

wIntegrate

Client Scripting 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

1. Reference - Script Commands	32
Elements of Syntax Statements	33
#	34
\$	35
&	37
&=	38
*	39
*=	40
+	41
+=	42
-	43
-=	44
/	45
/=	46
:	47
:=	48
<	49
<=	50
=	51
>	52
>=	53
Abs	54
Activate	55
AddCheckBox	57
AddComboBox	58
AddGap	59
AddIcon	60
AddLargelcon	62
AddLargeSeparator	64
AddLargeStylelcon	65
AddSeparator	67
AddStylelcon	68
AddTool	70

AddToolTip.....	72
AddWideIcon.....	73
AddWideStyleIcon.....	74
AlwaysOnTop.....	76
And.....	77
AppFolder	78
Asc	80
AsciiToBS	82
AsciiToFT	83
AsciiToHex.....	84
Attribute.....	85
AutoWrapOn	88
BaseBar	89
Blue.....	90
BSToAscii	92
Capture AddData	93
Capture Continue.....	94
Capture Delete.....	95
Capture Off	97
Capture On	99
Capture Pause	103
CaptureInfo	105
CaptureText	107
Chain.....	109
Change	111
ChangeUserSession	113
Char	116
CheckServer	118
ChooseColor	119
ChooseFont	121
Clipboard Read	124
Clipboard Write	126
CloseServer	128
Color	129
CommandBar Add	131
CommandBar OnRightClick.....	133
CommandBar Remove	134
CommandBar Replace.....	135
CommandBar SetScript	136

CommandBar Show	137
CommandBarInfo	138
CommandBarList	140
ConfigFile Read	142
ConfigFile ReadInteger	144
ConfigFile ReadSection	146
ConfigFile Remove	148
ConfigFile Write	149
Configure	151
Convert	153
Count	154
CreatelconBar	155
CreateToolBar	157
Cursor	159
CursorShape	161
Date	163
DCount	164
DDE Advise	166
DDE Execute	169
DDE HostLink	171
DDE Initiate	173
DDE Poke	175
DDE Terminate	177
DDE Timeout	179
DDE Unadvise	180
DDEChanged	183
DDERequest	184
Debug	185
Define	187
Del	189
Destroy	191
Dialog	193
DialogBox AddToList	195
DialogBox Animate	196
DialogBox AxControl	198
DialogBox Background	200
DialogBox BaseBar	201
DialogBox Caption	202
DialogBox CheckBox	203

DialogBox CheckListBox.....	205
DialogBox ColorButton.....	207
DialogBox ComboBox.....	209
DialogBox Control	211
DialogBox ControlCommand	214
DialogBox Create.....	216
DialogBox CText.....	218
DialogBox DateTime	220
DialogBox Default	222
DialogBox DefPushButton	223
DialogBox Delete	225
DialogBox DeleteFromList	227
DialogBox EditText	228
DialogBox Enable	230
DialogBox EnableAll	232
DialogBox End	234
DialogBox EndCreate	236
DialogBox EventLibrary	238
DialogBox ExStyle	239
DialogBox Font	241
DialogBox Graphic.....	242
DialogBox GraphicButton.....	244
DialogBox Grid.....	246
DialogBox GroupBox	247
DialogBox Header.....	249
DialogBox Icon.....	250
DialogBox IconBar	252
DialogBox InitCommand	253
DialogBox InputOK	254
DialogBox LimitText.....	256
DialogBox ListBox.....	258
DialogBox ListView	260
DialogBox LoseFocus	261
DialogBox LText.....	263
DialogBox MaxSize.....	265
DialogBox MinIcon	267
DialogBox MinSize.....	269
DialogBox Move.....	271
DialogBox MoveControl	272

DialogBox New	274
DialogBox NextControl.....	276
DialogBox OnDialogEvent	277
DialogBox OnEvent.....	280
DialogBox Option	281
DialogBox Panel	284
DialogBox PopupMenu	286
DialogBox PopupWindow	289
DialogBox ProgressBar.....	291
DialogBox PushButton	292
DialogBox RadioButton.....	294
DialogBox Reply	296
DialogBox ResetControls.....	297
DialogBox RText.....	299
DialogBox ScrollBar	301
DialogBox ScrollStep	303
DialogBox Select.....	305
DialogBox SetColor.....	306
DialogBox SetFont.....	308
DialogBox SetKey	310
DialogBox SetMenu	312
DialogBox SetRange.....	313
DialogBox SetTabs	315
DialogBox SetText	317
DialogBox Show.....	319
DialogBox ShowControl.....	321
DialogBox Size.....	322
DialogBox Style.....	324
DialogBox TabControl.....	326
DialogBox ToolTip.....	327
DialogBox Trackbar	328
DialogBox TreeView	330
DialogBox UpDownButton	332
DialogBox Validate.....	333
DialogBox Variable	335
DialogBox WebStyleName.....	336
DialogBox Window.....	338
DialogBox WindowState	339
DialogIndex	341

DialogList	342
DialogListFind	343
DialogNextEvent	345
DialogRect	347
DialogUnit	349
DialogWindowState	352
DirExist	354
Display At	355
Display Attribute	357
Display Box	359
Display CharFill	361
Display Command	363
Display Direct	366
Display Effect	367
Display EffectFill	369
Display Text	372
Display UpdateOff	373
Display UpdateOn	375
DisplayOn	376
Draw Arc	377
Draw Brush	379
Draw Chord	381
Draw CopyTo	383
Draw Ellipse	385
Draw Erase	387
Draw Font	390
Draw Image	392
Draw Line	394
Draw Move	396
Draw Pen	397
Draw Pie	400
Draw Polygon	402
Draw Rect	404
Draw Size	406
Draw Text	407
Draw To	409
EditInput	411
EditInput2	415
EditInputInfo	419

Effect.....	421
Else	423
EndCreateIconBar	424
EndIf.....	425
EndOfFile	426
EndScript	428
EndSub	429
EndWith	430
Enter	431
Event Delete	432
Event OnExit.....	433
Event OnHostText.....	435
Event OnPortClose	437
Event OnPortOpen.....	438
Event OnRestore	439
ExecAccess	440
Execute	441
Extract.....	443
Field	445
Fields	447
File Close	448
File Copy.....	450
File Create	452
File CreateDir.....	454
File Delete.....	456
File DeleteDir	458
File Move	460
File Open	462
File Pointer.....	464
File Read.....	465
File ReadLine.....	468
File Write.....	470
FileBrowse	473
FileExist	475
FileInfo	476
FileList.....	478
Files	480
FileTempName	481
FindHWnd	482

