

## Parameters

The following table describes the parameters of the WIN.OBNEW command:

Parameter	Description
<i>NAME</i>	The name of the object.
<i>PROGID</i>	The automation object or its class ID in { }.
<i>RESP</i>	0 or the error code (see table in Object New script command)

## Related Subroutines

[WIN.OBREL](#)

## Related Script Commands

Object New

## Version

4.0.3 Original

# WIN.OBREL

## Syntax

**WIN.OBREL** ( *NAME* )

## Description

This subroutine releases an automation object. It should be called on any automation objects you use. This includes objects created with WIN.OBNEW or WIN.OBGET or retrieved from properties or method calls using WIN.OBSET or WIN.OBMETHOD.

## Parameters

The following table describes the parameters of the WIN.OBREL command:

Parameter	Description
<i>NAME</i>	The name of the object.

## Related Subroutines

[WIN.OBNEW](#), [WIN.OBSET](#), [WIN.OBGET](#), [WIN.OBMETHOD](#)

## Related Script Commands

Object Release

## Version

4.0.3 Original

# WIN.OBSET

## Syntax

**WIN.OBSET** ( *NAME*, *PROPERTY*, *VALUE* )

## Description

This subroutine sets a new object with the value of an objects property. Use this when an objects property returns a dispatch pointer to create a wIntegrate object which uses the dispatch pointer.

## Parameters

The following table describes the parameters of the WIN.OBSET command:

Parameter	Description
<i>NAME</i>	Objects name
<i>PROPERTY</i>	Name of the property to which returns a dispatch pointer
<i>VALUE</i>	New value of the property

## Related Script Commands

Object Set

## Version

4.0.3 Original

# WIN.OBSETPRP

## Syntax

**WIN.OBSETPRP** ( *NAME*, *PROPERTY*, *VALUE* )

## Description

This subroutine sets the value for an objects property.

## Parameters

The following table describes the parameters of the WIN.OBSETPRP command:

Parameter	Description
<i>NAME</i>	Objects name
<i>PROPERTY</i>	Name of the property to set
<i>VALUE</i>	New value of the property

## Version

4.0.3 Original

# WIN.OBVTYPE

## Syntax

WIN.OBVTYPE ( *TYPE*, *VALUE* )

## Description

This routine modifies *VALUE* so that it is treated as a specific type when used as an argument for WIN.OBMETHOD. See the script command VarType for details.

## Parameters

The following table describes the parameters of the WIN.OBVTYPE command:

Parameter	Description
<i>TYPE</i>	The type to return the object in
<i>VALUE</i>	The value to be converted

## Related Subroutines

[WIN.OBMETHOD](#)

## Related Script Commands

VarType

## Version

4.0.3 Original



# WIN.PCBROWSE

## Syntax

**WIN.PCBROWSE** ( *PROMPT*, *DEFAULTFILE*, *OPTS*, *TYPES*, *EXT*, *RESULT* )

## Description

This subroutine shows the file open/save dialog box on the PC and allows the user to select a filename.

## Parameters

The following table describes the parameters of the WIN.PCBROWSE command:

Parameter	Description
<i>PROMPT</i>	The title to display in the title of the File Browse dialog box.
<i>DEFAULTFILE</i>	The default file name to be displayed when the dialog box is shown.
<i>OPTS</i>	Options: 0 show open dialog box, 1 show save as dialog box.
<i>TYPES</i>	The file types to show in the dialog box. List the name of the file type first, then the extension for this type separated by a " ".
<i>EXT</i>	The default extension to use if a file name without an extension is entered in the file name field.
<i>RESULT</i>	Name of a variable to set with the filename chosen by the user. If the user pressed the cancel button a null ("" ) filename will be returned.

## Example

Browse using the defaults

---

```
CALL WIN.PCBROWSE("Browse using defaults","", "", "", "", RESP)
PRINT "File name entered = ":RESP
```

---

### Browse from 'c:\'

---

```
CALL WIN.PCBROWSE('Browse with "Save"', "C:\*.*",1,""," ",RESP)
```

---

### Browse with types and default extension

---

```
TYPES = "Text 'Files' | *.txt | "  
TYPES = TYPES : 'All "Files" | *.*'  
*  
CALL WIN.PCBROWSE("Browse with types", "C:\*.TXT",1,TYPES, "TXT",  
RESP)  
PRINT "File name entered = ":RESP
```

---

## Related Script Commands

FileBrowse

## Version

4.3/5.01 Original

# WIN.PCCLOSE

## Syntax

WIN.PCCLOSE ( *NAME* )

## Description

This subroutine closes a PC file that was opened by WIN.PCOPEN.

## Parameters

The following table describes the parameters of the WIN.PCCLOSE command:

Parameter	Description
<i>NAME</i>	The identifier given to the file in WIN.PCOPEN

## Example

Write text to a file.

---

```
CALL WIN.PCOPEN("new", "C:\ptest\pnew.txt", OK)
IF OK THEN
  * open pnew.txt
  NEW.DEF = "This is written by the WIN.PCWRITE subroutine"
  CALL WIN.PCWRITE("new", NEW.DEF, "TEXT")
  * Close the file
  CALL WIN.PCCLOSE("new")
END
```

---

## Related Subroutines

[WIN.PCOPEN](#), [WIN.PCREAD](#), [WIN.PCWRITE](#)

## Related Script Commands

File Close

# WIN.PCCOPY

## Syntax

WIN.PCCOPY ( *FROMFILE*, *TOFILE*, *OVERWRITE* )

## Description

This subroutine makes a copy of a file on the PC.

## Parameters

The following table describes the parameters of the WIN.PCCOPY command:

Parameter	Description
<i>FROMFILE</i>	The name of the file to copy.
<i>TOFILE</i>	The name for the copy of the file.
<i>OVERWRITE</i>	1 to overwrite any existing file with the same name, otherwise 0.

## Related Script Commands

File Copy

## Version

4.3/5.0.1 Original

# WIN.PCCREATE

## Syntax

WIN.PCCREATE ( *FILE* )

## Description

This subroutine creates a PC file. Use this call if your application needs to create a file. Then use the WIN.PCDELETE subroutine to delete it when it is no longer necessary.

The subroutine assumes the file should be created in the wintegrate directory if the full path name of the file is not specified.

## Parameters

The following table describes the parameters of the WIN.PCCREATE command:

Parameter	Description
<i>FILE</i>	The name you specify for the new PC file. Include the full path name if the file is not created in the wintegrate directory.

## Example

\* Demo program for WIN.PC subroutines

---

```
* Make a new directory C:\ptest
CALL WIN.PCMKDIR("C:\ptest")
*
* Verify the PC directory C:\ptest
CALL WIN.PCDIR("C:\ptest", PCTEST.FOUND)
IF PCTEST.FOUND THEN
*
* Check if pcnew.txt exists in the pctest directory
CALL WIN.PCFILE("C:\ptest\pcnew.txt", EXIST)
IF EXIST THEN GOSUB 100 ELSE GOSUB 200
STOP
100
CALL WIN.PCEDIT("C:\ptest\pcnew.txt")
RETURN
200
```

```
CALL WIN.PCCREATE("C:\ptest\pnew.txt")  
CALL WIN.PCEDIT("C:\ptest\pnew.txt")  
RETURN  
END
```

---

## **Related Subroutines**

[WIN.PCEDIT](#), [WIN.PCFILE](#)

# WIN.PCDELETE

## Syntax

WIN.PCDELETE ( *FILE* )

## Description

This subroutine deletes a PC file. Use this call if your application needs to create a temporary file and you want to delete it when it is no longer necessary.

The subroutine assumes the file is in the wintegrate directory if the full path name of the file is not specified.

## Parameters

The following table describes the parameters of the WIN.PCDELETE command:

Parameter	Description
<i>FILE</i>	The name of the PC file to delete. Include the full path name if the file is not created in the wintegrate directory.

## Example

This example deletes the file created in the WIN.PCCREATE example program.

```
* Check if pcnew.txt exists in the pctest directory
CALL WIN.PCFILE("C:\pctest\pcnew.txt", EXIST)
IF EXIST THEN
    CALL WIN.PCDELETE("C:\pctest\pcnew.txt")
END
```

## Related Subroutines

[WIN.PCCREATE](#)

# WIN.PCDIR

## Syntax

WIN.PCDIR ( *DIR*, *EXIST* )

## Description

This subroutine verifies that a PC directory exists.

The subroutine assumes the directory is in the wintegrate directory if the full path name is not specified.

## Parameters

The following table describes the parameters of the WIN.PCDIR command:

Parameter	Description
<i>DIR</i>	The name of the PC directory to verify.
<i>EXIST</i>	Returns a 1 if the directory exists, 0 if it does not.

## Example

\* Demo program for WIN.PC subroutines

---

```
* Make a new directory C:\ptest
CALL WIN.PCMKDIR("C:\ptest")
*
* Verify the PC directory C:\ptest
CALL WIN.PCDIR("C:\ptest", PCTEST.FOUND)
IF PCTEST.FOUND THEN
*
* Check if pcnew.txt exists in the pctest directory
CALL WIN.PCFILE("C:\ptest\pcnew.txt", EXIST)
IF EXIST THEN GOSUB 100 ELSE GOSUB 200
STOP
100 CALL WIN.PCEDIT("C:\ptest\pcnew.txt")
RETURN
200 CALL WIN.PCCREATE("C:\ptest\pcnew.txt")
CALL WIN.PCEDIT("C:\ptest\pcnew.txt")
RETURN
END
```

---



## **Related Subroutines**

[WIN.PCMKDIR](#), [WIN.PCCREATE](#)

# WIN.PCEDIT

## Syntax

WIN.PCEDIT ( *FILE* )

## Description

This subroutine opens a PC file into the editor specified in Setup Application.

The subroutine assumes the file is in the wintegrate directory if the full path name of the file is not specified.

## Parameters

The following table describes the parameters of the WIN.PCEDIT command:

Parameter	Description
<i>FILE</i>	The name of the PC file to edit. Include the full path name if the file is not created in the wintegrate directory.

## Example

Demo program for WIN.PC subroutines

```
* Make a new directory C:\pctest
CALL WIN.PCMKDIR("C:\pctest")
*
* Verify the PC directory C:\pctest
CALL WIN.PCDIR("C:\pctest", PCTEST.FOUND)
IF PCTEST.FOUND THEN
*
* Check if pcnew.txt exists in the pctest directory
CALL WIN.PCFILE("C:\pctest\pcnew.txt", EXIST)
IF EXIST THEN GOSUB 100 ELSE GOSUB 200
STOP
100
CALL WIN.PCEDIT("C:\pctest\pcnew.txt")
RETURN
200
CALL WIN.PCCREATE("C:\pctest\pcnew.txt")
CALL WIN.PCEDIT("C:\pctest\pcnew.txt")
RETURN
```

END

---

## **Related Subroutines**

[WIN.PCCREATE](#), [WIN.PCFILE](#)

## **Related Script Commands**

Dialog FileEdit, Set, Invoke

# WIN.PCEOF

## Syntax

WIN.PCEOF ( *NAME*, *ATEOF* )

## Description

This command checks a file opened by WIN.PCOPEN to see if it is currently at the end of file.

## Parameters

The following table describes the parameters of the WIN.PCEOF command:

Parameter	Description
<i>NAME</i>	Name used to open the file
<i>ATEOF</i>	Returns 1 when at the end of the file, otherwise 0

## Related Subroutines

[WIN.PCOPEN](#)

## Version

4.1 Original version

# WIN.PCFILE

## Syntax

WIN.PCFILE ( *FILE*, *EXIST* )

## Description

This subroutine verifies the existence of a PC file.

## Parameters

The following table describes the parameters of the WIN.PCFILE command:

Parameter	Description
<i>FILE</i>	The full path name of the PC file to verify.
<i>EXIST</i>	Returns a 1 if the file exists, 0 if it does not.

## Example

\* Verify the PC file C:\windows\system\sysedit.exe

---

```
* Verify the PC file C:\windows\system\sysedit.exe
CALL WIN.PCFILE("C:\windows\system\sysedit.exe", SYSEdit.OK)
IF SYSEdit.OK THEN PRINT "Run sysedit.exe"
```

---

The following example demonstrates how this subroutine works where the file does not already exist.

---

```
* Demo program for WIN.PC subroutines
* Make a new directory C:\pctest
CALL WIN.PCMKDIR("C:\pctest")
*
* Verify the PC directory C:\pctest
CALL WIN.PCDIR("C:\pctest", PCTest.FOUND)
IF PCTest.FOUND THEN
*
* Check if pcnew.txt exists in the pctest directory
CALL WIN.PCFILE("C:\pctest\pcnew.txt", EXIST)
IF EXIST THEN GOSUB 100 ELSE GOSUB 200
STOP
100
CALL WIN.PCEDIT("C:\pctest\pcnew.txt")
RETURN
```

```
200
CALL WIN.PCCREATE("C:\ptest\pnew.txt")
CALL WIN.PCEDIT("C:\ptest\pnew.txt")
RETURN
END
```

---

## Related Subroutines

[WIN.PCCREATE](#), [WIN.PCEDIT](#)

## Related Script Commands

FileExist

# WIN.PCINFO

## Syntax

**WIN.PCINFO** ( *FILE*, *INFO* )

## Description

This subroutine returns information about a file or directory. If the file does not exist, it returns Null.

## Parameters

The following table describes the parameters of the WIN.PCINFO command:

Parameter	Description
<i>FILE</i>	The name of the file on which you want information.
<i>INFO</i>	The information returned is in the following order: 1 = file length 2 = date of last update 4 = time of last update 8 = attributes

## Example

\* Get info on pcnew.txt

---

```
* Get info on pcnew.txt
PC.INFO = ""
CALL WIN.PCINFO("C:\pctest\pcnew.txt", PC.INFO)
PRINT "Information on file pcnew.txt " : PC.INFO
* This program returns the following information: Information on file
pcnew.txt
* 34 02/02/1996 14:25:12 A
```

---

## Related Script Commands

FileInfo

# WIN.PCLIST

## Syntax

WIN.PCLIST ( *FILESPEC*, *OPTS*, *RESP* )

## Description

This command checks a folder and returns a list of the files or subdirectories matching the criteria that you specify in a dynamic array.

## Parameters

The following table describes the parameters of the WIN.PCLIST command:

Parameter	Description
<i>FILESPEC</i>	The files to return. This can be a wild card expression to match files
<i>OPTS</i>	Specifies other matching options. See the following Options table for more information.
<i>RESP</i>	A dynamic array of the files/folders which match the specification

## Values for OPTS

The options field specifies which files or subdirectories to match. The default option is 0. The options 4, 8, 16, and 32 can be used alone or added to other option numbers

Value	Description
0	Returns files only.
1	Returns files and directories.
2	Returns files, and includes directories in square brackets
4	Returns only subdirectories.
8	Returns only hidden files.
16	Returns only system files.



Value	Description
32	Do not include current directory.

## Related Script Commands

FileList

## Version

4.0.3 Original version

# WIN.PCMKDIR

## Syntax

WIN.PCMKDIR ( *DIR* )

## Description

This subroutine creates a new PC directory. Use this with the WIN.PCDIR subroutine to verify that the directory does not already exist.

## Parameters

The following table describes the parameters of the WIN.PCMKDIR command:

Parameter	Description
<i>DIR</i>	The name of the directory to create. Include the full path name if the file is not created in the wintegrate directory.

## Example

\* Demo program for WIN.PC subroutines

---

```
* Make a new directory C:\pctest
CALL WIN.PCMKDIR("C:\pctest")
*
* Verify the PC directory C:\pctest
CALL WIN.PCDIR("C:\pctest", PCTEST.FOUND)
IF PCTEST.FOUND THEN
*
* Check if pcnew.txt exists in the pctest directory
CALL WIN.PCFILE("C:\pctest\pcnew.txt", EXIST)
IF EXIST THEN GOSUB 100 ELSE GOSUB 200
STOP
100
CALL WIN.PCEDIT("C:\pctest\pcnew.txt")
RETURN
200
CALL WIN.PCCREATE("C:\pctest\pcnew.txt")
CALL WIN.PCEDIT("C:\pctest\pcnew.txt")
RETURN
END
```

---

## **Related Subroutines**

[WIN.PCDIR](#), [WIN.PCCREATE](#)

# WIN.PCMOVE

## Syntax

WIN.PCMOVE ( *FROM.FILE*, *TOFILE*, *OVERWRITE* )

## Description

This subroutine moves/renames a file on the PC.

## Parameters

The following table describes the parameters of the WIN.PCMOVE command:

Parameter	Description
<i>FROM.FILE</i>	File to move/rename.
<i>TOFILE</i>	Destination/new name for the file.
<i>OVERWRITE</i>	1 to overwrite the destination file otherwise 0.

## Related Script Commands

File Move

## Version

4.3/5.0.1 Original

# WIN.PCOPEN

## Syntax

**WIN.PCOPEN** ( *NAME*, *FILE*, *OK* )

## Description

This subroutine opens a PC file. To close the file, use WIN.PCCLOSE.

## Parameters

The following table describes the parameters of the WIN.PCOPEN command:

Parameter	Description
<i>NAME</i>	The identifier you give to the PC file.
<i>FILE</i>	The name of the PC file to open.
<i>OK</i>	Returns 1 if the file exists, 0 if not.

## Example

Verifies if pcnew.txt is open

---

```
CALL WIN.PCOPEN("new", "C:\pctest\pcnew.txt", OK)
IF OK THEN GOSUB 100
100 *open pcnew.txt
NEW.DEF = "This is written by the WIN.PCWRITE subroutine"
CALL WIN.PCWRITE("new", NEW.DEF, "TEXT")
* Close the file
CALL WIN.PCCLOSE("new")
RETURN
END
```

---

## Related Subroutines

[WIN.PCCLOSE](#), [WIN.PCREAD](#), [WIN.PCWRITE](#)

## Related Script Commands

File Open

# WIN.PCPOS

## Syntax

WIN.PCPOS ( *NAME*, *POS* )

## Description

This routine moves the file pointer a number of bytes into a file opened by WIN.PCOPEN. The position is the number of bytes from the beginning of the file. The first character in the file is 0. To move to the end of the file, use position -1. The default is -1, the end of the file.

## Parameters

The following table describes the parameters of the WIN.PCPOS command:

Parameter	Description
<i>NAME</i>	The identifier for the file, designated in WIN.PCOPEN.
<i>POS</i>	The number of bytes from the beginning of the file. The character in the file is 0. To move to the end of the file, position -1. The default is the end of the file.

## Example

Verifies if pcnew.txt is open

---

```
CALL WIN.PCOPEN("new", "C:\pctest\pcnew.txt", OK)
IF OK THEN GOSUB 100
100 *open pcnew.txt
NEW.DEF = "This is written by the WIN.PCWRITE subroutine"
CALL WIN.PCPOS("new", 0) ; * Go to the start of the file
CALL WIN.PCWRITE("new", NEW.DEF, "TEXT")
* Close the file CALL WIN.PCCLOSE("new")
RETURN
END
```

---

## **Related Subroutines**

[WIN.PCOPEN](#), [WIN.PCREAD](#), [WIN.PCWRITE](#), [WIN.PCCLOSE](#)

## **Related Script Commands**

File Pointer

# WIN.PCPRINT

## Syntax

WIN.PCPRINT ( *FILE* )

## Description

This subroutines prints a PC file to the printer designated in File Printer Setup. If the file is not in the wIntegrate directory, you must specify the full path of the file.

## Parameters

The following table describes the parameters of the WIN.PCPRINT command:

Parameter	Description
<i>FILE</i>	The name of the file to print.

## Example

Print the new pcnew.txt file

---

```
CALL WIN.PCPRINT( "C:\ptest\pcnew.txt" )
```

---

## Related Subroutines

[WIN.PCEDIT](#)



# WIN.PCREAD

## Syntax

**WIN.PCREAD** ( *NAME*, *VAR*, *MAX.BYTES*, *CONVERT* )

## Description

Reads data from a file opened with the WIN.PCOPEN routine. There is a maximum of 15K of data that can be read in one call to this routine. The file pointer is moved to the end of the file so that subsequent calls to the routine will read more data from the file.

Note This routine has no error checking when the data is transferred.

## Parameters

The following table describes the parameters of the WIN.PCREAD command:

Parameter	Description
<i>NAME</i>	The identifier for the file designated in WIN.PCOPEN
<i>VAR</i>	The variable for data to read to.
<i>MAX.BYTES</i>	The maximum number of bytes to read.
<i>CONVERT</i>	The conversion of the data read. See the table below.

## Values for CONVERT

This table describes the data conversion options for this routine.

Value	Description
<i>""</i>	No conversion, data reads directly to the variable.
<i>"TEXT"</i>	Converts field marks to carriage returns.
<i>"HEX"</i>	Converts data to 2-byte hex format.
<i>"FT"</i>	Converts data in file transfer format.

## Example

Read Excel spreadsheet as hex and store on host file

---

```
CALL WIN.PCOPEN("SHEET", PCFILENAME, OK)
IF OK THEN
  COUNT = 0
  LOOP
    CALL WIN.PCREAD("SHEET", REC, "", "HEX")
    UNTIL REC = "" DO
      COUNT = COUNT + 1
      WRITE REC ON F.SHEETS, PCFILENAME : COUNT
    REPEAT
  CALL WIN.PCCLOSE("SHEET")
END
```

---

## Related Subroutines

[WIN.PCOPEN](#), [WIN.PCPOS](#), [WIN.PCWRITE](#)

## Related Script Commands

File Read

# WIN.PCREADAL

## Syntax

**WIN.PCREADAL** ( *NAME*, *VAR*, *CONVERT* )

## Description

This command reads data from a file opened by WIN.PCOPEN. It reads data from the current file position to the end of the file.

Note: This routine does not check for the maximum size of a variable on the host system.

## Parameters

The following table describes the parameters of the WIN.PCREADAL command:

Parameter	Description
<i>NAME</i>	Name used to open the file
<i>VAR</i>	The variable the data will be read in to
<i>CONVERT</i>	The conversion of the data to be read. See the table in WIN.PCREAD for details

## Example

Read the entire file readme.txt

---

```
CALL WIN.PCOPEN("FH", " C:\readme.txt ", OK)
IF OK THEN
  CALL WIN.PCREADAL("FH",REC,"TEXT")
  CALL WIN.PCCLOSE("FH")
END
```

---

## Related Script Commands

File Read

## Version

4.2.1 Original version

# WIN.PCREADLN

## Syntax

**WIN.PCREADLN** ( *NAME*, *VALUE*, *MAX.BYTES* )

## Description

This command reads a single line from a file opened by WIN.PCOPEN.

## Parameters

The following table describes the parameters of the WIN.PCREADLN command:

Parameter	Description
<i>NAME</i>	Name used to open the file
<i>VALUE</i>	Returned text of the line
<i>MAX.BYTES</i>	Maximum number of bytes to read. "" is the default (256)

## Example

Print out the lines from a file previously opened

---

```
LOOP
  CALL WIN.PCEOF ( FH , EOF )
UNTIL EOF DO
  CALL WIN.PCREADLN ( FH , VALUE , " " )
  PRINT VALUE
REPEAT
```

---

## Related Subroutines

[WIN.PCREAD](#), [WIN.PCREADAL](#)

## Related Script Commands

File Read

## Version

4.1 Original version

# WIN.PCRMDIR

## Syntax

WIN.PCRMDIR ( *DIR* )

## Description

This routine deletes a folder on the PC.

Note: The folder must exist before this routine is called

## Parameters

The following table describes the parameters of the WIN.PCRMDIR command:

Parameter	Description
<i>DIR</i>	The path of the folder to delete. If a full path isn't given the folder is assumed to be a subfolder of the wIntegrate folder

## Related Script Commands

File DeleteDir

# WIN.PCRUN

## Syntax

WIN.PCRUN ( *PROG*, *ARGS* )

## Description

This subroutine runs another Windows application on the PC.

## Parameters

The following table describes the parameters of the WIN.PCRUN command:

Parameter	Description
<i>PROG</i>	The name of the application to run.
<i>ARGS</i>	The path of the application executable and the file of a particular document, if necessary.

## Example

This example is part of the WIN.DDEDEMO demonstration program.

```
* Excel DDE demonstration
200 CALL WIN.COLOR("Yellow","Blue")
CALL WIN.TWOPEN("ddedemo", "Excel Dynamic Data Exchange demo",10, 5,
70, 16,1)
PRINT "To find out the topics Excel supports:-"
PRINT "First open a link to the system topic as follows:-"
PRINT ' CALL WIN.DDEOPEN("EXCEL_LINK","excel","System", OK)'
CALL WIN.DDEOPEN("EXCEL_LINK","excel","System", OK)
*
IF NOT(OK) THEN
  PRINT
  PRINT "We couldn't start the link so we shall try to start Excel"
  PRINT 'and try again'
  PRINT ' CALL WIN.PCRUN("EXCEL","")'
  PRINT ' CALL WIN.DDEOPEN("EXCEL_LINK","excel","System", OK)'
  CALL WIN.PCRUN("EXCEL","")
  CALL WIN.DDEOPEN("EXCEL_LINK","excel","System", OK)
  IF NOT(OK) THEN
    PRINT
    CALL WIN.COLOR("lightred","")
    PRINT 'We are unable to start Excel, so the demo will stop.'
```



```
        PRINT "If you do own Excel, try starting it and then re-running  
the demo."    END  
END  
*
```

---

## **Related Script Commands**

Dialog Run Program, Set, Invoke

# WIN.PCRUN2

## Syntax

**WIN.PCRUN2** ( *PATH*, *ARGUMENTS*, *OPTS*, *RESP* )

## Description

This command runs an application or file on the PC. It extends WIN.PCRUN to allow further control over the starting up of the application and to return a status code.

## Parameters

The following table describes the parameters of the WIN.PCRUN2 command:

Parameter	Description
<i>PATH</i>	The path to the program or file name
<i>ARGUMENTS</i>	Additional arguments if <i>PATH</i> is a program
<i>OPTS</i>	Startup options (See below)
<i>RESP</i>	Returns a value specifying whether the file or program ran. If the value is greater than 32, the program was launched correctly. If it was less than or equal to 32, an error occurs. See error code below.

## Fields for OPTS

The OPTS is a dynamic array which contains the following. Setting OPTS to "" gives the default value for all the options

Field	Description
<i>1</i>	Show option. The show option is a request. The applications will not necessarily start with the specified show state. The default is 0. It can be one of the following: 0 - Show and activate, 1 - Minimize, 2 - Maximize, 3 - Hide, 4 - Show but do not activate

Field	Description
2	Verb to run on the application. Either "Open" to open a document or run an application, or "Print" to print a file (Print is not supported by all applications).
3	Default directory (default none) for program to start in

## Values for RESP

The following table shows the error codes returned

Value	Description
31	No association found for file type, or verb not supported.
0	System was out of memory or executable file was corrupt.
2	File was not found.
3	Path was not found.
<i>Others</i>	Executable would not load or run under Windows.

## Example

Run notepad minimized

---

```
CALL WIN.PCRUN2("notepad.exe", "", 1, RESP)
IF RESP < 32 THEN
    PRINT "Failed to run notepad, error code ":RESP
END ELSE PRINT "Notepad started"
```

---

## Related Subroutines

[WIN.PCRUN](#)

## Related Script Commands

Run

## Version

4.0.3 Original version

# WIN.PCSCRIPT

## Syntax

**WIN.PCSCRIPT** ( *SCRIPT*, *ARGS* )

## Description

This subroutine runs a script from the PC. The default file extension is .wis and the directory is assumed to be wintegrate, unless you specify otherwise.

## Parameters

The following table describes the parameters of the WIN.PCSCRIPT command:

Parameter	Description
<i>SCRIPT</i>	The full path of the script. If the script is in the directory, this is null ("")
<i>ARGS</i>	The arguments to pass to the script.

## Example

Run the wc.wis (Command Line dialog box) script

---

```
CALL WIN.PCSCRIPT("example\script\wc.wis", "")
```

---

## Related Script Commands

Chain

# WIN.PCWRITE

## Syntax

**WIN.PCWRITE** ( *NAME*, *BYTES*, *CONVERT* )

## Description

This subroutine writes data to a PC file that was opened with WIN.PCOPEN. There is a maximum of 15K of data that can be written in one call to this routine. The file pointer is moved to the end of the file so that subsequent calls to the routine will write more data to the file.

Note: This routine has no error checking when the data is transferred.

## Parameters

The following table describes the parameters of the WIN.PCWRITE command:

Parameter	Description
<i>NAME</i>	The identifier for the file designated in WIN.PCOPEN.
<i>BYTES</i>	The bytes to write to a file.
<i>CONVERT</i>	The conversion of the data during transfer. See the table below

## Values for CONVERT

This table describes the data conversion options for this routine.

Value	Description
<i>""</i>	No conversion, data reads directly to the variable.
<i>"TEXT"</i>	Converts field marks to carriage returns.
<i>"HEX"</i>	Converts data to 2-byte hex format.
<i>"FT"</i>	Converts data in file transfer format.

## Example

Writes text to a file

---

```
CALL WIN.PCOPEN("new","C:\pctest\pcnew.txt", OK)
IF OK THEN GOSUB 100
100 *open pcnew.txt
NEW.DEF = "This is written by the WIN.PCWRITE subroutine"
CALL WIN.PCWRITE("new", NEW.DEF, "TEXT")
* Close the file
CALL WIN.PCCLOSE("new")
RETURN
END
```

---

## Related Subroutines

[WIN.PCOPEN](#), [WIN.PCCLOSE](#), [WIN.PCREAD](#)

## Related Script Commands

File Write

# WIN.PIESUB

## Syntax

WIN.PIESUB ( *TITLE*, *LABELS*, *REGIONS*, *OPTS* )

## Description

This subroutine draws a pie chart on the screen. The subroutine uses the default options if the parameter "OPTS" is null.

## Parameters

The following table describes the parameters of the WIN.PIESUB command:

Parameter	Description
<i>TITLE</i>	The title you specify for the pie chart.
<i>LABELS</i>	Labels for the regions.
<i>REGIONS</i>	The attribute names for the list of data.
<i>OPTS</i>	(See the Options Table in this section)

## Fields for OPTS

Field	Description
<i>1</i>	Reserved
<i>2</i>	Screen Position
<i>2.1</i>	Left (default is 0)
<i>2.2</i>	Top (default is 0)
<i>2.3</i>	Right (default is 79)
<i>2.4</i>	Bottom (default is 23)
<i>3</i>	Suppress Labels
<i>3.1</i>	Suppresses Values in the legend.
<i>3.2</i>	Suppresses percentages around chart.



Field	Description
4	Colors. Mult-valued list of the colors to use. Leave as "" for the default six colors: RGB_LightRed, RGB_Yellow, RGB_LightGreen, RGB_LightBlue, RGB_LightMagenta and RGB_LightCyan.

## Example

This example is part of the WIN.PIEDEMO demonstration program

---

```
*
CALL WIN.BOX(0,1,79,1,1)
*
LABELS = ''
VALUES = ''
LABELS<1> = 'Computers'; VALUES<1> =200
LABELS<2> = 'Stationery'; VALUES<2> = 500
LABELS<3> = 'Accessories'; VALUES<3> = 50
*
R.OPTS = ''
R.OPTS<2,1> = 15
R.OPTS<2,2> = 3
R.OPTS<2,3> = 65
R.OPTS<2,4> = 17
*
CALL WIN.PIESUB("Products sold", LABELS, VALUES, R.OPTS)
*
PRINT @(0,18):
PRINT "The above graph was created by the subroutine call"
PRINT
PRINT " CALL WIN.PIESUB('Products sold',LABELS, VALUES, R.OPTS)"
PRINT @(0,22):"Press <CR> to continue:"
INPUT DUM:
*
STOP
```

---

## Related Subroutines

[WIN.BARSUB](#)

## Related Script Commands

(See Draw Commands)

# WIN.PLAYBACK

## Syntax

**WIN.PLAYBACK** ( *FILE* )

## Description

This subroutine plays back a recorded file.

## Parameters

The following table describes the parameters of the WIN.PLAYBACK command:

Parameter	Description
<i>FILE</i>	The name of the recorded file. You must specify the full path of the PC file, or the subroutine assumes the file is in the wintegrate directory.

## Example

Record and playback a report

---

```
* Record a report to file "recrept"
CALL WIN.RECON("recrept.txt", 1)
* Demonstrate status bar text
CALL WIN.STATLINE("Selecting data...")
* Run a report
EXECUTE "LIST STUDENT LNAME MAJOR"
CALL WIN.STATLINE("Creating report...")
* Turn off recording
CALL WIN.RECOFF(1)
*
INPUT DUM
CALL WIN.STATLINE("Displaying report...")
* Display the report on the screen
CALL WIN.PLAYBACK("recrept.txt")
CALL WIN.STATLINE(" ")
END
```

---

## Related Subroutines

[WIN.RECOFF](#), [WIN.RECON](#)

## Related Script Commands

Dialog EditPlay, Set, Invoke

# WIN.POPUPIN

## Syntax

**WIN.POPUPIN** ( *MENU*, *X*, *Y*, *RMENU*, *OPT* )

## Description

This subroutine pops up the menu defined in *MENU* and waits for input. The subroutine returns the menu name in *RMENU* and the option in *ROPT*. If no option is selected, then *RMENU* is "null" and *OPT* is "Esc". See the examples below.

## Parameters

The following table describes the parameters of the WIN.POPUPIN command:

Parameter	Description
<i>MENU</i>	The name you specify for the popup menu.
<i>X</i>	The column position of the top left of the menu.
<i>Y</i>	The row position of the top left of the menu.
<i>RMENU</i>	The name of the menu where the option is selected.
<i>OPT</i>	The selected option.

## Example

This example is part of the WIN.MENUDEMO demonstration program.

---

```
*
IF MENU.SHOW = 1 THEN
  CALL WIN.POPUPIN( "Demo" , 45 , 10 , MENU , OPT
END ELSE
  CALL WIN.MENUIN( " " , "Demo" , MENU , OPT )
END
```

---

## Related Subroutines

[WIN.MENULOAD](#), [WIN.MENUDEL](#), [WIN.MENUATT](#), [WIN.MENUDET](#)

## **Related Script Commands**

Menu PopUp

# WIN.PRINTOFF

## Syntax

**WIN.PRINTOFF** ( *SCREEN.ON* )

## Description

This subroutine turns off local (PC) printing that was turned on with WIN.PRINTON. The SCREEN.ON parameter needs to match the SCREEN.OFF parameter in WIN.PRINTON.

## Parameters

The following table describes the parameters of the WIN.PRINTOFF command:

Parameter	Description
<i>SCREEN.ON</i>	0 - Turns the screen off while printing, 1 - Turns the screen on while printing

## Example

Send a host report to the local printer

---

```
CALL WIN.PRINTON(0)
GOSUB 100
100
*Send report to screen and printer
.
.
.
CALL WIN.PRINTOFF(0)
```

---

## Related Subroutines

[WIN.PRINTON](#)

## Related Script Commands

Capture, Screen On

# WIN.PRINTON

## Syntax

**WIN.PRINTON** ( *SCREEN.OFF* )

## Description

This subroutine turns on local (PC) printing. Any text that displays after this command also goes to the printer. When *SCREEN.OFF* is set to 1, the data goes to the printer, and is not displayed on the screen. Turn off printing with the *WIN.PRINTOFF* subroutine.

## Parameters

The following table describes the parameters of the *WIN.PRINTON* command:

Parameter	Description
<i>SCREEN.OFF</i>	0 - Turns the screen on while printing, 1 - Turns the screen off while printing

## Example

Send a report to the local printer

```
CALL WIN.PRINTON(0)
GOSUB 100
100
*Send report to screen and printer
.
.
.
CALL WIN.PRINTOFF(0)
```

## Related Subroutines

[WIN.PRINTOFF](#)

## Related Script Commands

Capture, Screen Off



# WIN.PRTPAUSE

## Syntax

**WIN.PRTPAUSE** ( *PAUSE*, *SCREEN.OFF* )

## Description

This routine pauses/continues the output to a printer from the WIN.PRINTON subroutine to allow the data to be sent to the screen or other subroutines to be called which are not recorded.

## Parameters

The following table describes the parameters of the WIN.PRTPAUSE command:

Parameter	Description
<i>PAUSE</i>	1 pause the printing, 0 restart the printing after the pause
<i>SCREEN.OFF</i>	This should match the SCREEN.OFF parameter of WIN.PRINTON. When set to 1 it turns screen output back on while the printing is paused

## Example

Update the status bar with the printouts progress

---

```
CALL WIN.PRTPAUSE(1,0)
CALL WIN.STATLINE(PC:"% complete")
CALL WIN.PRTPAUSE(0,0)
```

---

## Related Subroutines

[WIN.PRINTON](#)

## Related Script Commands

Capture Pause, Capture Continue

## **Version**

4.1 Original version

# WIN.RECOFF

## Syntax

**WIN.RECOFF** ( *SCREEN.ON* )

## Description

This subroutine stops the recording of raw data from the host to a PC file named in WIN.RECON. The *SCREEN.ON* parameter should match the *SCREEN.OFF* parameter you set in WIN.RECON.

## Parameters

The following table describes the parameters of the WIN.RECOFF command:

Parameter	Description
<i>SCREEN.ON</i>	Turns the screen back on if it was turned off in WIN.RECON. 0 - Keeps the screen turned off. 1 - Turns the screen back on.

## Example

This example demonstrates how to begin recording a file

---

```
* Record a report to file "recrept"
CALL WIN.RECON("recrept.txt", 1)
* Demonstrate status bar text
CALL WIN.STATLINE("Selecting data...")
* Run a report
EXECUTE "LIST STUDENT LNAME MAJOR"
CALL WIN.STATLINE("Creating report...")
* Turn off recording
CALL WIN.RECOFF(1)
*
INPUT DUM
CALL WIN.STATLINE("Displaying report...")
* Display the report on the screen
CALL WIN.PLAYBACK("recrept.txt")
CALL WIN.STATLINE(" ")
END
```

---

## Related Subroutines

[WIN.RECON](#), [WIN.PLAYBACK](#)

## Related Script Commands

Capture, Screen Off

# WIN.RECON

## Syntax

**WIN.RECON** ( *FILE*, *SCREEN.OFF* )

## Description

This subroutine records raw data from the host computer to a PC file. You must specify the full path of the PC file, or the subroutine assumes the file is in the wintegrate directory. Turn the recording off with the WIN.RECOFF subroutine.

## Parameters

The following table describes the parameters of the WIN.RECON command:

Parameter	Description
<i>FILE</i>	The name of the file where the recording is stored.
<i>SCREEN.OFF</i>	Turns the screen off if you want to hide processing. Turn the screen back on with WIN.RECOFF set to 1. 0 - Keeps the screen turned on. 1 - Turns the screen off.

## Example

This example demonstrates how to record a file

---

```
* Record a report to file "recrept"
CALL WIN.RECON("recrept.txt", 1)
* Demonstrate status bar text
CALL WIN.STATLINE("Selecting data...")
* Run a report
EXECUTE "LIST STUDENT LNAME MAJOR"
CALL WIN.STATLINE("Creating report...")
* Turn off recording
CALL WIN.RECOFF(1)
*
INPUT DUM
CALL WIN.STATLINE("Displaying report...")
* Display the report on the screen
CALL WIN.PLAYBACK("recrept.txt")
CALL WIN.STATLINE("")
END
```

---

## Related Subroutines

[WIN.RECOFF](#), [WIN.PLAYBACK](#)

## Related Script Commands

Capture, Screen Off

# WIN.RECPAUSE

## Syntax

**WIN.RECPAUSE** ( *PAUSE*, *SCREEN.OFF* )

## Description

This routine pauses/continues the output to a file from the WIN.RECON subroutine to allow the data to be sent to the screen or other subroutines to be called which are not recorded.

## Parameters

The following table describes the parameters of the WIN.RECPAUSE command:

Parameter	Description
<i>PAUSE</i>	1 pause the recording, 0 restart the recording after the pause
<i>SCREEN.OFF</i>	This should match the SCREEN.OFF parameter of WIN.RECON. When set to 1 it turns screen output back on while the recording is paused

## Example

Update the status bar with the recordings progress

---

```
CALL WIN.RECPAUSE(1,0)
CALL WIN.STATLINE(PC:"% complete")
CALL WIN.RECPAUSE(0,0)
```

---

## Related Subroutines

[WIN.RECON](#)

## Related Script Commands

Capture Pause, Capture Continue

## Version

4.1 Original version



# WIN.RSEXEC

## Syntax

**WIN.RSEXEC** ( *SCRIPT*, *SESSION* )

## Description

This subroutine allows you to define a script, then transfer it to the PC as a single script and run it on a different session from the current session.