FolderBrowse	483
FontInfo	485
Ftp Close	487
Ftp Connect	488
Ftp CreateDir	490
Ftp Delete	491
Ftp DeleteDir	492
Ftp Disconnect	493
Ftp Get	494
Ftp Move	496
Ftp Open	498
Ftp Pointer	500
Ftp Put	502
Ftp Read	504
Ftp SetDir	506
Ftp Write	507
FtpDirExist	509
FtpExist	511
FtpGetDir	513
FtpInfo	514
FtpList	517
FTToAscii	519
Get	520
GetFocus	521
GetFocusHWnd	523
GetFromHost Next	524
GetFromHost RawData	526
GetIconColumn	528
GetMachineType	529
Global	530
Green	532
HCall	533
HDelete	534
Help TopicId	535
Help TopicName	536
HexToAscii	537
Host Break	538
Host Get	539
Host Send	540

HotSpot Add.....	541
HotSpot AutoDelete	544
HotSpot Delete.....	545
HotSpot DeleteAll.....	546
HotSpot Display	547
HotSpot Off	549
HotSpot On	550
HourGlass Off	551
HourGlass On	552
HourGlass Smash.....	553
HRead.....	554
HReadU	556
HReadV	557
HReadVU.....	559
HRelease	560
HWnd	561
HWrite	562
HWriteU	564
HWriteV.....	565
HWriteVU	566
IconBar.....	567
Iconv.....	568
If	571
ImageList Add	573
ImageList AddIcon	575
ImageList Delete	576
ImageList GetIconSize.....	577
ImageList GetImageCount.....	578
ImageList GetImageInfo.....	579
ImageList Load	580
ImageList New	582
ImageList Remove	583
Index	584
InfoBox.....	586
Ins	588
InString.....	590
Invoke	592
IsApp.....	594
IsCommandBar	595

IsCommandBarShown	596
IsControl.....	597
IsDialog	598
IsEvent	599
IsFile	600
IsFullPath	601
IsMenu	602
IsMenuItem	603
IsNumber	605
IsObject.....	606
IsOnMenu	607
IsRemotePath	608
IsShown	609
IsTask	610
IsTrigger.....	612
IsVScreen	613
Items	614
Key ClearAll	615
Key ClearSet.....	616
Key LoadEmulation.....	618
Key Run	619
KeyboardLocked	620
KeyIn.....	621
LastErrorText	623
Left.....	625
Length	626
Library	627
LicenseInfo.....	629
Local	630
LocalMode	632
LocalPrint	634
Locate	635
Loop	638
Mail Address	640
Mail Delete	642
Mail LookUp	644
Mail Read.....	646
Mail Send	649
MailAvailable.....	651

MailFindAll	652
MailFindNext	655
MakeCommandBar	658
MapChars	659
Match	660
Menu AddItem	662
Menu Attach	664
Menu Command	666
Menu Create	667
Menu Delete	669
Menu DeleteItem	671
Menu Detach	672
Menu Enable	673
Menu EndCreate	674
Menu Help	676
Menu Main	677
Menu New	679
Menu OnClose	680
Menu OnOpen	681
Menu PopUp	682
Menu Separator	684
Menu SetCheck	685
MenuChecked	686
MenuEnabled	688
MenuGUI Active	689
MenuGUI Command	690
MenuGUI DefaultMap	691
MenuGUI GetScript	693
MenuGUI Initialise	694
MenuGUI Load	695
MenuGUI LoadDefault	696
MenuGUI LoadEmulationKeys	697
MenuGUI MapVariable	699
MenuGUI Save	701
MenuGUI SetupComms	702
MenuGUI Update	703
MenuList	704
MessageBox	706
Mid	709

Mouse	711
MultiExec	713
MultiScript	715
NextRow	717
Not	718
Notify	719
Object Enum	720
Object Get	722
Object New	724
Object Release	726
Object Set	727
OConv	730
OpenServer	735
Or	736
PathAddExt	737
PathAppend	739
PathComponent	741
Play Abort	743
Play Start	744
Play Stop	745
PlayInfo	746
PopupCalculator	747
PortInfo	749
Print	751
PrinterInfo	752
Random	754
Receive	756
Red	757
Rep	758
Repeat	760
ResolveEffect	761
Return	763
RGB	764
Right	765
Run	766
Screen CopyPage	769
Screen Load	770
Screen Off	772
Screen On	774

Screen Remove	776
Screen Restore	778
Screen Save	780
Screen ScrollRegion	782
Screen Store	784
ScreenOn	786
ScreenText	787
ScreenWord	789
Script	791
ScriptError End	792
ScriptError EndAll	793
ScriptError Exit	794
ScriptError Ignore	795
ScriptError ReplaceLine	796
ScriptError RestartScript	797
ScriptError Skip	798
ScriptError Step	799
ScriptErrorInfo	800
ScriptGlobalInfo	802
ScriptInfo	803
ScriptVersion	807
Select Area	808
Select Clear	810
Select End	811
Select Extend	813
Select Keyboard	815
Select Start	816
SelectInfo	818
SendKeys	820
SendMessage	823
ServerErrMsg	826
Session Enable	827
Session Execute	828
Session LockSize	829
Session Move	830
Session Script	832
Session Show	833
Session Size	834
SessionInfo	835

SessionName.....	837
Sessions	838
Set.....	840
SetFocus.....	842
SetIconColumn	844
Show	845
ShowURL.....	847
Sound Loop.....	849
Sound Play.....	851
Sound PlaySystem.....	853
Sound Stop	855
Stamp.....	856
StopList	857
Store	858
StoreAccess.....	860
String.....	861
Strip.....	862
Sub.....	863
SystemColor	865
SystemFolder.....	868
SystemFont.....	869
SystemMetrics	871
T.....	873
Time	874
ToolSpace.....	875
Translation Load	877
Translation Unload.....	879
Translation Use.....	880
Trigger Add	881
Trigger Delete	884
Trigger DeleteAll	885
Trim.....	886
Type	888
Until.....	890
Update	891
UserGlobal	893
VarType	895
VirtualScreen Background	897
VirtualScreen Delete.....	899

VirtualScreen DeleteAll	900
VirtualScreen MoveWindow	901
VirtualScreen New	903
VirtualScreen Pan	905
VirtualScreen PrintOff	906
VirtualScreen PrintOn	907
VirtualScreen Resize	909
VirtualScreen ResizeWindow	910
VirtualScreen Restore	912
VirtualScreen Scroll	914
VirtualScreen Store	916
VirtualScreen Window	918
VSAttribute	920
VSCursor	922
VSInfo	923
VSText	925
Wait Delay	926
Wait DuringTask	927
Wait ForText	928
Wait RxEmpty	930
Wait TxEmpty	932
WebStyle ControlType	933
WebStyle Global	936
WebStyle Menu	938
While	940
WindowPos	941
With	943
WorkArea	945
Yield	947
.....	948
=	949
2. Reference - Script Controls	950
Animate	951
Calculator	954
CheckBox	956
CheckListBox	959
ColorButton	964
ComboBox	967
DateTime	972

Divider.....	975
Draw.....	977
EditText.....	978
Graphic.....	983
GraphicButton.....	985
Grid.....	987
GroupBox.....	996
Header.....	998
Icon.....	1001
ListBox.....	1002
ListView.....	1007
ProgressBar.....	1014
PushButton.....	1016
RadioButton.....	1019
Rectangle.....	1023
ScrollBar.....	1025
TabControl.....	1027
Text.....	1031
TrackBar.....	1033
TreeView.....	1037
UpDownButton.....	1042
Properties.....	1044
AnchorIndex.....	1045
BackColor.....	1046
BackStyle.....	1047
Base.....	1048
BorderStyle.....	1049
Buddy.....	1050
Caption.....	1051
CaretIndex.....	1052
CellValue.....	1053
Check.....	1054
Check.....	1055
CheckIndex.....	1056
CheckList.....	1057
CheckText.....	1058
ColHeaders.....	1059
ColText.....	1060
ColWidths.....	1061

Column	1062
ColumnInfo	1063
ColumnWidth	1064
Columns.....	1065
CustomFormat	1066
Data	1068
DateMax	1069
DateMin	1070
DecimalPlaces	1071
Default	1072
Display	1073
DragData	1074
DragMode	1075
DragTarget.....	1076
Dropped	1077
Dropped	1078
Enabled.....	1079
EnterAction	1080
FastEntry	1081
File	1082
FloatingCells	1083
FocusItem	1084
Font.....	1085
ForeColor.....	1086
FormatType	1087
Height	1088
Horizontal.....	1089
HorizontalExtent	1090
Image.....	1091
Image.....	1092
Image.....	1093
ImageHeight	1094
ImageWidth.....	1095
Indent.....	1096
LabelWidths	1097
Labels	1098
Labels	1099
Left.....	1100
LimitText	1101

LineSize	1102
List	1103
MaximumPosition	1104
MinimumPosition	1105
Modified	1106
NoEdit	1107
Optional	1108
PageSize	1109
PasswordChar	1110
Position	1111
Position	1112
Position	1113
Position	1114
Radio	1115
Range	1116
Range	1117
Range	1118
RangeMax	1119
RangeMin	1120
ReadOnly	1121
Redraw	1122
Row	1123
RowHeaders	1124
RowHeights	1125
RowText	1126
Rows	1127
SelEnd	1128
SelIndex	1129
SelIndex	1130
SelStart	1131
SelText	1132
SelectMode	1133
SelectedItem	1134
SortColOnDbClick	1135
SortRowOnDbClick	1136
Step	1137
TabStops	1138
TabStops	1139
Text	1140

TextBackColor	1141
ThumbLength.....	1142
TickFrequency	1143
Top.....	1144
TopIndex.....	1145
TrackHeight	1146
TrackWidth.....	1147
UpdateMode	1148
UpdateMode	1149
UpdateMode	1150
ValidMode.....	1151
Value.....	1152
Value.....	1153
View	1154
Visible	1155
WebStyle	1156
Width.....	1157
Methods	1158
AddLine.....	1159
AddLine.....	1160
AdjustRect	1161
Arrange	1162
CharFromPos	1164
CheckAll.....	1165
Clear	1166
ClearRange.....	1167
ClearSel	1168
ClearSel	1169
ClearTicks.....	1170
Close.....	1171
Copy	1172
Count	1173
Count	1174
Count	1175
Count	1176
Count	1177
Count	1178
Cut	1179
DeleteAllItems.....	1180

DeleteAllItems.....	1181
DeleteAllItems.....	1182
DeleteCols	1183
DeleteColumn	1184
Deleteltem	1185
Deleteltem	1186
Deleteltem	1187
Deleteltem	1188
DeleteLine.....	1189
DeleteLine.....	1190
DeleteRows	1191
DeltaPos	1192
EditLabel.....	1193
EditLabel.....	1194
EditSelect.....	1195
EditSelectInfo.....	1196
Empty.....	1197
Empty.....	1198
EnsureVisible.....	1199
EnsureVisible.....	1200
Expand.....	1201
FillRange.....	1202
Find.....	1203
Find.....	1204
Find.....	1206
FindExact.....	1207
FindExact.....	1208
FindItem.....	1209
FirstVisibleLine	1211
GetCellAttribute	1212
GetCellAutoSize	1213
GetCellBackColor	1214
GetCellBorder	1215
GetCellControl	1216
GetCellEnabled.....	1217
GetCellFloat.....	1218
GetCellFont.....	1219
GetCellForeColor.....	1220
GetCellJustify.....	1221

GetCellList	1222
GetCellReadOnly	1223
GetCellValue.....	1224
GetCheck.....	1225
GetColWidth	1226
GetColumn.....	1227
GetColumnWidth	1229
GetCountPerPage	1230
GetCurrentCell.....	1231
GetCurrentText	1232
GetItem	1233
GetItem	1235
GetItem	1237
GetItemData	1240
GetItemData	1241
GetItemData	1242
GetItemImage	1243
GetItemPosition	1244
GetItemState.....	1245
GetItemState.....	1247
GetItemText.....	1249
GetItemText.....	1250
GetItemText.....	1251
GetItemText.....	1252
GetItemWidth.....	1253
GetNextItem.....	1254
GetNextItem.....	1256
GetOrigin	1258
GetRangeAttribute	1259
GetRangeAutoSize	1260
GetRangeBackColor.....	1261
GetRangeBorder.....	1262
GetRangeControl.....	1263
GetRangeEnabled	1264
GetRangeFloat	1265
GetRangeForeColor	1266
GetRangeJustify	1267
GetRangeList.....	1268
GetRangeReadOnly	1269