Note: A script that is run on the remote session in this way should either complete quickly or run a multi-tasking script command (e.g. Yield, Enter etc).

## Parameters

The following table describes the parameters of the WIN.RSEXEC command:

Parameter	Description
<i>SCRIPT</i>	The text of the script to transfer
<i>SESSION</i>	The name of the session to run the script on

## Example

Putting text in remote status bar

---

```
CALL WIN.RSEXEC("Print 'This text came from a remote session", NAME)
```

---

## Related Subroutines

[WIN.RSEXIST](#), [WIN.RSSCRIPT](#), [WIN.RSNAME](#), [WIN.RSSTART](#)

## Related Script Commands

Session Execute

## Version

4.2.1 Original

# WIN.RSEXIST

## Syntax

**WIN.RSEXIST** ( *SESSION*, *RESP* )

## Description

This subroutine checks if the named session is currently running.

## Parameters

The following table describes the parameters of the WIN.RSEXIST command:

Parameter	Description
<i>SESSION</i>	The session name to check
<i>RESP</i>	Set to 1 if the session exists, 0 if it does not.

## Example

Check if a session called session2 is running

---

```
CALL WIN.RSEXIST("Session2", RESP)
IF RESP = 1 THEN PRINT "Session 2 is running"
```

---

## Related Subroutines

[WIN.RSEXEC](#), [WIN.RSSCRIPT](#), [WIN.RSNAME](#), [WIN.RSSTART](#)

## Related Script Commands

Sessions

## Version

4.2.1

# WIN.RSNAME

## Syntax

WIN.RSNAME ( *NAME* )

## Description

This subroutine returns the name of the session it is run upon.

## Parameters

The following table describes the parameters of the WIN.RSNAME command:

Parameter	Description
<i>NAME</i>	The session name returned

## Example

Check if this session is session2

---

```
CALL WIN.RSNAME(NAME)
IF NAME = "Session2" THEN
    PRINT "This program should not be run on Session2"
    STOP
END
```

---

## Related Subroutines

[WIN.RSEXEC](#), [WIN.RSEXIST](#), [WIN.RSSCRIPT](#), [WIN.RSSTART](#)

## Related Script Commands

Get(Name)

## Version

4.2.1 Original

# WIN.RSSCRIPT

## Syntax

**WIN.RSSCRIPT** ( *FILENAME*, *ARGUMENT*, *SESSION* )

## Description

This subroutine allows you to run a script that exists on the client PC in a session other than the one that runs this command.

Note: A script that is run on the remote session in this way should either complete quickly or run a multi-tasking script command (e.g. Yield, Enter etc).

Tip: To find/modify the session name for a session use the Setup Preferences Options page

## Parameters

The following table describes the parameters of the WIN.RSSCRIPT command:

Parameter	Description
<i>FILENAME</i>	The PC filename of the script to run
<i>ARGUMENT</i>	The argument for the script. The remote session receives this in the CommandLine global variable
<i>SESSION</i>	The name of the session to run the script on.

## Example

Run RsTest on another session

---

```
* Run RsTest script with the parameter "hello" on session2
PRINT "Runing RsTest.wis"
CALL WIN.RSSCRIPT("RsTest","hello", "session2")
```

---

## Related Subroutines

[WIN.RSEXEC](#), [WIN.RSEXIST](#), [WIN.RSNAME](#), [WIN.RSSTART](#)

## **Related Script Commands**

Session Script

## **Version**

4.2.1 Original

# WIN.RSSTART

## Syntax

**WIN.RSSTART** ( *FILENAME* )

## Description

This subroutine starts the named session file. This is identical to doing File Another from the menu. Due to the multi-tasking nature of Windows the session is unlikely to have finished starting up when the routine returns.

## Parameters

The following table describes the parameters of the WIN.RSSTART command:

Parameter	Description
<i>FILENAME</i>	The file name of the session file to start

## Example

Start up another session

---

\* Start up session file SFILE that has session name NAME.

```
PRINT "Starting remote session"
CALL WIN.RSSTART(SFILE)
*
* Give the session 10 seconds to start up
J = 0
LOOP
  CALL WIN.RSEXIST(NAME, RESP)
  J = J + 1
UNTIL J = 10 OR RESP = 1
  SLEEP 1 ;* RQM on some host systems
REPEAT
*
IF RESP = 0 THEN
  PRINT "Session isn't running. Can't do tests!"
  STOP
END
```

---

## Related Subroutines

[WIN.RSEXEC](#), [WIN.RSEXIST](#), [WIN.RSSCRIPT](#), [WIN.RSNAME](#)

## Version

4.2.1



# WIN.SCREEN

## Syntax

**WIN.SCREEN** ( *TURN.ON* )

## Description

This subroutine turns screen output on or off. You may want to use this command when you want to hide system messages, or just want to keep the screen blank while processing a call.

## Parameters

The following table describes the parameters of the WIN.SCREEN command:

Parameter	Description
<i>TURN.ON</i>	0 or OFF - Turns off screen output. 1 or ON - Turns on screen output.

## Example

Turn off the display while executing SSELECT

---

```
CALL WIN.SCREEN(0) ;* Turn screen off
EXECUTE "SSELECT COURSES"
CALL WIN.SCREEN(1) ;* Turn screen on
```

---

## Related Subroutines

[WIN.DISPLAY](#)

## Related Script Commands

Screen On, Screen Off

# WIN.SDUMP

## Syntax

**WIN.SDUMP** ( *DEVICE*, *FORMAT* )

## Description

This subroutine copies the current screen, then stores it in a file or sends it to the printer. To store the screen as a file, the file name must have a text (.txt) or bitmap (.bmp) extension. Specify the full path of the storage file, or wIntegrate assumes it to be in the wintegrate directory.

## Parameters

The following table describes the parameters of the WIN.SDUMP command:

Parameter	Description
<i>DEVICE</i>	The device is "Printer" or the PC file name. The default is "Printer".
<i>FORMAT</i>	Specifies how to save the file, as "TEXT" or "BITMAP"

## Example

Run a report on students

---

```
EXECUTE "SORT STUDENT LNAME FNAME BY LNAME"  
* Copy the current on-screen report to PC file  
CALL WIN.SDUMP("C:\temp\report.txt", "TEXT")  
END
```

---

## Related Script Commands

Dialog EditCopyTo, Set, Invoke, Invoke EditSelectScreen, Invoke EditCopySpecial

# WIN.SENDKEYS

## Syntax

**WIN.SENDKEYS** ( *KEYS*, *WAIT* )

## Description

This subroutine sends keystrokes to an application. Use this with applications that do not support DDE. The keystrokes will be sent to the active window, so use WIN.ACTIVATE to change the focus to the application window.

## Parameters

The following table describes the parameters of the WIN.SENDKEYS command:

Parameter	Description
<i>KEYS</i>	The keystrokes that you want to send to the other application. See the table below for special key combinations.
<i>WAIT</i>	WAIT returns the following: 0 = Return immediately 1 = Wait until the keystrokes have been sent.

## Values for KEYS

This table describes how to use a character to designate special keys like shift, control, etc. Use the character in front of the key it modifies.

Value	Description
+	Designates the Shift key.
^	Designates the CTRL key.
%	Designates the ALT key.
+^%	Designates the use of all three of the above together.
{ <i>char num</i> }	Designates the use of a key more than once.

## Example

Put a report name on line one of a text document using notepad.

---

This example prints out a message if the task is not running. If the task is running, WIN.ACTIVATE changes the focus to the Notepad document and prints the report name on the first line.

```
* Verify the report "recrept.txt" is running
CALL WIN.TASK("Notepad - RECREPT.TXT", RUNNING)
IF RUNNING THEN GOSUB 100 ELSE
PRINT "recrept.txt is not running, open it and try again"
RETURN
100
* Change the active window from wIntegrate to Notepad
CALL WIN.ACTIVATE("Notepad - RECREPT.TXT")
* Send the report name to the Notepad report
CALL WIN.SENDKEYS("STUDENT REPORT", 1)
RETURN
END
```

---

## Related Subroutines

[WIN.ACTIVATE](#)

## Related Script Commands

[SendKeys](#)

# WIN.SERIAL

## Syntax

WIN.SERIAL ( *SERIAL* )

## Description

This subroutine returns the wIntegrate serial number.

## Parameters

The following table describes the parameters of the WIN.SERIAL command:

Parameter	Description
<i>SERIAL</i>	Returns the wIntegrate serial number.

## Example

Return the wIntegrate serial number

---

```
CALL WIN.SERIAL(SERIAL.NO)
PRINT "Your wIntegrate serial number is ":SERIAL.NO
```

---

## Related Subroutines

[WIN.VERSION](#)

## Related Script Commands

Serial (built-in constant)

# WIN.SERVER

## Syntax

WIN.SERVER

## Description

This subroutine is the host side of the PC server program. You can trap this command and execute it at your inputs to allow server actions to run while not at a database prompt.

The server script library calls this subroutine. For more information, see the Client Scripting Reference manual.

## Parameters

None

# WIN.SETDATA

## Syntax

**WIN.SETDATA** ( *VARNAME*, *VALUE* )

## Description

This subroutine sets the value or contents of a global or dialog variable. The variable can contain ASCII characters less than 32 and greater than 126, including tabs and carriage returns. If your variables contain only ASCII characters between 32 and 126, use the faster WIN.SETVAR subroutine.

## Parameters

The following table describes the parameters of the WIN.SETDATA command:

Parameter	Description
<i>VARNAME</i>	The name of the variable.
<i>VALUE</i>	The value of the variable specified in <i>VARNAME</i> .

## Example

Set and print the current value of UserName

---

```
CALL WIN.SETDATA("UserName", VALUE)
CALL WIN.HSCRIPT("Print UserName")
```

---

## Related Subroutines

[WIN.GETVAR](#), [WIN.SETLIST](#), [WIN.SETVAR](#), [WIN.DBGET](#), [WIN.DBSET](#)

# WIN.SETEFFECT

## Syntax

WIN.SETEFFECT ( *NAME*, *FORE.COL*, *BACK.COL*, *STYLE*, *JOIN* )

## Description

This subroutine sets the color and style for a terminal effect. This works the same as Setup Colors, when you change the foreground and background colors and the style for a certain effect.

## Parameters

The following table describes the parameters of the WIN.SETEFFECT command:

Parameter	Description
<i>NAME</i>	The name of the effect. "Normal", "Dim", "Reverse", "Underline", "Flash", "Bold" or "Secret". Effects can be combined by concatenating their names e.g. DimReverse.
<i>FORE.COL</i>	The foreground or text color. See WIN.COLOR for available colors.
<i>BACK.COL</i>	The background color. See WIN.COLOR for the colors.
<i>STYLE</i>	The style to use. See Styles in this section.
<i>JOIN</i>	Joins lines on drawn boxes. This is like turning on Join Lines box in Setup Colors.

## Values for STYLE

Value	Description
<i>None</i>	No border for the effect.
<i>Raised</i>	A raised border that makes text appear above the background.
<i>Inset</i>	An inset border.



Value	Description
<i>TextBox</i>	A defined inset border around the text.
<i>Border</i>	A plain black border around the text.
<i>Raised2</i>	A heavy raised border around the text.
<i>Inset2</i>	A heavy inset border around the text.

## Example

Changes the Bold effect to Magenta text on a blue background with a defined raised border

---

```
CALL WIN.SETEFFECT("Bold", "Magenta", "Blue", "Raised2", "")
```

---

## Related Script Commands

Set (Effect)

# WIN.SETLIST

## Syntax

**WIN.SETLIST** ( *VARNAME*, *VALUE* )

## Description

Since PC items generally have lines separated by carriage returns, and values are separated by tabs, this routine is provided to convert these characters to field marks (char 254) and value marks (char 253).

This subroutine sets the value or contents of a global or local variable. The variable can contain ASCII characters less than 32 and greater than 126, including tabs and carriage returns. If your variables contain only ASCII characters between 32 and 126, use the faster WIN.SETVAR subroutine.

## Parameters

The following table describes the parameters of the WIN.SETLIST command:

Parameter	Description
<i>VARNAME</i>	The name of the variable to return.
<i>VALUE</i>	The value of the variable specified in <i>VARNAME</i> .

## Example

Set the value of the PC item PCREC, convert from the host item format

---

```
READ HOSTREC FROM FILE, ID THEN
CALL WIN.SETLIST( "PCREC", HOSTREC )
END
```

---

## Related Subroutines

[WIN.GETDATA](#), [WIN.GETVAR](#), [WIN.SETDATA](#), [WIN.DBGET](#), [WIN.DBSET](#),  
[WIN.SETVAR](#)

# WIN.SETPARAM

## Syntax

**WIN.SETPARAM** ( *NAME*, *VALUE* )

## Description

This subroutine sets a session variable.

See the "Reference - Menu Options" chapter of the "Client Scripting Reference" manual for a full list of the session variables.

## Parameters

The following table describes the parameters of the WIN.SETPARAM command:

Parameter	Description
<i>NAME</i>	The name of the parameter or variable to set.
<i>VALUE</i>	The value of the parameter specified in NAME.

## Example

This example sets the wIntegrate Title.

---

```
CALL WIN.SETPARAM("Title", "Title changed by WIN.SETPARAM")
```

---

## Related Subroutines

[WIN.GETPARAM](#)

## Related Script Commands

Set

# WIN.SETVAL

## Syntax

**WIN.SETVAL** ( *VALUE*, *OPTS* )

## Description

This subroutine sends a value from the host to the Host Get script command. A Host Get command must be run before this subroutine is called.

## Parameters

The following table describes the parameters of the WIN.SETVAL command:

Parameter	Description
<i>VALUE</i>	The value to send to the PC
<i>OPTS</i>	Reserved. Must be set to ""

## Example

Send a brief message to the user

---

```
CALL WIN.COMSUB("Host Get msg;MessageBox msg")
CALL WIN.SETVAL("Please logoff the system","")
```

---

## Related Script Commands

Host Get

## Version

4.1.0 Original

# WIN.SETVAR

## Syntax

**WIN.SETVAR** ( *VARNAME*, *VALUE* )

## Description

This routine sets the contents of a global variable. It is like WIN.SETDATA, but it can be used only if the variable has a simple value. This makes the transfer faster. No characters with ASCII values less than 32 or more than 126. This means there can be no carriage returns, tabs, or eight-bit characters.

## Parameters

The following table describes the parameters of the WIN.SETVAR command:

Parameter	Description
<i>VARNAME</i>	The name of the variable.
<i>VALUE</i>	The value of the variable specified in VARNAME.

## Example

Set and print the current value of UserName

---

```
CALL WIN.SETVAR("UserName", VALUE)
CALL WIN.HSCRIPT("Print UserName")
```

---

## Related Subroutines

[WIN.GETVAR](#), [WIN.SETDATA](#), [WIN.DBGET](#), [WIN.DBSET](#), [WIN.SETLIST](#)

# WIN.SHARE

## Syntax

WIN.SHARE

## Description

This program enables another user account to use the host programs. It should be executed from the account where the host programs were originally installed.

It sets up the necessary file pointers and catalogs the host programs.

## Parameters

None

# WIN.SHOW

## Syntax

**WIN.SHOW** ( *MENU.DLG* )

## Description

This subroutine displays the dialog box that is normally called from a menu command. You may want to open wIntegrate boxes like FileOpen, FileEdit, etc., from a script.

See the "Reference - Script Menu Options" chapter of the "Client Scripting Reference" manual for a full list of the available menu options.

## Parameters

The following table describes the parameters of the WIN.SHOW command:

Parameter	Description
<i>MENU.DLG</i>	The name of the menu command that calls the dialog

## Example

Open the Import File dialog box

---

```
CALL WIN.SHOW("RunImportFile")
```

---

## Related Subroutines

[WIN.INVOKE](#)

## Related Script Commands

Show

# WIN.SLOAD

## Syntax

WIN.SLOAD ( *FILE* )

## Description

This subroutine loads or restores a screen previously saved with WIN.SSAVE.

## Parameters

The following table describes the parameters of the WIN.SLOAD command:

Parameter	Description
<i>FILE</i>	The PC file that the screen was saved to with the WIN.SSAVE subroutine.

## Example

The following example saves the current screen named white1 (white background, black text).

---

```
* Then the program executes a sort statement to a new screen (blue
background, magenta text).
* When the user enters a carriage return after the report, the screen
returns to white 1.
* Save the current screen
CALL WIN.SSAVE("white1")
CALL WIN.COLOR("Magenta", "Blue")
EXECUTE "SORT CUSTOMER BY NAME CITY"
INPUT DUM
* Restore the screen after executing the sort
CALL WIN.SLOAD("white1")
```

---

## Related Subroutines

[WIN.SSAVE](#), [WIN.SSTORE](#), [WIN.SRESTORE](#)



## **Related Script Commands**

Screen Load, Screen Save

# WIN.SPOOL

## Syntax

WIN.SPOOL ( *PCFILE*, *WIDTHS*, *OPTS*, *STATUS* )

## Description

This subroutine prints a PC file to the host spooler, as in Run Spool File. When the file is transferred to the host for printing, the Spool File Monitor appears until the transfer is complete.

## Parameters

The following table describes the parameters of the WIN.SPOOL command:

Parameter	Description
<i>PCFILE</i>	The name and full path of the PC file.
<i>WIDTHS</i>	Specifies the width of one or more columns. If this null, (""), the entire line prints.
<i>OPTS</i>	A multifield dynamic array that allows modification other file transfer options. See the Options table below
<i>STATUS</i>	(This parameter is not implemented.)

## Fields for OPTS

Field	Description
1	File Transfer Format. Defaults to Extension of PCFILE.
2	Separator character between columns. For instance, a vertical line between columns improves readability. Default " " (one space)
3	Multiple Disks - The file is on more than one floppy disk. Default 0.

Field	Description
4	Data Per Line - Amount of data on each line. Default "OneRecord"
5	(Reserved)
6	Translate - Use the Translation table. Default "Ascii"
7	Translation Table. Default "\255,\r\n\f\r\n,\254,\r\n"
8	AutoExit - Exit the monitor at end. Default True
9	Inform - Notify when finished. Default False
10	Timeout - Timeout in seconds. Default 5
11	Retries - Maximum number of error retries. Default 3

## Example

Print text document on host printer

---

```
STATUS = ""
OPTS = ""
OPTS<2> = "| "
OPTS<9> = "True"
CALL WIN.SPOOL("C:\winword\wbond.txt", "", OPTS, STATUS)
```

---

## Related Subroutines

[WIN.EXPORT](#), [WIN.IMPORT](#), [WIN.TRANSFER](#)

## Related Script Commands

Dialog RunSpoolFile, Set, Invoke

# WIN.SPULL

## Syntax

WIN.SPULL

## Description

This subroutine restores a screen that was saved with WIN.SPUSH. Use this when you want to restore the screen while running commands, reports, etc.

## Parameters

None

## Example

The following example saves the current screen executes the SORT statement, then restores the screen to show the SORT results. The SORT statement is not shown at the database prompt.

---

```
* Save the current screen
CALL WIN.SPUSH
EXECUTE "SORT CUSTOMER BY NAME CITY"
* Restore the screen after executing the sort
CALL WIN.SPULL
```

---

## Related Subroutines

[WIN.SPUSH](#)

## Related Script Commands

Screen Restore, Screen Remove

# WIN.SPUSH

## Syntax

WIN.SPUSH

## Description

This subroutine stores the current screen in memory. Restore the screen with WIN.SPULL.

## Parameters

None

## Example

The following example saves the current screen, executes the SORT statement, then restores the screen to show the SORT results. The SORT statement is not shown at the database prompt.

---

```
* Save the current screen
CALL WIN.SPUSH
EXECUTE "SORT CUSTOMER BY NAME CITY"
* Restore the screen after executing the sort
CALL WIN.SPULL
```

---

## Related Subroutines

[WIN.SPULL](#)

## Related Script Commands

Screen Restore, Screen Remove

# WIN.SREMOVE

## Syntax

WIN.SREMOVE ( *NAME* )

## Description

This subroutine deletes the screen stored in PC memory by the WIN.SSTORE subroutine.

## Parameters

The following table describes the parameters of the WIN.SREMOVE command:

Parameter	Description
<i>NAME</i>	The name of the host file where the screen is stored.

## Example

This example is taken from the WIN.DEMO demonstration program.

---

```
*
PRINT @(5,20):"Enter M to see how the mouse can be programmed to
select
options":
PRINT @(0,22):"Enter your choice or * to quit:"
CALL WIN.SSTORE("demo");* Store current screen in memory
.
.
.
*

CALL WIN.SREMOVE("demo") ;* Delete saved screen
CALL WIN.COLOR("off","") ;* Turn off colouring of text
```

---

## Related Subroutines

[WIN.SSTORE](#), [WIN.SSAVE](#), [WIN.SLOAD](#)

## **Related Script Commands**

Screen Remove

# WIN.SRESTORE

## Syntax

WIN.SRESTORE ( *NAME* )

## Description

This subroutine restores a screen saved in PC memory by WIN.SSTORE.

## Parameters

The following table describes the parameters of the WIN.SRESTORE command:

Parameter	Description
<i>NAME</i>	The name of the screen stored in memory with WIN.SSTORE

## Example

This example is taken from the WIN.DEMO demonstration program.

---

```
* Run a demonstration
1000 CALL WIN.COLOR("off",""); * Turn off text colouring
CALL @EXECSUB(PROG, '',0); * Execute/Perform program
PRINT @(-1):
CALL WIN.SRESTORE("demo");* Restore menu screen
RETURN
```

---

## Related Subroutines

[WIN.SSTORE](#), [WIN.SREMOVE](#), [WIN.SSAVE](#)

## Related Script Commands

Screen Restore



# WIN.SSAVE

## Syntax

WIN.SSAVE ( *FILE* )

## Description

This subroutine saves the current screen. It can be restored with WIN.SLOAD.

## Parameters

The following table describes the parameters of the WIN.SSAVE command:

Parameter	Description
<i>FILE</i>	The name of the PC file that will store the screen.

## Example

The following example saves the current screen named white1 (white background, black text).

---

```
* Then the program executes a sort statement to a new screen (blue
background, magenta text).
* When the user enters a carriage return after the report, the screen
returns to white 1.
* Save the current screen
CALL WIN.SSAVE("white1")
CALL WIN.COLOR("Magenta", "Blue")
EXECUTE "SORT CUSTOMER BY NAME CITY"
INPUT DUM
* Restore the screen after executing the sort
CALL WIN.SLOAD("white1")
```

---

## Related Subroutines

[WIN.SLOAD](#), [WIN.SSTORE](#), [WIN.SRESTORE](#)

## Related Script Commands

Screen Save

# WIN.SSTATE

## Syntax

**WIN.SSTATE** ( *OLD.STATE*, *NEW.STATE* )

## Description

This subroutine saves the current screen state (cursor position, current effects, etc.), and restores it.

## Parameters

The following table describes the parameters of the WIN.SSTATE command:

Parameter	Description
<i>OLD.STATE</i>	The state of the screen before the change.
<i>NEW.STATE</i>	The change to the screen. Null if there is no change.

## Example

Save the screen during a program

---

```
*
CALL WIN.SSTATE(OLD.STATE,"")
CALL WIN.COLOR("White","Blue")
.
.
.
*
GOSUB 120; * Restore effect we changed
CALL WIN.TWCLOSE("INFO")
CALL WIN.SSTATE(" ",OLD.STATE)
*
STOP
```

---

## Related Subroutines

[WIN.SSAVE](#), [WIN.SSTORE](#), [WIN.SREMOVE](#)

## **Related Script Commands**

Cursor(6)

# WIN.SSTORE

## Syntax

WIN.SSTORE ( *NAME* )

## Description

This subroutine stores the current screen in memory. The saved screen is restored with WIN.SRESTORE. WIN.SSTORE stores the screen in the PC memory, and WIN.SSAVE saves the screen to a PC file.

## Parameters

The following table describes the parameters of the WIN.SSTORE command:

Parameter	Description
<i>NAME</i>	The name you give to the stored screen.

## Example

This example is taken from the WIN.DEMO demonstration program.

---

```
(The ellipses indicate other lines of code.)
*
PRINT @(5,20):"Enter M to see how the mouse can be programmed to
select
options":
PRINT @(0,22):"Enter your choice or * to quit:"
CALL WIN.SSTORE("demo");* Store current screen in memory
.
.
.
*
CALL WIN.SREMOVE("demo") ;* Delete saved screen
CALL WIN.COLOR("off","") ;* Turn off colouring of text
```

---

## Related Subroutines

[WIN.SRESTORE](#), [WIN.SSAVE](#), [WIN.SREMOVE](#)

## **Related Script Commands**

Screen Store

# WIN.STACK

## Syntax

**WIN.STACK** ( *TURN.ON* )

## Description

This subroutine turns the database line stacker on and off. All lines typed into the computer are recorded after WIN.STACK is turned on. Use this routine in a program where you have previously called WIN.TCL with the ".E", to display the TCL Command Stack dialog box. You may want to use this routine if you want to use the stack of commands on another host.

Note: Another way to access the TCL Command Stack dialog box to edit lines, run the PC script wintsys\ script\TCLStack.wis.

## Parameters

The following table describes the parameters of the WIN.STACK command:

Parameter	Description
<i>TURN.ON</i>	0 or "OFF" - Turns off the line stacker. 1 or "ON" - Turns on the line stacker.

## Related Subroutines

[WIN.STACKOFF](#), [WIN.STACKON](#)

# WIN.STACKOFF

## Syntax

WIN.STACKOFF

## Description

Run this program at the ECL prompt to turn off the capture of ECL commands for the TCL command stack.

These routines are used in conjunction with the TCL Command Stack; a small dialog box that contains a pull-down listing of the most current commands that were issued from the database prompt.

See the WIN.TCL subroutine with the ".E" option.

## Parameters

None

## Related Subroutines

[WIN.STACKON](#), [WIN.STACK](#), [WIN.TCL](#)

# WIN.STACKON

## Syntax

WIN.STACKON

## Description

Run this program at the ECL prompt to turn on the capture of ECL commands for the TCL command stack.

These routines are used in conjunction with the TCL Command Stack; a small dialog box that contains a pull-down listing of the most current commands that were issued from the database prompt.

See the WIN.TCL subroutine with the ".E" option.

## Parameters

None

## Related Subroutines

[WIN.STACKOFF](#), [WIN.STACK](#), [WIN.TCL](#)



# WIN.STATLINE

## Syntax

**WIN.STATLINE** ( *TEXT* )

## Description

This subroutine displays a line of text on the wIntegrate Status Bar.

## Parameters

The following table describes the parameters of the WIN.STATLINE command:

Parameter	Description
<i>TEXT</i>	The text that you want to display on the status bar.

## Example

This example informs the user what is happening while the report is generated and recorded to a file

---

```
* Record a report to file "recrept.txt"
CALL WIN.RECON("recrept.txt", 1)
* Demonstrate status bar text
CALL WIN.STATLINE("Selecting data...")
* Run a report
EXECUTE "LIST STUDENT LNAME MAJOR"
CALL WIN.STATLINE("Creating report...")
* Turn off recording
CALL WIN.RECOFF(1)
*
INPUT DUM
CALL WIN.STATLINE("Displaying report...")
* Display the report on the screen
CALL WIN.PLAYBACK("recrept.txt")
CALL WIN.STATLINE(" ")
END
```

---

## Related Script Commands

Print

# WIN.TASK

## Syntax

**WIN.TASK** ( *TASKNAME*, *RUNNING* )

## Description

This subroutine checks to see if a Windows task is running. The task name must be exactly as it displays in the Windows task list (usually the title of the main application window).

## Parameters

The following table describes the parameters of the WIN.TASK command:

Parameter	Description
<i>TASKNAME</i>	The task name as it displays in the Windows task list
<i>RUNNING</i>	Returns the following: 0 = task is not running 1 = task is running

## Example

This example prints out a message if the task is not running.

---

```
* If the task is running, WIN.ACTIVATE changes the focus to the
Notepad document and prints the report name on the first line.
* Verify the report "recept.txt" is running
CALL WIN.TASK("Notepad - RECREPT.TXT", RUNNING)
IF RUNNING THEN GOSUB 100 ELSE
PRINT "recept.txt is not running, open it and try again"
RETURN
100 * Change the active window from wIntegrate to Notepad
CALL WIN.ACTIVATE("Notepad - RECREPT.TXT")
* Send the report name to the Notepad report
CALL WIN.SENDKEYS("STUDENT REPORT", 1)
RETURN
END
```

---

## **Related Subroutines**

[WIN.APP](#)

## **Related Script Commands**

IsTask

# WIN.TCL

## Syntax

WIN.TCL

## Description

This subroutine saves the current screen and enters the database or ECL (TCL) interpreter. You may enter WIN.TCL at the database prompt. Enter a "?" to display the full list of available commands.

The commands available are:

.E Turns on the ECL stacker. See example in this section.

.W Starts the wIntegrate command line as in the WIN.COMLINE subroutine.

.A Starts the Query Builder Dialog Box. See the Using wIntegrate manual for how to use Query Builder.

EX Quits the TCL interpreter and restores the screen.

Q Quits the TCL interpreter and restores the screen.

? Lists the available commands.

## Parameters

None

## Related Subroutines

[WIN.STACK](#), [WIN.STACKON](#), [WIN.STACKOFF](#), [WIN.TCL](#)

# WIN.TITLE

## Syntax

**WIN.TITLE** ( *TITLE* )

## Description

This subroutine sets the title for a wIntegrate session. The title appears in the title bar of the wIntegrate window.

## Parameters

The following table describes the parameters of the WIN.TITLE command:

Parameter	Description
<i>TITLE</i>	The new name of the wIntegrate session.

## Example

Put the wIntegrate version in the title

---

```
* Return the wIntegrate version number
CALL WIN.VERSION(VERSION)
PRINT "You are running wIntegrate version ": VERSION
* Change the Title to "wIntegrate x.x"
CALL WIN.TITLE("wIntegrate ":VERSION)
END
```

---

## Related Script Commands

Set Title

# WIN.TRANSFER

## Syntax

WIN.TRANSFER

## Description

This program is the host side of the file transfer routines (Run Import File, Run Export File, Run Spool File and Run Bridge Copy).

If file transfers are executed from within a host program, WIN.TRANSFER must be executed to enable the host side of the transfer to operate.

## Parameters

None

## Example

Simple host menu that allows file transfers

---

```
* At a host menu input the user can enter options to run host
programs.
* Also allow for files transfer initiated from the PC to start the
host
* side of the file transfer, removing the need for the user to
* run file transfers at the command prompt (ECL)
INPUT CMND:    ;* from keyboard or wIntegrate program
BEGIN CASE
CASE CMND = 1  ;* User keyboard input
    EXECUTE "CUST.MAINT"
CASE CMND = 2  ;* User keyboard input
    EXECUTE "PROD.MAINT"
CASE CMND = "WIN.TRANSFER"; * From Run File Import/Export etc
    EXECUTE "WIN.TRANSFER"
CASE CMND = "WIN.SERVER"; * From Query Builder etc.
    EXECUTE "WIN.SERVER"
END CASE
```

---

Run a file transfer from a script on the host

---

```
READ TRANSFER.SCRIPT FROM SCRIPT.FILE, "CUST.IMPORT" THEN
    CALL WIN.HSCRIPT(TRANSFER.SCRIPT)
    EXECUTE "WIN.TRANSFER"
END
```

---

## **Related Subroutines**

[WIN.IMPORT](#), [WIN.EXPORT](#), [WIN.SPOOL](#)

# WIN.TWCLOSE

## Syntax

WIN.TWCLOSE ( *NAME* )

## Description

This subroutine closes a text window opened with WIN.TWOPEN.

## Parameters

The following table describes the parameters of the WIN.TWCLOSE command:

Parameter	Description
<i>NAME</i>	The name of the text window specified in WIN.TWOPEN.

## Example

This example is part of the WIN.TWDEMO demonstration program.

---

```
*
PRINT @(0,22) : 'Press <CR> to create DEMO1 window ' :
INPUT DUM:
CALL WIN.COLOR("Yellow","blue")
CALL WIN.TWOPEN("DEMO1","Text window DEMO1",10,13,75,16,2)
PRINT "This window was created with:-"
PRINT 'CALL WIN.TWOPEN("DEMO1","Text window DEMO1",10,13,75,16,2)'
PRINT
PRINT 'Press <CR> to create DEMO2 window ' :
INPUT DUM:
CALL WIN.TWOPEN("DEMO2","Text window DEMO2", 34,2,72,6,1)
PRINT 'We now have two windows opened using WIN.TWOPEN, DEMO1 & DEMO2, ' :
PRINT 'which we can switch between with WIN.TWUSE'
PRINT
PRINT 'Press <CR> to go to the DEMO1 window ' :
INPUT DUM:
CALL WIN.TWUSE("DEMO1")
PRINT @(-1):'We are now back in our DEMO1 window. This was achieved by the call'
PRINT ' CALL WIN.TWUSE("DEMO1")'
PRINT 'Press <CR> to close both windows and continue ' :
INPUT DUM:
```



```
CALL WIN.TWCLOSE ( "DEMO1" )  
CALL WIN.TWCLOSE ( "DEMO2" )
```

---

## **Related Subroutines**

[WIN.TWOPEN](#), [WIN.TWFOOT](#), [WIN.TWPULL](#), [WIN.TWPUSH](#), [WIN.TWMSG](#)

## **Related Script Commands**

Screen Restore, Screen ScrollRegion

# WIN.TWFOOT

## Syntax

**WIN.TWFOOT** ( *NAME*, *TEXT*, *POS* )

## Description

This subroutine adds a footer to a text box created with the WIN.TWOPEN subroutine.

## Parameters

The following table describes the parameters of the WIN.TWFOOT command:

Parameter	Description
<i>NAME</i>	The name of the textbox created with WIN.TWOPEN.
<i>TEXT</i>	The text of the footer.
<i>POS</i>	Specifies the position of the text on the line. L = Left-justified R = Right-justified C = Centered on the line

## Example

This example is taken from the code of the WIN.CEDEMO demonstration program.

---

```
* Set up box to display information
CALL WIN.COLOR("White","Blue")
CALL WIN.TWOPEN("INFO","Information",1,18,78,22,"DOUBLE")
CALL WIN.COLOR("Black","White")
CALL WIN.TWFOOT("INFO","Press <CR> to continue","R")
CALL WIN.COLOR(TEXT.COL,"Blue")
```

---

## Related Subroutines

[WIN.TWOPEN](#), [WIN.TWMSG](#), [WIN.TWPULL](#), [WIN.TWPUSH](#), [WIN.TWCLOSE](#)

# WIN.TWMSG

## Syntax

WIN.TWMSG ( *TEXT*, *FGCOL*, *BGCOL*, *BORDER* )

## Description

This subroutine opens a text message window called "MSG", which is centered on the screen. Close this window with a call to WIN.TWCLOSE("MSG").

## Parameters

The following table describes the parameters of the WIN.TWMSG command:

Parameter	Description
<i>TEXT</i>	The text to display in the message window.
<i>FGCOL</i>	Specifies the foreground or text color. (See WIN.COLOR a list of colors.)
<i>BGCOL</i>	Specifies the background color. (See WIN.COLOR a list of colors.)
<i>BORDER</i>	Specifies the style for the window. SPACES = blank border SINGLE = single-line border DOUBLE = double-line border MIX = alternate dots and block BLOCK = solid block

## Example

This example is a part of the WIN.DBDEMO demonstration program.

---

```
* Create and show dialog
100 DBX=' '
DLG.NAME="DemoDlg"
CALL WIN.TWMSG("Creating Dialog ":DLG.NAME:"...", "Red", "Yellow",
"DOUBLE")
```

---

## Related Subroutines

[WIN.TWOPEN](#), [WIN.TWCLOSE](#)

# WIN.TWOPEN

## Syntax

**WIN.TWOPEN** ( *NAME*, *TITLE*, *LEFT*, *TOP*, *RIGHT*, *BOTTOM*, *STYLE* )

## Description

This subroutine opens a text window on the wIntegrate screen. A text window scrolls independently of the main wIntegrate window. You may have several of these boxes active at the same time, which means that each of them requires a unique name.

If you want to display more than one text window at a time, you can show them in different parts of the screen using WIN.TWUSE. If you want one window on top of another, use WIN.TWPUSH to "push" a window on top of an already active one, and WIN.TWPULL to "pull" the various text windows off the main screen.

## Parameters

The following table describes the parameters of the WIN.TWOPEN command:

Parameter	Description
<i>NAME</i>	The name or identifier of the text window. This is how the program will identify this text window for other subroutines.
<i>TITLE</i>	The title of the text window. This will display at the top of the text window. To add a footer to the window, see WIN.TWFOOT.
<i>LEFT</i>	Specifies the coordinate for the left side of the box.
<i>TOP</i>	Specifies the coordinate for the top of the box.
<i>RIGHT</i>	Specifies the coordinate for the right side of the box.
<i>BOTTOM</i>	Specifies the coordinate for the bottom of the box.
<i>STYLE</i>	Specifies the style for the box. See the table below

## Values for STYLE

Value	Description
"SPACES"	blank border
"SINGLE"	single line border
"DOUBLE"	double-line border
"MIX"	alternate dot and block
"BLOCK"	solid block

## Example

This example is a part of the WIN.TWDEMO demonstration program. It show how to open two different text windows.

---

```
*
PRINT @(0,22) : 'Press <CR> to create DEMO1 window ':
INPUT DUM:
CALL WIN.COLOR("Yellow","blue")
CALL WIN.TWOPEN("DEMO1","Text window DEMO1",10,13,75,16,2)
PRINT "This window was created with:-"
PRINT 'CALL WIN.TWOPEN("DEMO1","Text window DEMO1",10,13,75,16,2)'
PRINT
PRINT 'Press <CR> to create DEMO2 window ':
INPUT DUM:
CALL WIN.TWOPEN("DEMO2","Text window DEMO2", 34,2,72,6,1)
PRINT 'We now have two windows opened using WIN.TWOPEN, DEMO1 & DEMO2,
':
PRINT 'which we can switch between with WIN.TWUSE'
PRINT
PRINT 'Press <CR> to go to the DEMO1 window ':
INPUT DUM:
CALL WIN.TWUSE("DEMO1")
PRINT @(-1):'We are now back in our DEMO1 window. This was achieved by
the call'
PRINT ' CALL WIN.TWUSE("DEMO1")'
PRINT 'Press <CR> to close both windows and continue ':
INPUT DUM:
CALL WIN.TWCLOSE("DEMO1")
CALL WIN.TWCLOSE("DEMO2")
```

---

## Related Subroutines

[WIN.TWMSG](#), [WIN.TWCLOSE](#), [WIN.TWFOOT](#), [WIN.TWPULL](#), [WIN.TWPUSH](#)

## Related Script Commands

Screen Store, Screen ScrollRegion, Display Box

# WIN.TWPULL

## Syntax

WIN.TWPULL

## Description

This subroutine "pulls" or closes stacked text windows created by the WIN.TWPUSH subroutine. WIN.TWPULL closes the last window opened if there are multiple windows.

## Parameters

None

## Example

This example is a part of the WIN.TWDEMO demonstration program.

---

```
*
CALL WIN.COLOR("White","Red")
CALL WIN.TWPUSH("Stacked window 1", 20, 5, 60, 17, 3)
PRINT
PRINT "This window was created with:-"
PRINT
PRINT ' CALL WIN.TWPUSH("Stacked window 1", 20, 5, 60, 17, 3)'
PRINT
PRINT 'Press <CR> to stack another window':
INPUT DUM:
*
CALL WIN.TWPUSH("Stacked window 2",30,3,55,15,2)
PRINT 'This window was created'
PRINT 'on top of the last with'
PRINT
PRINT 'CALL WIN.TWPUSH("Stacked window 2",30,3,55,15,2)'
PRINT
PRINT 'Press <CR> to remove this'
PRINT 'window from the stack' :
INPUT DUM:
CALL WIN.TWPULL
PRINT @(-1):'We got back here by calling'
PRINT ' CALL WIN.TWPULL'
PRINT
PRINT 'Press <CR> to continue':
INPUT DUM:
CALL WIN.TWPULL
```

---

## Related Subroutines

[WIN.TWPUSH](#)

## Related Script Commands

Screen Restore, Screen ScrollRegion



# WIN.TWPUSH

## Syntax

**WIN.TWPUSH** ( *TITLE*, *LEFT*, *TOP*, *RIGHT*, *BOTTOM*, *STYLE* )

## Description

This subroutine opens a stackable text window. If you want one text window to display on top of another, use WIN.TWPUSH to "push" a window on top of an already active one, and WIN.TWPULL to "pull" the various text windows off the main screen.