GetRangeValue	1270
GetRowHeight	1271
GetSel	1272
GetSelectedCount	1273
GetTick	1274
GetTopIndex	1275
HideCols	1276
HideRows	1277
InitStorage	1278
InitStorage	1279
InsPosFromPoint	1280
InsertCols	1281
InsertColumn	1282
InsertItem	1284
InsertItem	1286
InsertItem	1289
InsertItemText	1292
InsertItemText	1293
InsertLine	1294
InsertLine	1295
InsertLine	1296
InsertRows	1298
ItemFromPoint	1299
ItemFromPoint	1300
Line	1301
Line	1302
Line	1303
LineCount	1304
LineFromChar	1305
LineIndex	1306
LineLength	1307
LineScroll	1308
Open	1309
Paste	1310
Play	1311
ReplaceSel	1312
ResizeColToFit	1313
ResizeRowToFit	1314
RowCount	1315

Scroll.....	1316
ScrollCaret.....	1317
ScrollTo.....	1318
ScrollToCell	1319
Seek.....	1320
SelCount.....	1321
SelLine.....	1322
SelLine.....	1323
SelRange.....	1324
Select.....	1325
Select.....	1326
SelectAll.....	1327
SelectCell.....	1328
SelectInfo.....	1329
SelectInfo.....	1330
SelectItem.....	1331
SelectRange	1332
SetCellAttribute.....	1333
SetCellAutoSize.....	1335
SetCellBackColor.....	1336
SetCellBorder	1337
SetCellControl.....	1338
SetCellEnabled	1340
SetCellFloat	1341
SetCellFont.....	1342
SetCellForeColor	1344
SetCellJustify	1345
SetCellList.....	1346
SetCellReadOnly	1347
SetCellValue	1348
SetCheck	1349
SetColWidth.....	1350
SetColumn.....	1351
SetColumnWidth.....	1353
SetCurrentCell	1354
SetCurrentText	1355
SetImageList.....	1356
SetImageList.....	1357
SetItem	1358

SetItem	1360
SetItem	1362
SetItemCount.....	1365
SetItemData.....	1366
SetItemData.....	1367
SetItemData.....	1368
SetItemImage	1369
SetItemPosition.....	1370
SetItemSize	1371
SetItemState	1372
SetItemState	1374
SetItemText	1376
SetItemText	1377
SetItemText	1378
SetItemText	1379
SetItemWidth	1380
SetPadding	1381
SetRangeAttribute	1382
SetRangeAutoSize	1383
SetRangeBackColor	1384
SetRangeBorder	1385
SetRangeControl	1386
SetRangeEnabled.....	1388
SetRangeFloat.....	1389
SetRangeFont.....	1390
SetRangeForeColor.....	1392
SetRangeJustify.....	1393
SetRangeList	1394
SetRangeReadOnly.....	1395
SetRangeValue.....	1396
SetRowHeight.....	1397
SetSel	1398
SetTick.....	1399
SortChildren.....	1400
SortCols	1401
SortRows	1403
StepIt	1405
Stop	1406
TickCount.....	1407

Undo	1408
Update	1409
VisibleCount	1410
Events	1411
Activate	1412
BeginLabelEdit	1413
BeginLabelEdit	1414
BeginTrack	1415
ButtonClick	1416
CellClick	1417
CellRightClick	1418
Change	1419
Changed	1420
Click	1421
Click	1422
CloseUp	1423
ColumnClick	1424
CurrentCell	1425
DblClick	1426
DblClick	1427
Deactivate	1428
DividerDblClick	1429
Drag	1430
DragEnd	1431
DragEnded	1433
DragEnter	1434
DragLeave	1436
DragOver	1437
DragStart	1439
DragStarted	1441
Drop	1442
DropDown	1444
EditChange	1445
EditUpdate	1446
EndEditing	1447
EndLabelEdit	1448
EndLabelEdit	1449
EndTrack	1450
EqualsPressed	1451

ErrSpace.....	1452
ErrSpace.....	1453
HScroll.....	1454
ItemChanged.....	1455
ItemChanging.....	1458
ItemClick.....	1461
ItemDbClick.....	1462
ItemExpanded.....	1463
ItemExpanding.....	1465
KeyDown.....	1467
KeyUp.....	1468
KillFocus.....	1469
MaxText.....	1470
ModifyCell.....	1471
OutOfMemory.....	1472
Return.....	1473
RightClick.....	1474
RightClick.....	1475
RightDbClick.....	1476
RightDbClick.....	1477
Scroll.....	1478
SelCancel.....	1479
SelChange.....	1480
SelChange.....	1481
SelChanged.....	1482
SelChanged.....	1483
SelChanging.....	1484
SelChanging.....	1485
SelEndCancel.....	1486
SelEndOK.....	1487
SetFocus.....	1488
Start.....	1489
StartEditing.....	1490
Stop.....	1491
Track.....	1492
Update.....	1493
VScroll.....	1494
ValidateCell.....	1495

3. Reference - Script Menu Options.....1496

EditCopy	1497
EditCopySpecial.....	1498
EditCopyTo	1499
EditPaste.....	1501
EditPlay	1502
EditRecord	1503
EditSearch	1505
EditSelectAll.....	1506
EditSelectScreen	1507
EditSelectWindow	1508
FileAnother.....	1509
FileDelete.....	1510
FileEdit	1511
FileExit	1512
FileExitAll	1513
FileOpen	1514
FilePrint.....	1515
FilePrinterSetup	1516
FileSaveAs.....	1517
HelpAbout	1518
HelpIndex.....	1519
HelpOnHelp	1520
HelpStatus	1521
HelpSupportLog	1522
RunBridgeCopy.....	1523
RunClearBackPages.....	1525
RunExportFile	1526
RunImportFile	1528
RunKermitFinish	1531
RunKermitGetFile	1532
RunKermitSendFile.....	1533
RunKermitServer	1534
RunProgram.....	1535
RunReceiveFile.....	1536
RunResizeWindow.....	1538
RunRestartPort	1539
RunScript	1540
RunSendFile	1541
RunSpoolFile	1543

ScriptInfo	1545
ScrollAcrossWidth	1546
ScrollBackWidth	1547
ScrollDownLine	1548
ScrollEndPage	1549
ScrollLeftColumn	1550
ScrollNextPage	1551
ScrollPrevPage	1552
ScrollRightColumn	1553
ScrollTopPage	1554
ScrollUpLine	1555
SetupApplication	1556
SetupCharacter	1558
SetupColors	1559
SetupCommunications	1562
SetupCommunications (PicLan)	1563
SetupCommunications (Serial)	1564
SetupCommunications (Windows Sockets)	1566
SetupEditKeys	1568
SetupFunctionKeys	1569
SetupMouse	1573
SetupPreferences	1574
SetupTerminal	1577
Appendix A. Scripting Libraries and Modules	1579
Loading Libraries into Memory	1581
Icon Library	1582
Tool Library	1590
Function Key Bar Library	1596
Server Library	1599
Capture Module	1604
Date and Time Module	1607
DDE Module	1608
Dialog Box Module	1611
Display Module	1618
Draw Module	1621
File Module	1625
HotSpot Module	1629
Menus Module	1632
Play Module	1637

Screen Module	1638
Select Module	1641
Session Module	1644
Session Variables Module	1645
String Manipulation Module	1646
Wait Module	1648
Appendix B. Using the Dialog Designer.....	1649
Dialog Designer	1650
Understanding the Toolbar	1652
Understanding the Toolbox.....	1654
Understanding the Status Bar.....	1660
Creating a Dialog Box	1661
Appendix C. The Script Monitor.....	1696
The Script Error Box	1697
The Script Monitor.....	1698
Appendix D. Script Constants.....	1702
Application Constants	1704
Boolean Constants.....	1705
Character Constants	1706
Control Styles.....	1707
Environment Constants.....	1708
Multiple Command Constants.....	1709
Script Constants.....	1713
Specific Commands	1714
Windows Styles.....	1715
Appendix E. Backslash format strings.....	1716
Appendix F. The Script Compiler.....	1717
Index	1718

Reference - Script Commands

This chapter is a reference for all the wIntegrate script commands that you can use to write complete wIntegrate scripts. They are listed alphabetically, and each includes syntax, description, parameters and examples. Many of them also include a list of related script commands to refer to for more information.

Elements of Syntax Statements

This reference manual uses a common method for stating syntax for wIntegrate commands and functions. The syntax statement includes the command or function name, required arguments, and options that you can use with the command. Italics represents a variable that you can replace with any valid option. The following figure illustrates the elements of a syntax statement:

COMMAND *required* [*option*] [*option1* | *option2*] {*option1* | *option2*} *required*... "string"

Element	Usage
bold	Command names appear in bold face
<i>italic</i>	The parameters for the command are in italic. No brackets or braces indicate a required argument.
[]	Square brackets indicate an optional argument
	A vertical line indicate that you may choose between the given arguments
{ }	Braces indicate that you must choose between the given arguments
...	An ellipsis indicates that you may enter more than one argument
"string"	Quotation marks must enclose a literal string.

#

Syntax

expr1 # expr2

Synonyms



Description

The # (not equal) operator returns true if two expressions are not the same.

If both expressions evaluate to a number a numeric comparison is used, otherwise a case-insensitive string compare is used.

Parameters

The following table describes the parameters of the # command:

Parameter	Description
<i>expr1</i>	The first expression for the compare
<i>expr2</i>	The second expression for the compare

\$

Syntax

\$ variable

Description

The \$ (alias) operator is used to substitute the value of a variable at any point that the script language requires a name (e.g. command handles, variable names, property names).

It's use enables the substitution of a value at run time.

Note: There must be no spaces between the \$ symbol and the first letter of the variable name.

Parameters

The following table describes the parameters of the \$ command:

Parameter	Description
<i>variable</i>	The name of the variable that contains the value to be used.

Example

Indirectly set the value of variable b.

```
* Set b to 1
b = 1
* Set a to the name of our variable "b"
a = "b"
* Set b to 2 using an alias
$a = 2
```

The following code extract from "wintsys\lib\fkeylib.wis" uses aliases to specify the name of the dialog box and also the name of each button control in the dialog box

.
.
.

```
DialogBox Create $name 0,100,400,size
Style WS_CHILD
Font 8,"helv"
Loop
    j+=1
    but = "F":j
    but_text = Field(text,'|',j)
    If but_text = "" Then but_text = but
    Pushbutton but_text,$but,col,row,but_width,row_size
    ControlCommand $but,"Key Run ":"but:\""
    col += but_width + but_gap
    If (j&3) = 0 then col+=2
Until j = no_keys
Repeat
.
.
.
```

&

Syntax

expr1 & expr2

Description

The & operator returns the "binary and" value of the two numbers.

It compares each bit of the number and sets the corresponding bit in the result if both bits are set.

This operator is frequently used to check a flag value.

Note: The script language only understands integer numbers from -2147483648 to 2147483647.

Parameters

The following table describes the parameters of the & command:

Parameter	Description
<i>expr1</i>	An expression that evaluates to a number
<i>expr2</i>	An expression that evaluates to a number

&=

Syntax

variable **&=** *value*

Description

The **&=** assignment uses the "binary and" operator to modify a variable.

It is equivalent to: *variable* = *variable* & *number*

Note: The script language only understands integer numbers from -2147483648 to 2147483647.

Parameters

The following table describes the parameters of the **&=** command:

Parameter	Description
<i>variable</i>	The name of the variable to modify
<i>number</i>	The number to combine with the variable current value

Syntax

*{expr1 * expr2 / * [comment] }*

Description

The asterisk symbol is used to multiply two numbers or to start a comment:

When placed between two numbers in an expression it will multiply the numbers together and return the result.

Note: The script language only understands integer numbers from -2147483648 to 2147483647.

When placed at the start of a statement it treats the rest of the line (i.e. up until the next carriage return) as a comment to be ignored when the script runs.

If the comment line starts with "\$Version" the comment can be read from the script with the ScriptVersion function.

Parameters

The following table describes the parameters of the * command:

Parameter	Description
<i>expr1</i>	An expression that evaluates to a number
<i>expr2</i>	An expression that evaluates to a number
<i>comment</i>	The text for the comment

Related Script Commands

[ScriptVersion](#)

*=

Syntax

*variable *= number*

Description

The *= assignment operator multiplies the variable by the given value and updates the original variable.