## Parameters

The following table describes the parameters of the WIN.TWPUSH command:

Parameter	Description
<i>TITLE</i>	The title of the text window.
<i>LEFT</i>	Specifies the coordinate for the left side of the box.
<i>TOP</i>	Specifies the coordinate for the top of the box.
<i>RIGHT</i>	Specifies the coordinate for the right side of the box.
<i>BOTTOM</i>	Specifies the coordinate for the bottom of the box.
<i>STYLE</i>	Style of the box. See table below

## Values for STYLE

Value	Description
<i>"SPACES"</i>	blank border
<i>"SINGLE"</i>	single line border
<i>"DOUBLE"</i>	double-line border
<i>"MIX"</i>	alternate dot and block
<i>"BLOCK"</i>	solid block

## Example

This example is a part of the WIN.TWDEMO demonstration program.

---

```
*
CALL WIN.COLOR("White","Red")
CALL WIN.TWPUSH("Stacked window 1", 20, 5, 60, 17, 3)
PRINT
PRINT "This window was created with:--"
PRINT
PRINT ' CALL WIN.TWPUSH("Stacked window 1", 20, 5, 60, 17, 3)'
PRINT
PRINT 'Press <CR> to stack another window':
INPUT DUM:
*
CALL WIN.TWPUSH("Stacked window 2",30,3,55,15,2)
PRINT 'This window was created'
PRINT 'on top of the last with'
PRINT
PRINT 'CALL WIN.TWPUSH("Stacked window 2",30,3,55,15,2)'
PRINT
PRINT 'Press <CR> to remove this'
PRINT 'window from the stack' :
INPUT DUM:
CALL WIN.TWPULL
PRINT @(-1):'We got back here by calling'
PRINT ' CALL WIN.TWPULL'
PRINT
PRINT 'Press <CR> to continue':
INPUT DUM:
CALL WIN.TWPULL
```

---

## Related Subroutines

[WIN.TWPULL](#)

## Related Script Commands

Screen Store, Screen ScrollRegion

# WIN.TWUSE

## Syntax

**WIN.TWUSE** ( *NAME* )

## Description

This subroutine switches focus between two or more text windows. Create text windows with the WIN.TWOPEN subroutine, and close them with WIN.TWCLOSE.

## Parameters

The following table describes the parameters of the WIN.TWUSE command:

Parameter	Description
<i>NAME</i>	The name of the text window to display.

## Example

This example is part of the WIN.TWDEMO demonstration program.

---

```
CALL WIN.TWOPEN("DEMO2","Text window DEMO2", 34,2,72,6,1)
PRINT 'We now have two windows opened using WIN.TWOPEN, DEMO1 & DEMO2,
':
PRINT 'which we can switch between with WIN.TWUSE'
PRINT
PRINT 'Press <CR> to go to the DEMO1 window ':
INPUT DUM:
CALL WIN.TWUSE("DEMO1")
PRINT @(-1):'We are now back in our DEMO1 window. This was achieved by
the call'
PRINT ' CALL WIN.TWUSE("DEMO1")'
PRINT 'Press <CR> to close both windows and continue ':
INPUT DUM:
CALL WIN.TWCLOSE("DEMO1")
CALL WIN.TWCLOSE("DEMO2")
```

---

## Related Subroutines

[WIN.TWOPEN](#), [WIN.TWCLOSE](#)

## Related Script Commands

Screen Restore, Screen ScrollRegion

# WIN.USESTYLE

## Syntax

**WIN.USESTYLE** ( *USE.STYLE* )

## Description

This subroutine turns effect drawing on. Use this subroutine when using chiselled effect styles.

## Parameters

The following table describes the parameters of the WIN.USESTYLE command:

Parameter	Description
<i>USE.STYLE</i>	True or False flag to turn on the chiselled effect style.

## Example

This example is part of the WIN.CEDEMO demonstration program.

---

```
* Raised input only
230 TEXT = "For an easy Chiselled effect we will just raise the input
fields"
TEXT = TEXT : " which are currently displayed in REVERSE."
GOSUB 2000;* Print text
PRINT
TEXT = "Before any Chiselled effects will be displayed they must be"
TEXT = TEXT : " enabled first."
GOSUB 2000;* Print Text
GOSUB 1000;* Continue prompt
CALL WIN.USESTYLE(TRUE)
CALL WIN.SETEFFCT("Reverse","Black","LightGray","Raised",0)
PRINT 'Enable Styles:'
PRINT 'Reverse to Raised:'
CALL WIN.COLOR(CODE.COL,"")
PRINT @(18,0):"CALL WIN.USESTYLE(TRUE)":
PRINT @(18,1):'CALL
WIN.SETEFFCT("Reverse","Black","LightGray","Raised",0)'
CALL WIN.COLOR(TEXT.COL,"")
GOSUB 1000;* Continue prompt
RETURN
```

## **Related Subroutines**

[WIN.SETEFFECT](#), [WIN.EFILL](#)

## **Related Script Commands**

Effect, EffectExtent

# WIN.VERSION

## Syntax

WIN.VERSION ( *VERSION* )

## Description

Returns the version number of the running wIntegrate session.

## Parameters

The following table describes the parameters of the WIN.VERSION command:

Parameter	Description
<i>VERSION</i>	Variable set to the current version number of wIntegrate

## Example

Return the wIntegrate version number

---

```
CALL WIN.VERSION(VERSION)
PRINT "You are running wIntegrate version ": VERSION
```

---

## Related Subroutines

[WIN.SERIAL](#)

## Related Script Commands

The built-in Version constant.

# WIN.XLADDWS

## Syntax

**WIN.XLADDWS** ( *XLFILE*, *ADDFILE*, *OPTS*, *RESP* )

## Description

This subroutine adds a worksheet from an Excel file to an existing Excel file.

Excel must be installed on the PC to use this routine. The final file will be written in the format for the version of Excel installed on the PC.

The source of this subroutine is an example of using the WIN.OB... subroutines to automate Excel.

## Parameters

The following table describes the parameters of the WIN.XLADDWS command:

Parameter	Description
<i>XLFILE</i>	The file name of the Excel file to add the worksheet to.
<i>ADDFILE</i>	The Excel file containing the worksheet to add.
<i>OPTS</i>	The number of the worksheet to add worksheet after. i.e. The first worksheet is number 1, the second is number 2. The worksheet must exist in the file or an error will be shown in the Script Monitor.
<i>RESP</i>	The result of the command. See the table below. If a file can not be found then Excel will display an error message.

## Values for RESP

The following error codes are returned:



Value	Description
0	No error
1	Unable to open XLFILE.
2	Unable to get the Automation interface to XLFILE
3	Unable to open ADDFILE
4	Unable to get the Automation interface to ADDFILE

## Example

Add XLMORD1 to 5 to XLMTEST

---

```
* USERDIR is the folder containing the files
*
MAIN.FILE = USERDIR:"XLMTEST.XLS"
*
PREFIX = "XLMORD"
RESP = " "
*
FOR J = 1 TO 5
  ADD.FILE = PREFIX:J:".XLS"
  PRINT "Adding ":ADD.FILE:"...":
  ADD.FILE = USERDIR:ADD.FILE
  CALL WIN.XLADDWS(MAIN.FILE, ADD.FILE, J, RESP)
  IF RESP = 0 THEN
    PRINT "Succeeded"
  END ELSE PRINT "Failed. Error code ":RESP
NEXT J
*
```

---

## Related Subroutines

[WIN.OBGET](#), [WIN.OBSET](#), [WIN.OBSETPRP](#), [WIN.OBREL](#), [WIN.OBMETHOD](#)

## Version

4.3/5.0.1 Original

# Appendix A

## Using Host Subroutines

---

The Host Program Demonstration (WIN.DEMO) shows some of the advanced functions of wIntegrate, and is divided into several topics. This program gives you a chance to see how the wIntegrate subroutines work, and how to develop your application interface with them. At the end of each section, there is a list of related subroutines. Find complete descriptions and examples in Host Subroutines in this manual. This main demonstration program incorporates several other demonstration programs. You can run any of the programs independently from the database prompt.

You can also take a look at the code of any wIntegrate subroutine or demonstration programs in the WIN.PROGS directory.

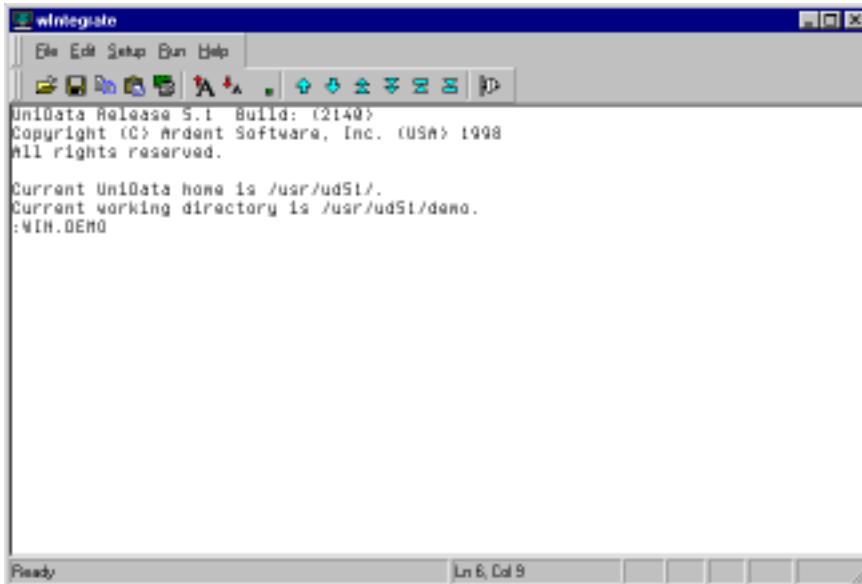
You can run the WIN.DEMO demonstration after the wIntegrate application and the host programs are installed. See the Using wIntegrate manual for application and host program installation instructions.

### **1. Start a wIntegrate Session**

Click on the wIntegrate application icon to open a new wIntegrate session.

### **2. Start the Demonstration**

Enter your database program. (The example uses UniData). From the database prompt, type in WIN.DEMO and hit the enter key to activate the Host Program Demonstration as shown in the following example:



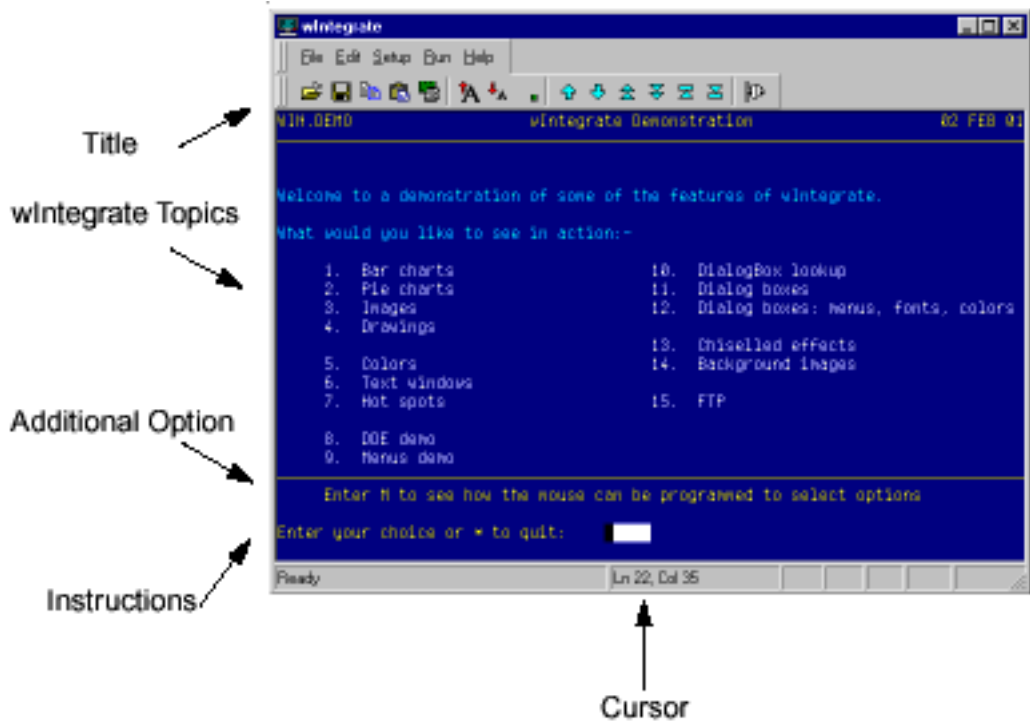
### 3. Understanding the Host Program Demonstration

The Main Screen of the wIntegrate Host Program Demonstration appears with a list of wIntegrate functions to choose from. Choose any of the topics by entering its corresponding number. When you are finished viewing the demo, enter a carriage return to go back to this main screen.

#### Tip:

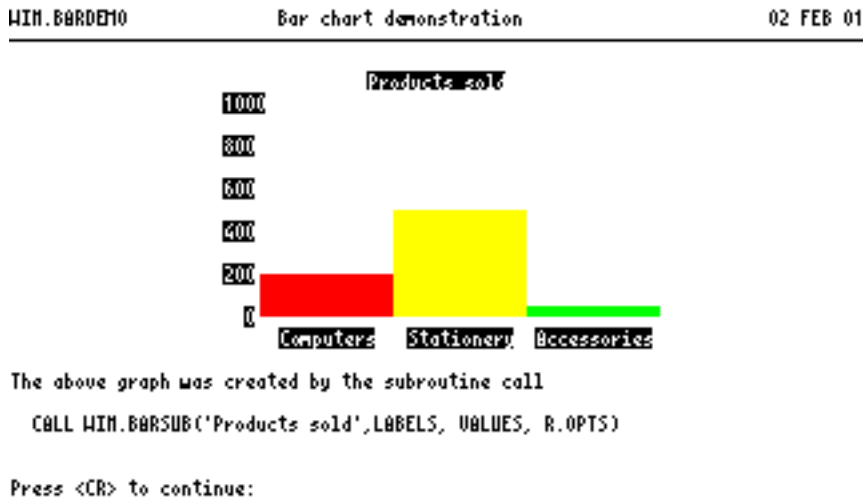
*You can also choose an option with one click of the left mouse button. Position the mouse cursor over one of the option numbers on the host screen, and click the left mouse button.*

You are given a choice of options showing how easy it is to get powerful results from wIntegrate with minimum programming effort. Each option shows the relevant line of Basic code used by the host program. All of the wIntegrate subroutines are fully described in the “Reference - Host Subroutines” section of this manual.



# 1. Bar Graphs

Select option 1 to display a bar graph.

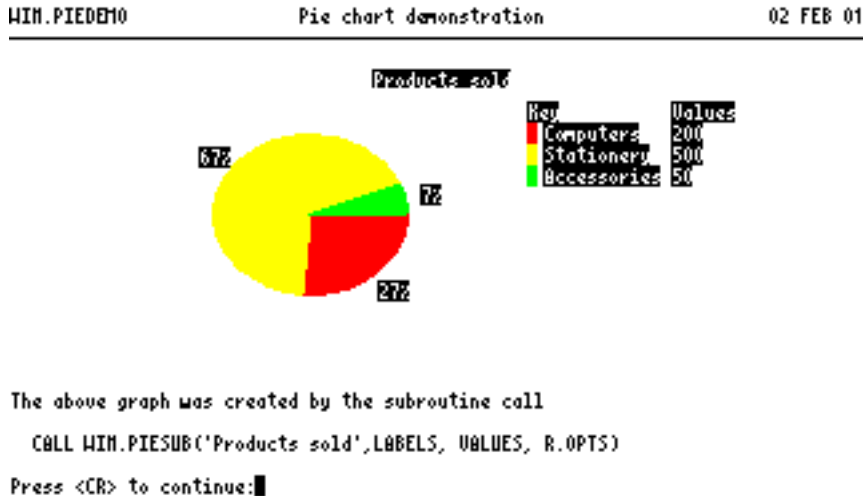


You can combine multiple bar graphs, pie charts and images on one screen. The subroutine that displays this graph is WIN.BARDEMO, which you can run from the database prompt.

Host Subroutine: WIN.BARSUB

## 2. Pie Charts

Select option 2 to display a pie chart. The subroutine that displays this graph is WIN.PIEDEMO, which you can run from the database prompt.



Host Subroutine: WIN.PIESUB


## 3. Images

Select option 3 to demonstrate how a graphic image is displayed. The subroutine that displays this image is WIN.IMDEMO, which you can run from the database prompt.

```

WIN.IMDEMO                               Image drawing demonstration          02 FEB 01
-----
Images can be displayed on the uIntegrate session screen or in a separate
window from files held on your PC in
a variety of formats:-

    .ico  Icons
    .bmp  windows bitmaps
    .wmf  windows metafiles
    .pcx  ZSoft image format
    .jpg  Joint Photographic Expert Group



The image above was displayed by :-
CALL WIN.IMAGE("IMAGE\COMPUTER.WMF", 40,4,70,18,0)

Press <CR> to continue

```

Host Subroutine: WIN.IMAGE



## 4. Drawing

Select option 4 to display the drawing capabilities of wIntegrate. The subroutine that displays these features is WIN.DRDEMO, which you can run from the database prompt.

```
WIN.DRDEMO           Drawing (overlay graphics) demonstration    02 FEB 01
```

---

1. Pen Styles
2. Brush Styles
3. Shapes
4. Brics/Pics
5. Text

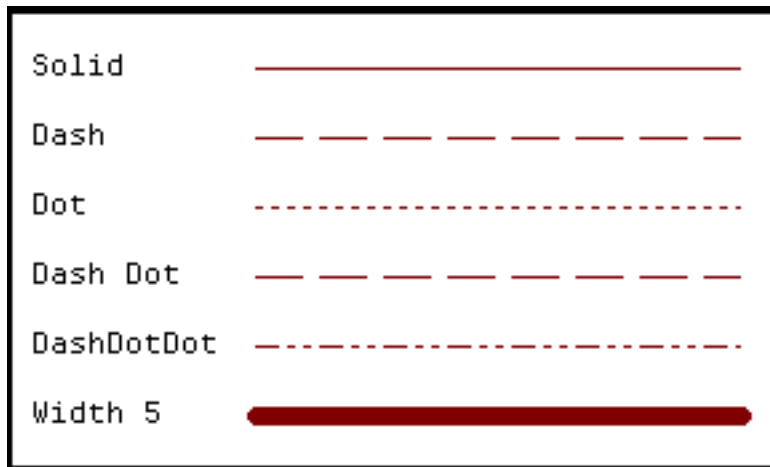
```
Enter option number or X to exit demo
```

As with images, charts and graphs, all the drawing features are scaled when the window size is changed.

Between each topic of this demonstration, the WIN.DRERASE subroutine is called to clear the drawings.

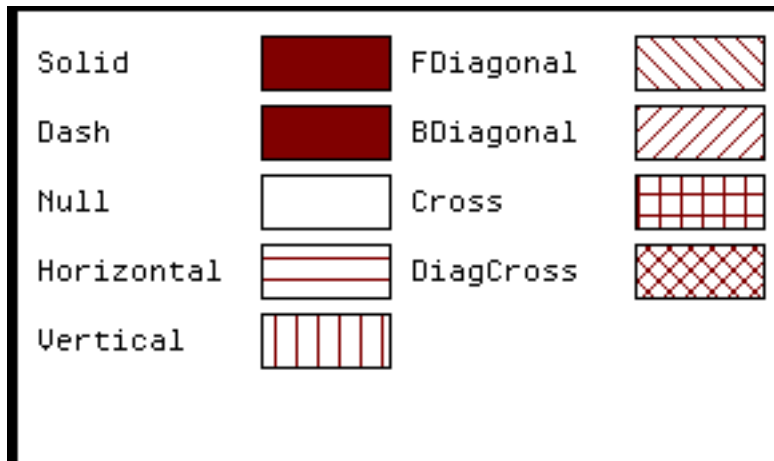
### 1. Pen

The pen determines the outlines of shapes and sets the text foreground color.



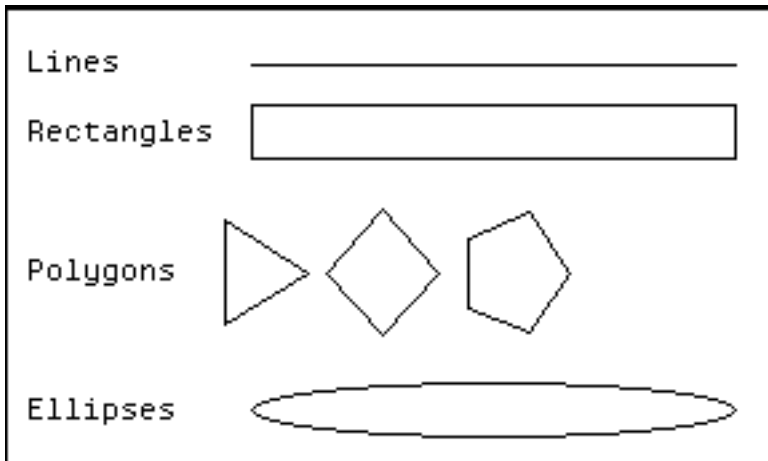
## 2. Brush

The brush determines the inside of drawn objects and sets the text background color.



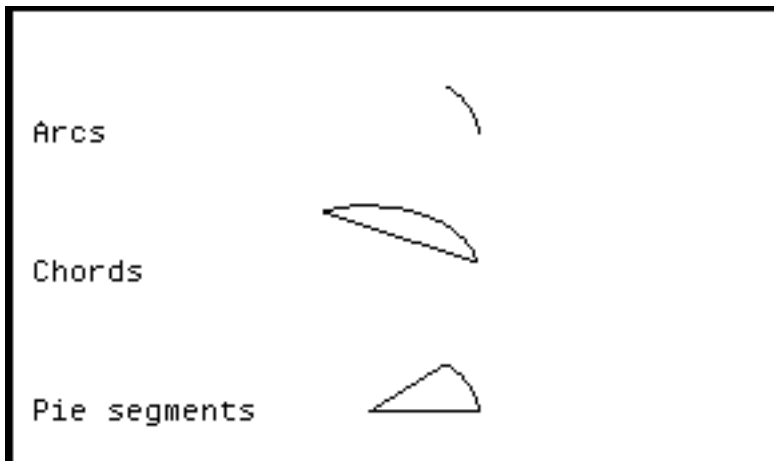
## 3. Shapes

wIntegrate can draw lines, rectangles, polygons and ellipses



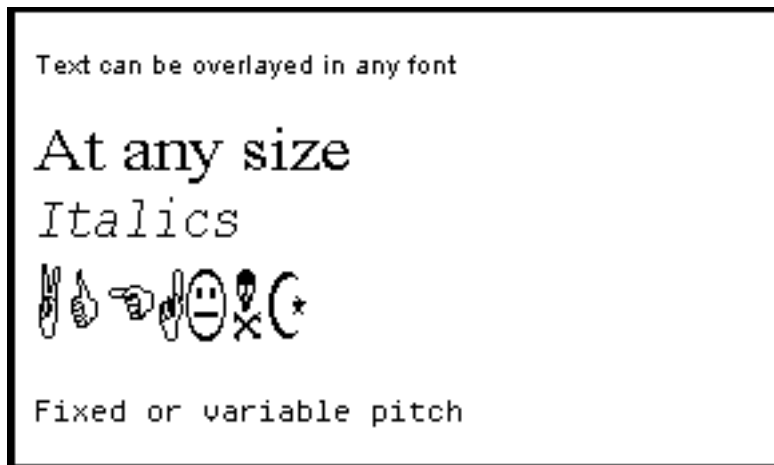
## 4. Arcs/Pies

wIntegrate can also draw arcs, chords and individual segments from a pie chart.



## 5. Text

You can display text in any font available to Windows, including the graphics font sets such as Symbol.



## Draw Subroutines

This table describes the draw subroutines

Subroutines	Description
WIN.DRARC	Draws the arc of an ellipse
WIN.DRBRUSH	Changes the color and style of the brush used in other draw routines.
WIN.DRCHORD	Draws the chord of an ellipse.
WIN.DRELL	Draws an ellipse.
WIN.DRERASE	Erases objects drawn by any of the draw subroutines.
WIN.DRFONT	Sets the font for calls to WIN.DRTEXT.
WIN.DRLINE	Draws a line.
WIN.DRMOVE	Moves the graphics cursor to a new position.
WIN.DRPEN	Changes the color, style and width of the pen used in drawing.
WIN.DRPIE	Draws a pie segment of an ellipse within a rectangle.

<b>Subroutines</b>	<b>Description</b>
WIN.DRPOLY	Draws a polygon by joining the vertices specified in points.
WIN.DRRECT	Draws a rectangle on the wIntegrate window.
WIN.DRTEXT	Draws text within a rectangle.

## 5. Colors

Select option 5 from the main menu to demonstrate the available colors. The demonstration within WIN.DEMO that displays colors is WIN.COLDEMO, which you can run from the database prompt.

```

WIN.COLDEMO           Demonstration of Colours in wIntegrate           02 FEB 01

wIntegrate supports 16 different colours. These colours can be individually
modified in the Setup Colours dialog to be any colour from your monitor's
palette.

They default to the following:-

0 Black      Background      8 Grey       Background
1 Blue      Background      9 LightBlue  Background
2 Green     Background     10 LightGreen Background
3 Cyan     Background     11 LightCyan Background
4 Red      Background     12 LightRed  Background
5 Magenta  Background     13 LightMagenta Background
6 Brown    Background     14 Yellow    Background
7 LightGrey Background     15 White     Background

To use colour just call the subroutine WIN.COLOUR(foreground,background)
Passing it either the number or name of the colour.

Press <CR> to continue:

```

Screen and text colors can be changed in Setup Colors. See the Using wIntegrate manual for instructions.

Host Subroutines: WIN.COLOR, WIN.COLOUR

## 6. Text Windows

Select option 6 from the main menu to demonstrate how to create text windows on the host screen. You can create simultaneous windows and switch between them, or stacked windows which must be removed from the stack in reverse order. The subroutine that displays text windows is WIN.TWDEMO, which you can run from the database prompt.

```
WIN.TWDEMO                                Text window demonstration                                02 FEB 01
```

---

```
Text window DEM02
We now have two windows opened using WIN.TWOPEN, DEM01 & DEM02, which we can switch between with WIN.TWUSE
Press <CR> to go to the DEM01 window
```

```
Text windows are areas of the screen. When a window is opened, the area underneath is restored.

WIN.TWDEMO provides two sets of routines to control text windows
WIN.TWOPEN(NAME, LEFT, TOP, RIGHT, BOTTOM, BORDER)
WIN.TWCLOSE(NAME)
WIN.TWUSE(NAME)
to set up
and
WIN.TWUP
WIN.TWUP
to stack
```

```
Text window DEM01
This window was created with:-
CALL WIN.TWOPEN("DEM01","Text window DEM01",10,13,75,16,2)
Press <CR> to create DEM02 window
```

```
Press <CR> to create DEM01 window
```

```

WIN.TWDEM0      Text window demonstration      02 FEB 01

Text windows are areas of the
of the rest of the
underneath is resto

wIntegrate provides
WIN.TWOPEN(NAME,
WIN.TWCLOSE(NAME)
WIN.TWUSE(NAME)
to set up one or mo

and
WIN.TWPUSH(LEFT,
WIN.TWPULL
to stack windows on

Stacked window 2
This window was created
on top of the last with
CALL WIN.TWPUSH("Stacked w
indow 2",30,3,55,15,2)
Press <CR> to remove this
window from the stack

Press <CR> to create DEM01 window

```

## Text Window Subroutines

This table describes all the wIntegrate text window subroutines.

Subroutines	Description
WIN.TWOPEN	Opens a text window.
WIN.TWCLOSE	Closes a text window opened with WIN.TWOPEN.
WIN.TWMSG	Opens a message window called "MSG".
WIN.TWUSE	Switches focus between two or more text windows.
WIN.TWFOOT	Adds a footer to a text box.
WIN.TWPULL	Closes stacked text windows.
WIN.TWPUSH	Opens a stackable text window.



## 7. HotSpots


Select option 7 from the main menu to demonstrates hot spots, which are areas of the screen that trigger a script when clicked on. You might want to display images or charts on the screen, and define these areas as hot spots leading to other displays or actions. The subroutine that displays hotspots is WIN.HSDEMO, which you can run from the database prompt.

```
WIN.HSDEMO                                HotSpots demonstration                                02 FEB 01
```

---

```
This program demonstrates setting Areas of the screen to return specific
values to the host when clicked upon by the mouse.

The Hot spots are defined by creating a record which contains a line for
Each Hot spot definition and then using:-
  CALL WIN.HOTSPOT(R.SPOTS)
The record holds the positions of the hotspots and the return value
as multivalues. Hotspots are removed by:-
  CALL WIN.HOTSPOT("")
```



```
Area 1 Blue
Area 2 LightRed
Area 3 Yellow
Area 4 LightCyan
Area 5 LightGreen
```

```
Click mouse on area or Enter x to exit
```

Click HERE to exit

Host Subroutine: WIN.HOTSPOT

## 8. Dynamic Data Exchange Demo

Choose option 8 to demonstrate the power of Dynamic Data Exchange, (DDE), a host program that controls another PC application. DDE is a Windows convention that wIntegrate incorporates as a server and as a client. For example, you can use wIntegrate to send or to receive data from another application. The Dynamic Data Exchange demo subroutine is WIN.DDEDEMO, which you can run from the database prompt.

This demonstration requires that Microsoft Excel is running, or that it is in your path.

### 1. Choose option 8

Type an "8" at the blinking cursor and hit the enter key, or click the left mouse button next to the number 8 option, DDE Demo. The following screen appears:

```
WIN.DDEDEMO           Dynamic Data Exchange demonstration           02 FEB 01
-----
This program demonstrates DDE. You must have Microsoft Excel installed
If Excel is not in your DOS PATH, you should start it manually now

To start our conversation with another application we use:-
CALL WIN.DDEOPEN(NAME, APP.NAME, TOPIC, STARTED.ON)

If the link opens ok, we can do the following:-
CALL WIN.DDEREQ(NAME, ITEM, VALUE) to fetch data
CALL WIN.DDEPOKE(NAME, ITEM, VALUE) to send data
CALL WIN.DDEEXEC(NAME, MACRO) to execute a macro

Finally we close the link with
CALL WIN.DDECLOSE(NAME)

The script language allows further control over DDE

Do you want to run the demonstration? (Y/N):
```

The screen lists the related Basic program code lines to use when developing an application that calls and controls another application.

### 2. Continue the Demonstration

The line on the bottom of the screen prompts you to continue or exit the demo. Type a "Y" and hit the enter key.

The program first checks if Excel is already running, and starts Excel if it is not started.

The screen below shows an error message if wIntegrate cannot successfully start Excel. If you get this message, hit a carriage return to exit the message box. Go to Program Manager and start Excel, then return to wIntegrate and run the DDE demo again.

```
WIN.DDEDEMO           Dynamic Data Exchange demonstration           02 FEB 01

This program demonstrates DDE. You must have Microsoft Excel installed
If Excel is running, the program will attempt to start Excel.
To start Excel, the program will call the following command:
CALL WIN.DDEOPEN("EXCEL_LINK","excel","System", 0H)
If the link is not found, the program will attempt to start Excel
CALL WIN.PCSTART("EXCEL", "", "", RESP)
CALL WIN.DDEOPEN("EXCEL_LINK","excel","System", 0H)
CALL WIN.DDEOPEN("EXCEL_LINK","excel","System", 0H)
CALL WIN.DDEOPEN("EXCEL_LINK","excel","System", 0H)
We are unable to start Excel, so the demo will stop.
If you do own Excel, try starting it and then re-running the
demo.
Finally, the program will call the following command:
CALL WIN.DDEOPEN("EXCEL_LINK","excel","System", 0H)
The script will continue.
```

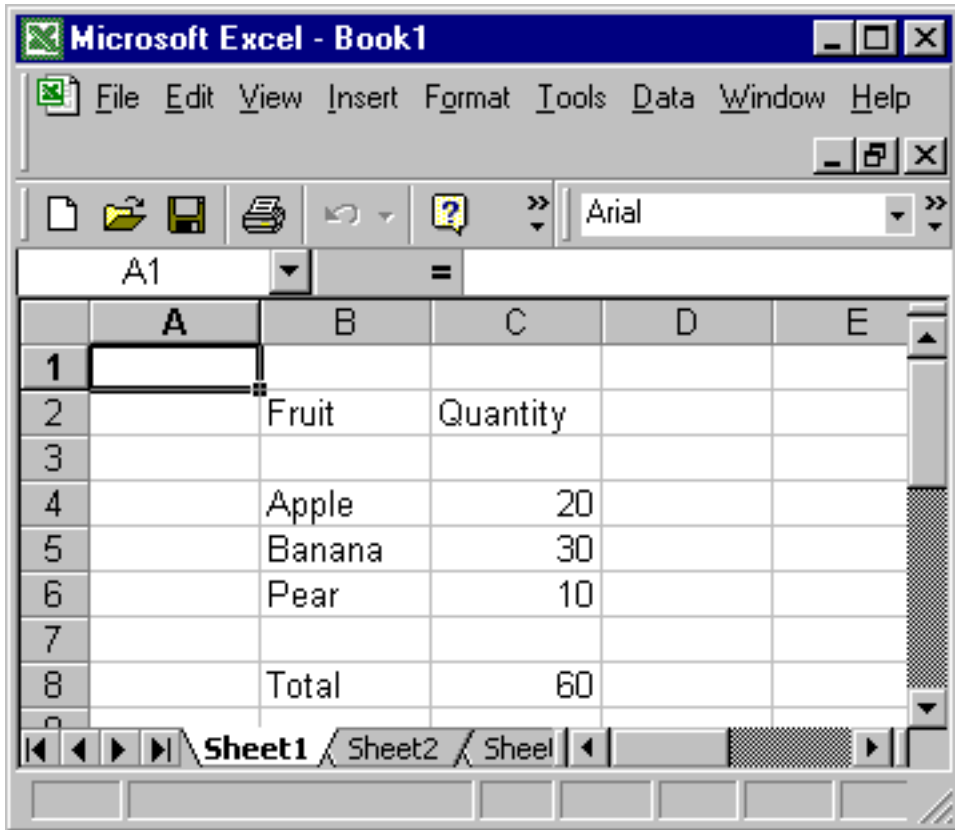
### 3. Resize and Move the Excel Window

When Excel is started, the screen displays it on top of the wIntegrate window. Click on the double arrow button in the upper right-hand corner to resize the Excel window, and move it so that you can see both the wIntegrate window and the Excel window.

### 4. Enter Through Screens

Click on the wIntegrate window to see the demo. The next few screens describe the process of the demo. Just hit the enter key to continue to the next screen. (Not all of the screens are displayed here.)

Now some descriptions and numbers are sent to the Excel spreadsheet SHEET1.XLS. A SUM command is sent to another cell to total the numbers. Finally the columns are selected and the Format Column Width Best Fit command is executed so that the columns accommodate the data correctly. The next example shows the result if the Dynamic Data Exchange in Excel:



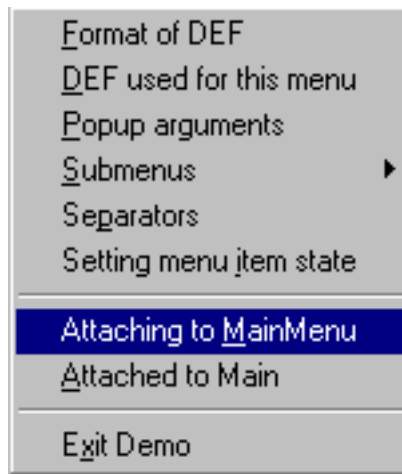
## DDE Subroutines

Subroutines	Description
WIN.DDEOPEN	Starts a DDE conversation with another application.

Subroutines	Description
WIN.DDECLOSE	Stops a DDE link.
WIN.DDEREQ	Gets data from another application.
WIN.DDEEXEC	Executes a macro or command in another application.
WIN.DDEPOKE	Sends data from wIntegrate to another application.
WIN.DDETIME	Sets the timeout for a DDE link.

## 9. Menus

Select option 9 to display how you can create wIntegrate menus from the host. Press Enter to display the first menu, shown below. The subroutine that displays menus is WIN.MENUDEMO, which you can run from the database prompt. The code for this demo program may be especially useful if you need to learn how to create and attach menus and submenus.



Note: The first menu is a popup menu. Click on "Attached to Main" to see the same menu appear on the Windows menu bar.

The commands for controlling menus are explained in Host Subroutines.

Click on "Exit demo" to return to the main demonstration menu.

### Menu Subroutines

Subroutines	Description
WIN.MENUATT	Attaches a submenu to an existing menu.
WIN.MENUDEL	Deletes a submenu from an existing menu.

<b>Subroutines</b>	<b>Description</b>
WIN.MENUDET	Detaches a menu.
WIN.MENUIN	Enables a menu, then waits for input.
WIN.MENULOAD	Loads a menu to the PC before it is attached and shown.

## 10. DialogBox lookup

Select option 10 to show how to create a Windows dialog box that can look up items for selection by the user.

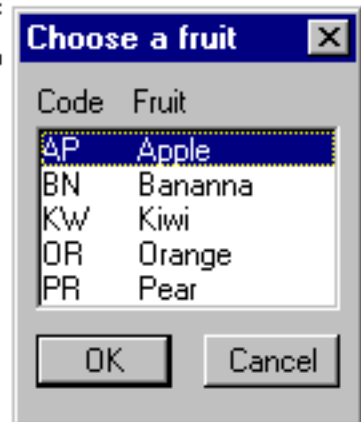
WIN.LUDE10

Dialog box data lookup demonstration

02 FEB 01

This program demonstrates popping up a dialog box to prompt for a choice of an item from a list.

The item can be chosen with the mouse or cursor keys then clicking on OK or pressing <CR>. Alternatively you can double click with the mouse on



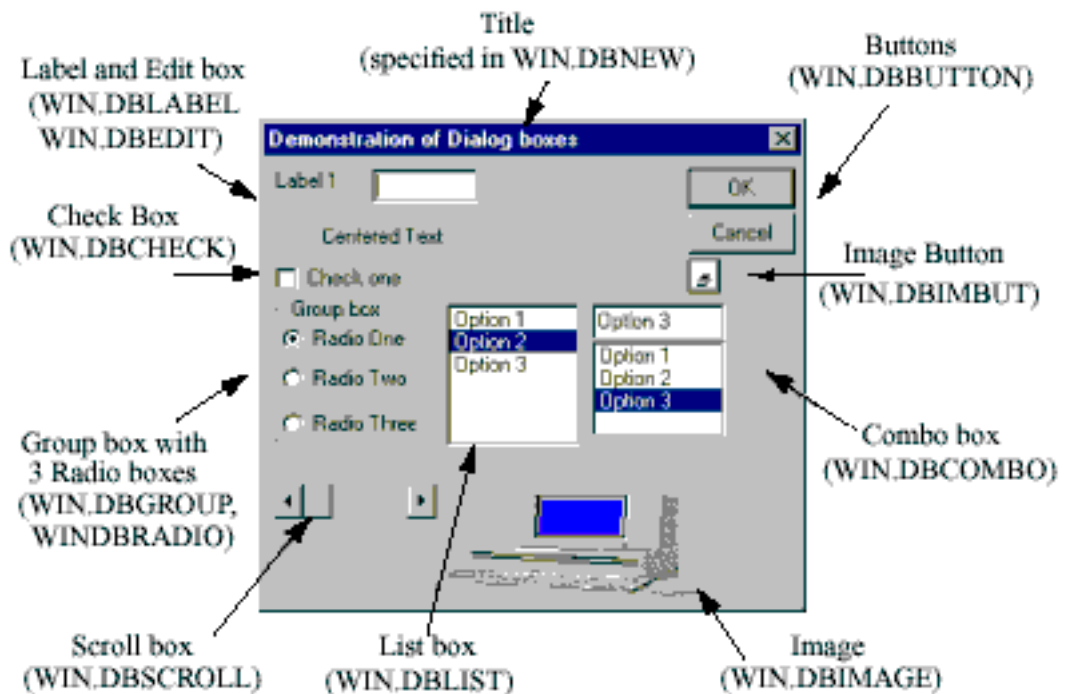
Host Subroutine: WIN.LOOKUP



## 11. Dialog Boxes

Select option 11 to show how a Windows dialog box is created and monitored from a host program. Each time the user clicks on a control (i.e. one of the elements of the dialog box), an appropriate message is sent to the host.