It is equivalent to: `variable = variable * value`

Note: The script language only understands integer numbers from -2147483648 to 2147483647.

Parameters

The following table describes the parameters of the *= command:

Parameter	Description
<i>variable</i>	The name of the variable to modify
<i>number</i>	The number to multiply the variable by.

+

Syntax

expr1 + expr2

Description

The + (add) operator adds two numbers and returns the result.

Note: The script language only understands integer numbers from -2147483648 to 2147483647.

Parameters

The following table describes the parameters of the + command:

Parameter	Description
<i>expr1</i>	An expression that evaluates to a number
<i>expr2</i>	An expression that evaluates to a number

+=

Syntax

variable += value

Description

The += assignment adds the specified number to the variable.

It is equivalent to: `variable = variable + number`

Note: The script language only understands integer numbers from -2147483648 to 2147483647.

Parameters

The following table describes the parameters of the += command:

Parameter	Description
<i>variable</i>	The name of the variable to modify
<i>number</i>	The number to add to the variable

-

Syntax

[expr1] - expr2

Description

The - (minus) operator is used to subtract one number from another and return the result.

If the first expression is omitted it will reverse the sign of the second expression (equivalent of subtracting the second expression from 0).

Note: The script language only understands integer numbers from -2147483648 to 2147483647.

Parameters

The following table describes the parameters of the - command:

Parameter	Description
<i>expr1</i>	An expression that evaluates to a number. If omitted it is equivalent of this evaluating to 0.
<i>expr2</i>	An expression evaluating to a number that will be subtracted from <i>expr1</i> .

-=

Syntax

variable -= value

Description

The -= assignment subtracts the specified number from the variable.

It is equivalent to: `variable = variable - number`

Note: The script language only understands integer numbers from -2147483648 to 2147483647.

Parameters

The following table describes the parameters of the -= command:

Parameter	Description
<i>variable</i>	The name of the variable to modify
<i>number</i>	The number to subtract from the variable

/

Syntax

expr1 / expr2

Description

The / (divide) operator divides two numbers and returns the result.

If the second number is 0 an error will occur and the script error dialog box will be shown.

If the result of the division would not be an exact integer the fractional part of the calculation is discarded.

Note: The script language only understands integer numbers from -2147483648 to 2147483647.

Parameters

The following table describes the parameters of the / command:

Parameter	Description
<i>expr1</i>	An expression that evaluates to an integer to be divided
<i>expr2</i>	An expression that evaluates to an integer to be divided into <i>expr1</i>

/=

Syntax

variable /= value

Description

The /= assignment divides the specified number into the variable.

It is equivalent to: `variable = variable / number`

If the number is 0 a script error will be reported.

If the result of the division would not be an exact integer the fractional part of the calculation is discarded.

Note: The script language only understands integer numbers from -2147483648 to 2147483647.

Parameters

The following table describes the parameters of the /= command:

Parameter	Description
<i>variable</i>	The name of the variable to modify
<i>number</i>	The number to divide into the variable

:

Syntax

expr1 : expr2

Description

The : (cat) operator concatenates two strings and returns the result.

Parameters

The following table describes the parameters of the : command:

Parameter	Description
<i>expr1</i>	String for the start of the result
<i>expr2</i>	String for the end of the result

:=

Syntax

variable := *string*

Description

The := assignment operator concatenates a string onto the end of the specified variable.

It is equivalent of: `variable = variable : string`

Parameters

The following table describes the parameters of the := command:

Parameter	Description
<i>variable</i>	The name of the variable to modify
<i>string</i>	The string to append to the variable

<

Syntax

expr1 < expr2

Description

The < (less than) operator compares two expressions and returns true if the first expression is less than the second expression.

If both expressions evaluate to a number a numeric comparison is used, otherwise a case insensitive string comparison is used.

Parameters

The following table describes the parameters of the < command:

Parameter	Description
<i>expr1</i>	First expression for the comparison
<i>expr2</i>	Second expression for the comparison

<=

Syntax

expr1 <= *expr2*

Description

The <= (less than or equal) operator compares two expressions and returns true if the first expression is less than or equal to the second expression.

If both expressions evaluate to a number a numeric comparison is used, otherwise a case insensitive string comparison is used.

Parameters

The following table describes the parameters of the <= command:

Parameter	Description
<i>expr1</i>	First expression for the comparison
<i>expr2</i>	Second expression for the comparison

=

Syntax

expr1 = expr2

Description

The = (equals) operator is used in two ways:

When used as the second word in a statement, the *expr1* is a variable name and the variable is assigned with the result of the *expr2*.

When used as part of an expression it compares the two expressions and returns true if they are the same and false if they are different.

String comparison's are case insensitive. e.g. "aBc" = "ABC" = "abc".

Parameters

The following table describes the parameters of the = command:

Parameter	Description
<i>expr1</i>	The variable name when used as an assignment operator or the first expression to compare.
<i>expr2</i>	The new value for the variable when used as an assignment, otherwise the second expression for the comparison.



Syntax

expr1 > expr2

Description

The > (greater than) operator compares two expressions and returns true if the first expression is greater than the second expression.

If both expressions evaluate to a number a numeric comparison is used, otherwise a case insensitive string comparison is used.

Parameters

The following table describes the parameters of the > command:

Parameter	Description
<i>expr1</i>	First expression for the comparison
<i>expr2</i>	Second expression for the comparison

>=

Syntax

expr1 >= *expr2*

Description

The >= (greater than or equal) operator compares two expressions and returns true if the first expression is greater than or equal to the second expression.

If both expressions evaluate to a number a numeric comparison is used, otherwise a case insensitive string comparison is used.

Parameters

The following table describes the parameters of the >= command:

Parameter	Description
<i>expr1</i>	First expression for the comparison
<i>expr2</i>	Second expression for the comparison

Abs

Syntax

Abs number

Description

This gives the absolute value of a number. i.e. it removes the minus sign on negative numbers.

Note: This command works with whole numbers (integers).

Parameters

The following table describes the parameters of the Abs command:

Parameter	Description
<i>number</i>	The number to get the absolute value for

Return Value

The absolute value of the number

Activate

Syntax

Activate *task*

Synonyms

AC

Description

The Activate command switches the input focus to the named task. The window that you want to show must be currently in the task list (accessed by pressing ALT+TAB to scroll through running applications). The task name must be the same as it displays in the Windows task list.