When you click OK to end the dialog box, the values of each control are sent to the host.



Use the WIN.DB subroutines to create dialog boxes. The following steps describe how to create a dialog like the one above:

1. You must use WIN.DBNEW to create a dialog box, as this command starts the definition of a dialog box.
2. Now you may add controls within the box to create edit and text boxes, labels, etc. (See WIN.DBEDIT, WIN.DBTEXT, and WIN.DBLABEL.)

3. Load the new dialog box using WIN.DBLOAD.
4. Show the box with WIN.DBSHOW.
5. To close the dialog box, use WIN.DBEND, then delete the box from memory with WIN.DBDEL.

## Dialog Box Subroutines

The following lists other dialog box subroutines available.

Subroutines	Description
WIN.DBATTACH	Attaches one dialog box to another.
WIN.DBBUTTON	Creates a push button to execute a command. The OK and Cancel buttons in the example use this subroutine.
WIN.DBCAPT	Changes the caption or title of a dialog box.
WIN.DBCHECK	Creates a check box that toggles on or off.
WIN.DBCOMBO	Creates a combo box, which is similar to a list box, but with an edit-type box showing the chosen option.
WIN.DBEDIT	Creates an edit box allowing the user to enter text.
WIN.DBENABLE	Enables or disables dialog box controls.
WIN.DBEND	Removes the dialog from the screen. (Use WIN.DBDEL to remove the box from memory.)
WIN.DBEVENT	Monitors input and returns dialog and control types.
WIN.DBFETCH	Retrieves the next event.
WIN.DBFOCUS	Sets the focus to a dialog box control.
WIN.DBGET	Gets the value of a dialog control.

Subroutines	Description
WIN.DBGROUP	Creates a group box for radio buttons. Use this when you want the user to choose one of a group of options.
WIN.DBIMAGE	Displays an image file in a dialog box.
WIN.DBIMBUT	Creates a push button from an image file.
WIN.DBINPOK	Validates the input after a validate event.
WIN.DBLABEL	Creates a label for an edit box or other control.
WIN.DBLIST	Creates a list box.
WIN.DBLOAD	Loads a dialog box created by WIN.DBNEW.
WIN.DBMSGBOX	Creates a message box that displays when a control is clicked.
WIN.DBMSIZE	Sets or changes the size and scroll steps for a dialog box.
WIN.DBNEVENT	Gets the next event when WIN.DBOPTION "Vn" is used.
WIN.DBNEW	Starts the creation of a dialog box.
WIN.DBOPTION	Adds Validate options for a dialog box.
WIN.DBPOST	Posts an event to an event queue.
WIN.DBRADIO	Creates radio buttons, which work like check boxes unless used in a group box.
WIN.DBSCROLL	Adds a scroll bar to a dialog box.
WIN.DBSELECT	Selects one of several options in a list or combo box.
WIN.DBSET	Sets or changes dialog box controls.
WIN.DBSHOW	Displays a dialog box created with WIN.DBNEW.
WIN.DBSTATUS	Determines if a dialog box is loaded.
WIN.DBTABS	Sets the tabstops in a list box.
WIN.DBTEXT	Creates a text box in a dialog.

Subroutines	Description
WIN.DBENALL	Enables or disables the dialog box window.

## 12. Chiselled Effects

This part of the demonstration is especially useful, because it shows you step-by-step how to make a basic data entry screen look like a Windows dialog box. With effects, you can give your screen depth and emphasis on important areas.

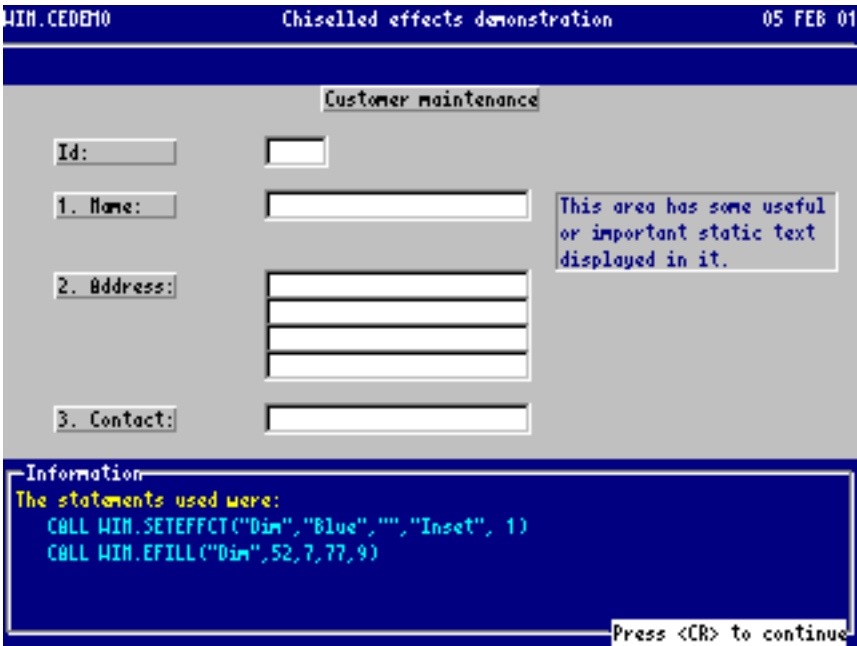
In the first part of the demo, it shows a simple data entry screen:

```
WIN.CED010      Chiselled effects demonstration      05 FEB 01
Customer maintenance
Id: [redacted]
1. Name: [redacted]      This area has some useful
                        or important static text
                        displayed in it.
2. Address: [redacted]
3. Contact: [redacted]

Information
The first thing to do is convert the normal text color to black on light
gray.
Press <CR> to continue
```

Continue to press Enter, and the screen changes with more effects. The bottom of the screen explains which subroutines are used to accomplish the different effects.

After effects have been added, the same screen shown above looks like the following example:



## Screen Effect Subroutines

Subroutines	Description
WIN.SETEFFECT	Sets the color and style for a terminal effect
WIN.USESTYLE	Turns effect drawing on.
WIN.EFFECT	Sets the effect of the text that follows.
WIN.EFILL	Fills an area of the screen with an effect.

# Appendix B

## The Service Subroutine

---

### Introduction

wIntegrate has client-side tools which allow the user to see data from the host database. These include the Query Builder, Import File, Export File and the new Report Wizard. Security is an issue here as not all users may be permitted to see the contents of all files, or fields within records. So we have created the concept of the "service subroutine" to filter what the user sees. It is called by three wIntegrate host programs. These are:

WIN.TRANSFER which acts as the host end of data transfer processing (ie whenever the File Transfer Monitor is shown)

WIN.SERVER, the host end of Query Builder, Report Wizard and other "client/server" processes.

WIN.RWSUB, a subroutine called by WIN.SERVER while the Report Wizard is running

The service subroutine is called by WIN.TRANSFER and WIN.SERVER at two points. First, before the host program sends data to the PC, for example a list of files, dictionary items or hold entries. Second, as a final check immediately before a process is executed, for example in Query Builder where the user might enter a query statement manually. Service subroutines are automatically used in the same way by both WIN.TRANSFER and WIN.SERVER.

### Service Subroutine

You can choose the name for the subroutine. To activate it, create a new item in file WIN.PROGS with the item ID of "SERVICESUB". Field 1 of this item should be the name of your service subroutine.

Note that parameter items such as "SERVICESUB" can be stored in a separate file WIN.PARAMS, so the user doesn't need system permissions to read the program file WIN.PROGS. (If in doubt, take a look at WIN.SERVER and see how it determines the name of the service subroutine.)

In these examples we will assume the service subroutine is called MYSERVICE, but you can use any name.

```
:AE WIN.PROGS SERVICESUB
Top of "SERVICESUB" in "WIN.PROGS", 1 line, 9 characters.
*--: P
001: MYSERVICE
Bottom.
*--:
```

The existence of this item will cause WIN.TRANSFER and WIN.SERVER to call your subroutine with the following parameters.

## Syntax: CALL MYSERVICE (ACTION, MAT SS.ARGS, ARG)

### Parameters:

Parameter	Description
ACTION	Action code for current call to the service routine
SS.ARGS	Array of 30 elements for additional information/parameters
ARG	Main parameter/return value

The service subroutine is always called with ACTION = 1 (SSA.INIT) which allows the service routine to tell the calling program at what times it requires to be called. This means only the actions that the service subroutine is required to process need any code.

### Parameter Array (MAT SS.ARGS)

The parameter array contains the following entries



Index	Name	Description
1	SS.PROGID	Program Id - set before first call. Currently 1 - Host Server (WIN.SERVER), 2 - File Transfer (WIN.TRANSFER), 3 - Report Wizard (WIN.RWSUB)
2	SS.PARAM	Environment parameters - This is the dynamic array set by the WIN.PARAM subroutine. It contains host-specific information such as the maximum record size.
3	SS.RESULT	Second result parameter - usually specifies if ARG validates or what to do with any change to ARG.
4	SS.EXTRA1	Additional information. See individual action descriptions for details.
5	SS.EXTRA2	Additional information. See action description
6 to 9	reserved	Reserved for future expansion
10 to 30	user	Available to the user

## Action code summary

Index	Name	Description
1	SSA.INIT	Initialization - first action which is always called
2	SSA.FILESEL	Set-up/modify select for List of files
3	SSA.FILECHK	Check each file name before displaying
4	SSA.FIELDSEL	Set-up/modify select for list of fields
5	SSA.FIELDCHK	Check each field name before displaying
6	SSA.EXECCHK	Check commands to be executed (WIN.SERVER only)
7	SSA.TRANCHK	Check parameters for file transfer (WIN.TRANSFER only)

Index	Name	Description
8	SSA.SLAVEON	Set-up/modify environment and turn on slave printing
9	SSA.SLAVEOFF	Reset environment and turn off slave printing
10	SSA.HPRTON	Set-up environment before host printing
11	SSA.HPRTOFF	Reset environment after host printing
12	SSA.READCHK	Check OK to read field/item
13	SSA.WRITECHK	Check OK to write field/item
14	SSA.DELETECHK	Check OK to delete field/item

## Action Code Details

The SS.PROGID and SS.PARAM members of the SS.ARGS array are set up for all calls. The user area of the array is left untouched between calls to the various subroutines.

Each action below has the same format. A title which consists of action code, action name and a brief description. "On Entry" lists the parameters that are set for the routine when it is called by the host program. "On Exit" lists the parameters that the routine can change. A more details description of the subroutine follows.

### Action code 1 - SSA.INIT - Initialization

Parameter	On Entry	On Exit
ARG	""	ARG is a dynamic array specifying on which actions to call the routine

This call is made at the beginning of the program with the SS.PROGID and SS.PARAM fields completed. It allows the user area of SS.ARGS to be initialized and ARG to be set up to specify when the service subroutine should be called. Each action to be specified is set up by putting the number 1 in the field with the same index as the action code.

### Example:

```
ARG<SSA.FILESEL> = 1      ;* Want file selection hook
ARG<SSA.FIELDSEL> = 1      ;* Want field selection hook
```

## **Action code 2 - SSA.FILESEL - File selection**

Parameter	On Entry	On Exit
ARG	Default select statement to use	New selection statement to use or list of filenames
SS.EXTRA1		Account name
SS.RESULT		1 - ARG contains select statement, 2 ARG is list of file names

This call is made before the select statement is run to provide the list of filenames for the Files script command or the Files button on the file transfer.

The service subroutine can modify the list of files as seen by the user.

## **Action code 3 - SSA.FILECHK - Check file name**

Parameter	On Entry	On Exit
ARG	file name	file name to use (SS.RESULT = 1)
SS.EXTRA1	Account Name	
SS.RESULT		1 - use file name, 0 - do not use file name

This call is made before a file name is passed up to the PC from the Files script command and the Files button on the file transfer. It allows extra checks to be made to a file name just before it is added to the list of files.

## **Action code 4 - SSA.FIELDSEL - Field selection**

Parameter	On Entry	On Exit
ARG	default select statement	new select statement or list of field names
SS.EXTRA1	Filename	
SS.EXTRA2	Account	
SS.RESULT		1 - ARG contains select statement, 2 - ARG is list of field names

This call is made before the select statement is run which provides the list of field names for the Fields script command or the Fields button on the File Transfer. It allows the select statement used for field listing to be modified or replaced by a list of specific field names.

## Action code 5 - SSA.FIELDCHK - Check field name

Parameter	On Entry	On Exit
ARG	field name	
SS.EXTRA1	Filename	
SS.EXTRA2	Account	
SS.RESULT		1 - use field name, 0 - do not use field name

This call is made before a selected field is transferred to the host in the Fields script command or from the Fields button on the File Transfer. It allows extra checks to be made to a field name just before it is added to the list of fields.

## Action code 6 - SSA.EXECCHK - Check commands to be executed

Parameter	On Entry	On Exit
ARG	dynamic array of lines to be executed	actual statement to be executed

Parameter	On Entry	On Exit
SS.RESULT		0 - Invalid execute statement, 1 - Statement OK
SS.EXTRA1		error message or "" for no message (SS.RESULT = 0)

This routine is called just before the host server executes the statements passed to it by the MExec command (which is used to run the query for the Query Builder). It allows the statements to be checked to ensure correct file names and field names.

## **Action code 7 - SSA.TRANCHK - Check parameters for file transfer**

Parameter	On Entry	On Exit
ARG	Information to be checked (format same as in USERCHECK subroutine)	
SS.RESULT		1 - OK, 0 - invalid parameters
SS.EXTRA1		error message (SS.RESULT = 0)

This routine is called just before the actual file transfer is started, after the details have been sent from the PC. It allows extra checks on the file transfer file names, fields etc.

## **Action code 8 - SSA.SLAVEON - Slave printer on**

Parameter	On Entry	On Exit
ARG	Destination for capture usually "Printer"	Destination for capture
SS.RESULT		0 - all processing done here, 1 - do standard processing, 2 - Just turn on capture to printer (in main program)

This routine is called to just before the data for slave printing is sent to the screen in the host server program. It can be used to set-up the host/PC environment for the local printout and also to run a specific Capture script command to turn on the local printer.

If SS.RESULT = 0 on exit the local printer must be turned on within this routine.

Use lines similar to:

```
SCRIPT = "Screen Off"
```

```
SCRIPT<-1> = 'Capture On ServerPrint,'" : ARG : '''  
CALL WIN.HSCRIPT(SCRIPT)
```

## Action code 9 - SSA.SLAVEOFF - Slave printer off

Parameter	On Entry	On Exit
SS.RESULT		0 - all processing done in subroutine, 1 - do standard processing, 2 - just turn off capture to printer

This routine is called after the listing for local printing has finished. Use it to reset local/PC environment if it was changed in SSA.SLAVEON or to turn off the capture.

If SS.RESULT = 0 on exit the local printer must be turned off in this subroutine.

## Action code 10 - SSA.HPRTON - Host printer set-up

Parameter	On Entry	On Exit
(none)	No entry parameters	No exit parameters

This routine is called before printing to the host. It allows the print destination to be changed or the settings to be modified.

## Action code 11 - SSA.HPRTOFF - Host printer reset

Parameter	On Entry	On Exit
(none)	No entry parameters	No exit parameters

This routine is called after printing to the host. It allows the host printer destination settings to be reset.

## **Action code 12 - SSA.READCHK - Check OK to read field/item**

Parameter	On Entry	On Exit
ARG	file name	
SS.RESULT		0 - Access denied, 1 - Access granted
SS.EXTRA1	item name	
SS.EXTRA2	field number (or "" for whole record)	

This routine is called to check that a user has authority to read a host field or item.

## **Action code 13 - SSA.WRITECHK - Check OK to write field/item**

Parameter	On Entry	On Exit
ARG	file name	
SS.RESULT		0 - Access denied, 1 - Access granted
SS.EXTRA1	item name	
SS.EXTRA2	field number (or "" for whole record)	

This routine is called to check that a user has authority to read a host field or item.

## Action code 14 - SSA.DELETECHK - Check OK to delete field/item

Parameter	On Entry	On Exit
ARG	file name	
SS.RESULT		0 - Access denied, 1 - Access granted
SS.EXTRA1	item name	

## Example Subroutines

These example subroutines are installed in the WIN.PROGS file where all wIntegrate host programs are stored.

### WIN.SSUB

This service subroutine reads the file/field select statements from an item called "FILE.SELECT" on WIN.PROGS.

The format of this record is:

- Field 1 - File name to select for list of filenames
- Field 2 - Alternate selection to use for list of filenames
- Field 3 - Alternate selection to use for list of fields

### WIN.SSUB1

This service subroutine demonstrates all the non-printing service hooks. It was compiled for UniBasic and will hence need minor modifications to get it to work correctly on other host systems.