Parameters

The following table describes the parameters of the Activate command:

Parameter	Description
<i>task</i>	The title of the task

Example

This example is a part of the COPYMENU.WIS program.

```
title_text = "Untitled - Notepad"
If IsTask(title_text) Then
  Invoke EditCopy
  Activate title_text
  SendKeys "^V" ;* Paste key for notepad
Else
  MessageBox "This example copies to ":title_text:" only", "Copy To
  Notepad"
EndIf
```

Related Script Commands

[IsTask](#), [SendKeys](#)

AddCheckBox

Syntax

AddCheckBox *name, text, width, command*

Description

The AddCheckBox command adds a check box control to a toolbar.

The AddCheckBox command is a Icon Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

The following table describes the parameters of the AddCheckBox command:

Parameter	Description
<i>name</i>	The name for the check box
<i>text</i>	The text to display beside the check box
<i>width</i>	The width of the check box in dialog units
<i>command</i>	The command the check box performs when it is clicked upon

Version

4.0 Original

AddComboBox

Syntax

AddComboBox *name, width, depth, editable*

Description

The AddComboBox command adds a combo box control to a toolbar.

The AddComboBox command is an Icon Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

The following table describes the parameters of the AddComboBox command:

Parameter	Description
<i>name</i>	The name for the combo box
<i>width</i>	The width of the combo box in dialog units
<i>depth</i>	The depth of the combo box in dialog units
<i>editable</i>	True if the combo box can be edited, false if the a value must be picked from the drop down list

Version

4.0 Original

AddGap

Syntax

AddGap

Description

The AddGap command adds a gap after an icon. The AddGap command is a Icon Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

None

Example

The following example is from the C:\wintegrate\wintsys\iconbar\bar_gen.wis iconbar script.

```
There is a gap between groups of icons, added with AddGap.  
CreateIconBar General_bar,1  
AddIcon 'image\i_open.bmp',"Show FileOpen"  
AddIcon 'image\i_save.bmp',"Show FileSaveAs"  
AddIcon 'image\i_copy.bmp',"Invoke EditCopy"  
AddIcon 'image\i_paste.bmp',"Invoke EditPaste"  
AddGap
```

Related Script Commands

[AddIcon](#), [AddLargeIcon](#), [CreateIconBar](#), [NextRow](#), [EndCreateIconBar](#)

AddIcon

Syntax

AddIcon *bitmap, command*

Description

The AddIcon command adds a small icon (16x16) to an icon bar. The AddIcon command is a Icon Library command. For more information, see "Appendix A -Scripting Libraries and Modules". Note To add large icons (32x32) to an iconbar, see AddLargeIcon.

Parameters

The following table describes the parameters of the AddIcon command:

Parameter	Description
<i>bitmap</i>	The file name of the bitmap (.bmp) or icon (.ico) to add as an icon. The dimensions must be 16x16 to match the default icon size as in bar_gen.wis.
<i>command</i>	The command that the icon performs when clicked on.

Example

The following example adds a series of .bmp files as icons to the icon bar General_bar. This example is from the C:\winteg\wintsys\iconbar\bar_gen.wis iconbar script.

```
Library 'wintsys\lib\iconlib'  
CreateIconBar General_bar,1  
AddIcon 'image\i_open.bmp',"Show FileOpen"  
AddIcon 'image\i_save.bmp',"Show FileSaveAs"  
AddIcon 'image\i_copy.bmp',"Invoke EditCopy"  
AddIcon 'image\i_paste.bmp',"Invoke EditPaste"  
AddGap
```

Related Script Commands

[AddGap](#), [AddLargeIcon](#), [AddWideIcon](#), [CreateIconBar](#), [NextRow](#),
[EndCreateIconBar](#)

AddLargeIcon

Syntax

AddLargeIcon *bitmap, command*

Description

The AddLargeIcon command adds a large icon (32x32) to an icon bar. The AddLargeIcon command is a Icon Library command. For more information, see "Appendix A - Scripting Libraries and Modules". Note To add smaller (16x16) icons to an iconbar, see AddIcon.

Parameters

The following table describes the parameters of the AddLargeIcon command:

Parameter	Description
<i>bitmap</i>	The file name of the bitmap to add as an icon. The bitmap dimensions must be 32x32 to match the default icon size as in bar_big.wis.
<i>command</i>	The command that the icon performs when clicked on.

Example

This example is from the e C:\winteg\wintsys\iconbar\bar_big.wis iconbar script.

```
Library 'wintsys\lib\iconlib'  
CreateIconBar BigIcon_bar,2  
AddLargeIcon 'icon\excel.ico','Dialog RunProgram;Set Filename =  
"excel.exe";Set Arguments = "";Invoke'  
AddLargeIcon 'icon\word.ico','Dialog RunProgram;Set Filename =  
"win-word.  
exe";Set Arguments = "";Invoke'
```

Related Script Commands

[AddGap](#), [AddIcon](#), [AddWideIcon](#), [CreateIconBar](#), [NextRow](#), [EndCreateIconBar](#)

AddLargeSeparator

Syntax

AddLargeSeparator

Description

The AddLargeSeparator command adds a vertical etched line to a toolbar which is two rows deep.

The AddLargeSeparator command is a Icon Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

None

Version

4.0 Original

AddLargeStyleIcon

Syntax

AddLargeStyleIcon *bitmap, command, style*

Description

The AddLargeStyleIcon command adds a large icon (32x32) to an icon bar allowing the style of the graphic button used for the icon to be set.

The AddLargeIcon command is a Icon Library command. For more information, see "Appendix A - Scripting Libraries and Modules".

This command is implemented in the NONE script library.

Parameters

The following table describes the parameters of the AddLargeStyleIcon command:

Parameter	Description
<i>bitmap</i>	The file name of the bitmap to add as an icon. The bitmap dimensions must be 32x32 to match the default icon size as in bar_big.wis.
<i>command</i>	The command that the icon performs when clicked on.
<i>style</i>	The style for the graphic button. See DialogBox GraphicButton for a list of styles.