# Index

---

## A

- Activating another Desktop Window, [13](#)
- Adding a worksheet to Excel, [464](#)
- Application directory, [16](#)
- ASCII to FT conversion, [17](#)
- Assigning a script variable, [18](#)
- Automation
  - Checking if an object exists, [334](#)
  - Creating a new object, [340](#)
  - Getting a property, [337](#)
  - Getting an object, [335](#)
  - Releasing an object, [341](#)
  - Running a method, [338](#)
  - Setting a new object from a property, [342](#)
  - Setting a property, [343](#)
  - Specifying an argument type, [344](#)

## B

- Background image for the session, [21](#)
- Bar chart creation, [19](#)
- Box drawing, [22](#)

## C

- Charts

- Creating a bar chart, [19](#)
  - Creating a pie chart, [384](#)
- Check if a Window's Task is running, [442](#)
- Check if another application is running, [15](#)
- Checking if wIntegrate is running, [28](#)
- Client information, [30](#)
- Clipboard
  - Reading text, [24](#)
  - Writing text, [26](#)
- Color for text, [31](#)
- Colour for text, [33](#)
- Command line scripts, [35](#)
- Conversions
  - ASCII to FT, [17](#)
  - FT to ASCII, [248](#)
- Creating a pie chart, [384](#)
- Cursor display, [37](#)

## D

- Dialog boxes
  - Adding a checkbox, [48](#)
  - Adding a combo box, [52](#)
  - Adding a control, [55](#)
  - Adding a date/time control, [58](#)
  - Adding a grid, [86](#)
  - Adding a group box, [88](#)
  - Adding a header control, [90](#)
  - Adding a list box control, [103](#)
  - Adding a list view control, [105](#)
  - Adding a menu, [143](#)
  - Adding a progress bar, [129](#)

---

- Adding a pushbutton, [45](#)
- Adding a radio button, [131](#)
- Adding a rectangle, [43](#)
- Adding a scroll bar, [135](#)
- Adding a tab control, [151](#)
- Adding a text control, [154](#)
- Adding a text label, [100](#)
- Adding a track bar control, [156](#)
- Adding a tree view control, [158](#)
- Adding an Active X control, [41](#)
- Adding an animation control, [38](#)
- Adding an edit control, [61](#)
- Adding an image control, [92](#)
- Adding an initialization script, [97](#)
- Adding an up/down control, [162](#)
- Adding push button with an image, [94](#)
- Attaching, [40](#)
- Changing a caption, [47](#)
- Checking for an event, [77](#)
- Checking status, [149](#)
- Checking the next event, [69](#)
- Closing a displayed dialog box, [67](#)
- Converting to/from dialog units, [160](#)
- Creating, [120](#)
- Deleting from memory, [57](#)
- Disabling controls, [64](#), [65](#)
- Displaying, [145](#)
- Displaying a child dialog box, [50](#)
- Enabling controls, [64](#), [65](#)
- Events to receive, [75](#)
- Getting data, [81](#), [83](#)
- Getting text from an edit control, [60](#)
- Getting the value of a property, [85](#)
- Initializing a control property, [96](#)
- Invoking a control method, [108](#)
- Loading the definition on the PC, [107](#)
- Moving, [112](#)
- Moving a control, [117](#)
- Options, [123](#)

- Pop up a dialog box, [147](#)
- Re-posting events, [127](#)
- Reading the next event, [70](#), [73](#)
- Returning the next event, [118](#)
- Returning the size and position of a control, [133](#)
- Selecting a list item, [137](#)
- Setting a control value, [139](#)
- Setting a controls color, [141](#)
- Setting a controls font, [142](#)
- Setting a controls property, [144](#)
- Setting list box tab stops, [153](#)
- Setting text in an edit control, [63](#)
- Setting the input focus, [79](#)
- Setting the maximum size, [115](#)
- Setting the minimum size, [111](#)
- Showing an attached message box, [113](#)
- Sizing a control, [117](#)
- Splitting the dialog into panels, [125](#)
- Verifying input, [98](#)

#### Directory

- Checking if exists, [352](#)
- Creating, [362](#)
- Deleting, [375](#)

#### Displaying text on the status bar, [441](#)

#### Drawing

- Arcs, [178](#)
- Brush style and color, [180](#)
- Chords, [182](#)
- Ellipses, [184](#)
- Erasing, [186](#)
- Lines, [190](#)
- Moving the graphics position, [191](#)
- Pen color and style, [192](#)
- Pie segments, [195](#)
- Polygons, [197](#)
- Rectangles, [199](#)
- Setting the text font, [188](#)
- Text, [201](#)

Dynamic Data Exchange (DDE)

- Closing a conversation, [164](#)
- Executing a macro, [166](#), [168](#)
- Opening a conversation, [170](#)
- Requesting data, [174](#)
- Sending data, [172](#)
- Setting the time out, [176](#)

E

E-mail

- Address lookup, [305](#)
- Check if email is available, [307](#)
- Deleting a message, [308](#)
- Finding a message, [310](#)
- Getting the position on a click, [327](#)
- Lookup recipient details, [313](#)
- Next available, [315](#)
- Reading a message, [318](#)
- Sending a message, [321](#)

ECL Command Stacker, [444](#)

ECL Stack

- Turning off, [439](#)
- Turning on, [440](#)

Editing host programs, [203](#)

Editing input, [208](#), [212](#), [291](#)

Evaluating a script expression, [215](#)

Exporting a file, [216](#)

F

File

- Browse for a file name, [345](#)
- Checking for end of file, [356](#)
- Checking if exists, [357](#)
- Closing, [347](#)

Copying, [348](#)

Creating, [349](#)

Deleting, [351](#)

Getting information, [359](#)

List of files and directories, [360](#)

Moving, [364](#)

Opening, [365](#)

Opening in a PC editor, [354](#)

Printing on the PC, [368](#)

Reading a line, [373](#)

Reading data, [369](#)

Reading the whole file, [371](#)

Setting read/write position, [366](#)

Writing data, [382](#)

File Transfer

Export, [216](#)

Host program, [446](#)

Import, [284](#)

Spooling to the host printer, [426](#)

Filling an area with an effect, [206](#)

FT to ASCII conversion, [248](#)

FTP

Checking if a directory exists, [225](#)

Checking if a file exists, [227](#)

Closing a file, [222](#)

Connecting to the server, [223](#)

Creating a directory, [233](#)

Deleting a file, [224](#)

Disconnecting from the server, [226](#)

Getting a file, [228](#)

Getting a list of files and directories, [231](#)

Getting file information, [230](#)

Getting the current directory, [229](#)

Opening a remote file, [234](#)

Reading data from a remote file, [238](#)

Removing a directory, [239](#)

Running a script stored on the server,  
[240](#)

Sending a file to the server, [237](#)

---

- Setting a script variable using FTP, [244](#)
- Setting the current directory, [243](#)
- Setting the read/write position, [236](#)
- Transferring and running a script, [241](#)
- Writing data to a remote file, [246](#)

## G

- Getting a parameter, [251](#)
- Getting a value, [252](#)
- Getting a variable as a list, [250](#)
- Getting a variables value, [249](#), [253](#)

## H

- Help
  - Command line invokation, [254](#)
  - Showing help for a topic id, [255](#)
  - Showing help for a topic name, [256](#)
- Hiding the screen, [177](#)
- Host Server, [414](#)
- Host version information, [258](#)
- HotSpots
  - Defining, [259](#)
  - Defining with styles and colors, [261](#)
- Hourglass, [257](#)

## I

- Image list
  - Adding a bitmap, [265](#)
  - Adding an icon, [268](#)
  - Counting images, [266](#)
  - Deleteing, [267](#)

- Getting information on an image, [269](#)
- Loading from a file, [271](#)
- New, [273](#)
- Removing an image, [275](#)
- Size of an image list, [270](#)

- Image on the screen, [276](#)

- Image Window

  - Changing the image, [278](#)
  - Closing the window, [280](#)
  - Opening, [282](#)

- Importing a file, [284](#)

- Information Window, [288](#)

- Input dialog box, [289](#)

- Invoking a menu option, [290](#)

## K

- Key programming, [219](#)

## L

- License information, [292](#)

- Lookup dialog box, [293](#)

## M

- Menu

  - Attaching a submenu, [295](#)
  - Checking an item, [332](#)
  - Deleting from memory, [297](#)
  - Detaching, [299](#)
  - Enabling an item, [332](#)
  - Input from a menu, [301](#)
  - Loading on the PC, [303](#)

- Popping up, [389](#)
- Setting the state of an item, [332](#)
- Menu Options
  - Getting a related variable, [251](#)
  - Invoking, [290](#)
  - Setting a related variable, [419](#)
  - Showing, [423](#)
- Message Box, [329](#)
- Mouse
  - Button programming, [324](#)
  - Setting up for WIN.MOUSEIN, [326](#)

## P

- Parameters
  - Getting value, [251](#)
  - Setting value, [419](#)
- Playing back a recorded file, [387](#)
- Printing
  - Pausing a printout, [393](#)
  - Turning off printing to the PC, [391](#)
  - Turning on printing to the PC, [392](#)

## R

- Recording
  - Pausing, [399](#)
  - Turning off, [395](#)
  - Turning on, [397](#)
- Remote Sessions
  - Checking if a session is running, [403](#)
  - Executing a script command, [401](#)
  - Getting the current sessions name, [404](#)
  - Running a script, [405](#)
  - Starting another session, [407](#)
- Running a host script, [263](#)

- Running a PC application, [376](#), [378](#)
- Running a script from the host, [264](#)
- Running a script on the PC, [381](#)

## S

- Screen
  - Copy to file or printer, [410](#)
  - Deleting a stored screen, [430](#)
  - Effect colors and styles, [204](#)
  - Effect filling, [206](#)
  - Hiding, [177](#)
  - Loading from a file, [424](#)
  - Pulling from a stack, [428](#)
  - Pushing onto a stack, [429](#)
  - Restoring a stored screen, [432](#)
  - Saving the output state, [434](#)
  - Saving to a file, [433](#)
  - Setting the display effect, [416](#)
  - Storing to memory, [436](#)
  - Turning output on and off, [409](#)
  - Using styles, [461](#)

### Scripts

- Running a script from the host, [263](#)
- Running a script on the PC, [381](#)
- Running a script stored on a FTP server, [240](#)
- Running from the command line, [35](#)
- Running from the host, [221](#), [264](#)
- Running host scripts, [36](#)
- Running on a remote session, [405](#)
- Transferring and running using FTP, [241](#)
- Sending Keystroke to another application, [411](#)
- Serial Number, [413](#)
- Setting a global variable, [421](#)
- Setting a global variable from a list, [418](#)

---

Setting a session variable, [419](#)

Setting a value, [420](#)

Setting display effects, [204](#)

Setting the session title, [445](#)

Setting the value of a global variable, [415](#)

Setting up the host programs in another account, [422](#)

Sharing the host programs, [422](#)

Showing a menu options, [423](#)

Spooling a file to the host printer, [426](#)

## T

TCL Command Stacker, [444](#)

TCL Stack

Turning on, [439](#), [440](#)

Text Window

Closing, [448](#)

Directing output to, [459](#)

Message window, [451](#)

Opening, [452](#)

Pulling from a stack, [455](#)

Pushing on to a stack, [457](#)

Setting a footer, [450](#)

## V

Variables

Assigning a value, [18](#)

Getting a list, [250](#)

Getting the value, [249](#), [253](#)

Setting, [415](#), [421](#)

Setting from a list, [418](#)

Version, [463](#)

## W

WC, [12](#)

WIN.ACTIVATE, [13](#)

WIN.APP, [15](#)

WIN.APPDIR, [16](#)

WIN.ASCTOFT, [17](#)

WIN.ASSIGN, [18](#)

WIN.BARSUB, [19](#)

WIN.BKIMAGE, [21](#)

WIN.BOX, [22](#)

WIN.CBREAD, [24](#)

WIN.CBWRITE, [26](#)

WIN.CHECK, [28](#)

WIN.CHECK.UD, [28](#)

WIN.CHECK.UV, [28](#)

WIN.CLINFO, [30](#)

WIN.COLOR, [31](#)

WIN.COLOUR, [33](#)

WIN.COMLINE, [35](#)

WIN.COMSUB, [36](#)

WIN.CURSOR, [37](#)

WIN.DBANIM, [38](#)

WIN.DBATTACH, [40](#)

WIN.DBAXCTRL, [41](#)

WIN.DBBOX, [43](#)

WIN.DBBUTTON, [45](#)

WIN.DBCAPT, [47](#)

WIN.DBCHECK, [48](#)

WIN.DBCHILD, [50](#)

WIN.DBCOMBO, [52](#)

WIN.DBCTRL, [55](#)

WIN.DBDEL, [57](#)

WIN.DBDDTIME, [58](#)

WIN.DBEDGET, [60](#)

WIN.DBEDIT, [61](#)

WIN.DBEDSET, [63](#)

WIN.DBENABLE, 64  
WIN.DBENALL, 65  
WIN.DBEND, 67  
WIN.DBESTACK, 69  
WIN.DBEVENT, 70  
WIN.DBEVENT2, 73  
WIN.DBEVENTS, 75  
WIN.DBFETCH, 77  
WIN.DBFOCUS, 79  
WIN.DBGET, 81  
WIN.DBGETM, 83  
WIN.DBGETPRP, 85  
WIN.DBGRID, 86  
WIN.DBGROUP, 88  
WIN.DBHEADER, 90  
WIN.DBIMAGE, 92  
WIN.DBIMBUT, 94  
WIN.DBINIPRP, 96  
WIN.DBINIT, 97  
WIN.DBINPOK, 98  
WIN.DBLABEL, 100  
WIN.DBLIST, 103  
WIN.DBLISTVW, 105  
WIN.DBLOAD, 107  
WIN.DBMETHOD, 108  
WIN.DBMNSIZE, 111  
WIN.DBMOVE, 112  
WIN.DBMSGBOX, 113  
WIN.DBMSIZE, 115  
WIN.DBMVCTRL, 117  
WIN.DBNEVENT, 118  
WIN.DBNEW, 120  
WIN.DBOPTION, 123  
WIN.DBPANEL, 125  
WIN.DBPOST, 127  
WIN.DBPRGRES, 129  
WIN.DBRADIO, 131  
WIN.DBRECT, 133  
WIN.DBSCROLL, 135  
WIN.DBSELECT, 137  
WIN.DBSET, 139  
WIN.DBSETCOL, 141  
WIN.DBSETFNT, 142  
WIN.DBSETMNU, 143  
WIN.DBSETPRP, 144  
WIN.DBSHOW, 145  
WIN.DBSHOWPU, 147  
WIN.DBSTATUS, 149  
WIN.DBTAB, 151  
WIN.DBTABS, 153  
WIN.DBTEXT, 154  
WIN.DBTRACK, 156  
WIN.DBTREEVW, 158  
WIN.DBUNIT, 160  
WIN.DBUPDOWN, 162  
WIN.DDECLOSE, 164  
WIN.DDEEXEC, 166  
WIN.DDEEXEC2, 168  
WIN.DDEOPEN, 170  
WIN.DDEPOKE, 172  
WIN.DDEREQ, 174  
WIN.DDETIME, 176  
WIN.DISPLAY, 177  
WIN.DRARC, 178  
WIN.DRBRUSH, 180  
WIN.DRCHORD, 182  
WIN.DRELL, 184  
WIN.DRERASE, 186  
WIN.DRFONT, 188  
WIN.DRLINE, 190  
WIN.DRMOVE, 191  
WIN.DRPEN, 192  
WIN.DRPIE, 195  
WIN.DRPOLY, 197  
WIN.DRRECT, 199  
WIN.DRTEXT, 201  
WIN.EDIT, 203  
WIN.EFFECT, 204

---

WIN.EFILL, [206](#)  
WIN.EI, [208](#)  
WIN.EI2, [212](#)  
WIN.EVAL, [215](#)  
WIN.EXPORT, [216](#)  
WIN.FKEY, [219](#)  
WIN.FSCRIPT, [221](#)  
WIN.FTPCLOSE, [222](#)  
WIN.FTPCON, [223](#)  
WIN.FTPDEL, [224](#)  
WIN.FTPDIR, [225](#)  
WIN.FTPDISC, [226](#)  
WIN.FTPFILE, [227](#)  
WIN.FTPGET, [228](#)  
WIN.FTPGETDR, [229](#)  
WIN.FTPINFO, [230](#)  
WIN.FTPLIST, [231](#)  
WIN.FTPMKDIR, [233](#)  
WIN.FTPOPEN, [234](#)  
WIN.FTPPOS, [236](#)  
WIN.FTPPUT, [237](#)  
WIN.FTPREAD, [238](#)  
WIN.FTPRMDIR, [239](#)  
WIN.FTPSCR, [240](#)  
WIN.FTPSCRPT, [241](#)  
WIN.FTPSETDR, [243](#)  
WIN.FTPSETV, [244](#)  
WIN.FTPWRITE, [246](#)  
WIN.FTTOASC, [248](#)  
WIN.GETDATA, [249](#)  
WIN.GETLIST, [250](#)  
WIN.GETPARAM, [251](#)  
WIN.GETVAL, [252](#)  
WIN.GETVAR, [253](#)  
WIN.HELP, [254](#)  
WIN.HELPID, [255](#)  
WIN.HELPNAME, [256](#)  
WIN.HGLASS, [257](#)  
WIN.HOSTVER, [258](#)

WIN.HOTSPOT, [259](#)  
WIN.HOTSPOT2, [261](#)  
WIN.HSCRIPT, [263](#)  
WIN.HSCRIPTC, [264](#)  
WIN.ILADD, [265](#)  
WIN.ILCOUNT, [266](#)  
WIN.ILDELETE, [267](#)  
WIN.ILICON, [268](#)  
WIN.ILINFO, [269](#)  
WIN.ILISIZE, [270](#)  
WIN.ILLOAD, [271](#)  
WIN.ILNEW, [273](#)  
WIN.ILREMOVE, [275](#)  
WIN.IMAGE, [276](#)  
WIN.IMCHANGE, [278](#)  
WIN.IMCLOSE, [280](#)  
WIN.IMOPEN, [282](#)  
WIN.IMPORT, [284](#)  
WIN.INFOBOX, [288](#)  
WIN.INPBOX, [289](#)  
WIN.INVOKE, [290](#)  
WIN.LI, [291](#)  
WIN.LICINFO, [292](#)  
WIN.LOOKUP, [293](#)  
WIN.MENUATT, [295](#)  
WIN.MENUDEL, [297](#)  
WIN.MENUDET, [299](#)  
WIN.MENUIN, [301](#)  
WIN.MENULOAD, [303](#)  
WIN.MLADDR, [305](#)  
WIN.MLAVAIL, [307](#)  
WIN.MLDELETE, [308](#)  
WIN.MLFIND, [310](#)  
WIN.MLLOOKUP, [313](#)  
WIN.MLNEXT, [315](#)  
WIN.MLREAD, [318](#)  
WIN.MLSEND, [321](#)  
WIN.MOUSE, [324](#)  
WIN.MOUSEDEF, [326](#)



WIN.MOUSEIN, [327](#)  
WIN.MSGBOX, [329](#)  
WIN.MSTATE, [332](#)  
WIN.OBEXIST, [334](#)  
WIN.OBGET, [335](#)  
WIN.OBGETPRP, [337](#)  
WIN.OBMETHOD, [338](#)  
WIN.OBNEW, [340](#)  
WIN.OBREL, [341](#)  
WIN.OBSET, [342](#)  
WIN.OBSETPRP, [343](#)  
WIN.OBVTYPE, [344](#)  
WIN.PCBROWSE, [345](#)  
WIN.PCCLOSE, [347](#)  
WIN.PCCOPY, [348](#)  
WIN.PCCREATE, [349](#)  
WIN.PCDELETE, [351](#)  
WIN.PCDIR, [352](#)  
WIN.PCEDIT, [354](#)  
WIN.PCEOF, [356](#)  
WIN.PCFILE, [357](#)  
WIN.PCINFO, [359](#)  
WIN.PCLIST, [360](#)  
WIN.PCMKDIR, [362](#)  
WIN.PCMOVE, [364](#)  
WIN.PCOPEN, [365](#)  
WIN.PCPOS, [366](#)  
WIN.PCPRINT, [368](#)  
WIN.PCREAD, [369](#)  
WIN.PCREADAL, [371](#)  
WIN.PCREADLN, [373](#)  
WIN.PCRMDIR, [375](#)  
WIN.PCRUN, [376](#)  
WIN.PCRUN2, [378](#)  
WIN.PCSCRIPT, [381](#)  
WIN.PCWRITE, [382](#)  
WIN.PIESUB, [384](#)  
WIN.PLAYBACK, [387](#)  
WIN.POPUPIN, [389](#)  
WIN.PRINTOFF, [391](#)  
WIN.PRINTON, [392](#)  
WIN.PRTPAUSE, [393](#)  
WIN.RECOFF, [395](#)  
WIN.RECON, [397](#)  
WIN.RECPAUSE, [399](#)  
WIN.RSEXEC, [401](#)  
WIN.RSEXIST, [403](#)  
WIN.RSNAME, [404](#)  
WIN.RSSCRIPT, [405](#)  
WIN.RSSTART, [407](#)  
WIN.SCREEN, [409](#)  
WIN.SDUMP, [410](#)  
WIN.SENDKEYS, [411](#)  
WIN.SERIAL, [413](#)  
WIN.SERVER, [414](#)  
WIN.SETDATA, [415](#)  
WIN.SETEFFCT, [416](#)  
WIN.SETLIST, [418](#)  
WIN.SETPARAM, [419](#)  
WIN.SETVAL, [420](#)  
WIN.SETVAR, [421](#)  
WIN.SHARE, [422](#)  
WIN.SHOW, [423](#)  
WIN.SLOAD, [424](#)  
WIN.SPOOL, [426](#)  
WIN.SPULL, [428](#)  
WIN.SPUSH, [429](#)  
WIN.SREMOVE, [430](#)  
WIN.SRESTORE, [432](#)  
WIN.SSAVE, [433](#)  
WIN.SSTATE, [434](#)  
WIN.SSTORE, [436](#)  
WIN.STACK, [438](#)  
WIN.STACKOFF, [439](#)  
WIN.STACKON, [440](#)  
WIN.STATLINE, [441](#)  
WIN.TASK, [442](#)  
WIN.TCL, [444](#)

---

WIN.TITLE, [445](#)  
WIN.TRANSFER, [446](#)  
WIN.TWCLOSE, [448](#)  
WIN.TWFOOT, [450](#)  
WIN.TWMSG, [451](#)  
WIN.TWOPEN, [452](#)  
WIN.TWPULL, [455](#)  
WIN.TWPUSH, [457](#)  
WIN.TWUSE, [459](#)  
WIN.USESTYLE, [461](#)  
WIN.VERSION, [463](#)  
WIN.XLADDWS, [464](#)