Example

Disable the spreadsheet icon on the big iconbar

```
Library 'wintsys\lib\iconlib'  
CreateIconBar BigIcon_bar,2  
AddLargeStyleIcon 'icon\excel.ico','Dialog RunProgram;Set Filename =  
"excel.exe";Set Arguments = "";Invoke', WS_DISABLED|GBS_FLAT  
...
```

Related Script Commands

[AddGap](#), [AddIcon](#), [AddWideIcon](#), [CreateIconBar](#), [NextRow](#), [EndCreateIconBar](#),
[AddStyleIcon](#), [AddLargeStyleIcon](#), [AddWideStyleIcon](#), [DialogBox](#) [GraphicButton](#)

Version

6.0 Original

AddSeparator

Syntax

AddSeparator

Description

The AddSeparator command adds a vertical etched line to a toolbar.

The AddSeparator command is a Icon Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

None

Version

4.0 Original

AddStyleIcon

Syntax

AddStyleIcon *bitmap, command, style*

Description

The AddStyleIcon command adds a small icon (16x16) to an icon bar allowing the style of the GraphicsButton used to show the icon to be set.

The AddStyleIcon command is a Icon Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

This command is implemented in the NONE script library.

Parameters

The following table describes the parameters of the AddStyleIcon command:

Parameter	Description
<i>bitmap</i>	The file name of the bitmap (.bmp) or icon (.ico) to add as an icon. The dimensions must be 16x16 to match the default icon size as in bar_gen.wis.
<i>command</i>	The command that the icon performs when clicked on.
<i>style</i>	The graphic button style to be used. See DialogBox GraphicButton for details.

Example

The following example adds icons, with the save as icon disabled.

```
Library 'wintsys\lib\iconlib'  
CreateIconBar Test_bar,1  
AddIcon 'image\i_open.bmp',"Show FileOpen"  
AddStyleIcon 'image\i_save.bmp',"Show FileSaveAs",  
GBS_FLAT|WS_DISABLED
```

Related Script Commands

[AddGap](#), [AddLargeIcon](#), [AddWideIcon](#), [CreateIconBar](#), [NextRow](#),
[EndCreateIconBar](#), [AddIcon](#), [AddLargeStyleIcon](#), [AddWideStyleIcon](#), [DialogBox](#)
[GraphicButton](#)

Version

6.0 Original version

AddTool

Syntax

AddTool *bitmap, command*

Description

The AddTool command adds a new icon to the toolbar. The AddTool command is a Tool Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

The following table describes the parameters of the AddTool command:

Parameter	Description
<i>bitmap</i>	The file name of the bitmap to add as an icon. If you are using the default icon size in CreateToolBar, the bitmap dimensions must be 16x16. If you specified large icons, they must be 32x32.
<i>command</i>	The command that the icon performs when clicked on.

Example

The following example demonstrates how to create a toolbar using the same icons as in the large icon bar.

```
* Big Tool bar
If IsShown(BigTool_bar) Then
EndScript
If Not(IsDialog(BigTool_bar)) Then
Library 'wintsys\lib\toollib'
CreateToolBar BigTool_bar,1,6,"L"
AddTool 'icon\excel.ico','Dialog RunProgram;Set Filename =
"excel.exe";Set
Arguments = "";Invoke'
AddTool 'icon\word.ico','Dialog RunProgram;Set Filename =
"winword.exe";Set
Arguments = "";Invoke'
```

```
AddTool 'icon\control.ico','Invoke FileControlPanel'
AddTool 'icon\write.ico','Dialog RunProgram;Set Filename =
"write.exe";Set
Arguments = "";Invoke'
AddTool 'icon\pbrush.ico','Dialog RunProgram;Set Filename =
"pbrush.exe";Set
Arguments = "";Invoke'
AddTool 'icon\help.ico','Invoke HelpIndex'
ToolSpace
EndCreate
EndIf
DialogBox Window BigTool_bar
```

Related Script Commands

[AddToolTip](#), [CreateToolBar](#), [ToolSpace](#), [EndCreateIconBar](#)

AddToolTip

Syntax

AddToolTip *text*

Description

The AddToolTip command adds a tooltip to the last icon defined using AddIcon, AddLargeIcon, or AddWideIcon script library commands.

A tooltip is a small rectangle that displays when the mouse is moved over an icon.

The AddToolTip command is a Icon Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

The following table describes the parameters of the AddToolTip command:

Parameter	Description
<i>text</i>	Tooltip text to display

Example

The following example creates a ToolTip that displays "copy".

```
AddIcon "image\i_copy.bmp", "Invoke EditCopy"  
AddToolTip "copy"
```

Related Script Commands

[DialogBox](#) [ToolTip](#)

AddWideIcon

Syntax

AddWideIcon *bitmap, width, command*

Description

The AddWideIcon command adds an icon with a specified width to an icon bar. The AddWideIcon command is a Icon Library command. For more information, see "Appendix A - Scripting Libraries and Modules".

Parameters

The following table describes the parameters of the AddWideIcon command:

Parameter	Description
<i>bitmap</i>	The file name of the bitmap to add as an icon. The dimensions of the bitmap must be 16 x width.
<i>width</i>	The width of the icon. You specify the width in dialog box units. For more information, see DialogUnit.
<i>command</i>	The command that the icon performs when clicked on.

Related Script Commands

[AddGap](#), [AddIcon](#), [AddLargeIcon](#), [CreateIconBar](#), [NextRow](#), [EndCreateIconBar](#)

AddWideStyleIcon

Syntax

AddWideStyleIcon *bitmap, width, command, style*

Description

The AddWideStyleIcon command adds an icon with a specified width and graphic button style to an icon bar.

The AddWideStyleIcon command is a Icon Library command. For more information, see "Appendix A - Scripting Libraries and Modules".

This command is implemented in the NONE script library.

Parameters

The following table describes the parameters of the AddWideStyleIcon command:

Parameter	Description
<i>bitmap</i>	The file name of the bitmap to add as an icon. The dimensions of the bitmap must be 16 x width.
<i>width</i>	The width of the icon. You specify the width in dialog box units. For more information, see DialogUnit.
<i>command</i>	The command that the icon performs when clicked on.
<i>style</i>	The style of the graphic button used to display the icon. See DialogBox Graphic button for available styles.

Related Script Commands

[AddGap](#), [AddIcon](#), [AddLargeIcon](#), [CreateIconBar](#), [NextRow](#), [EndCreateIconBar](#), [AddStyleIcon](#), [AddLargeStyleIcon](#), [AddWideIcon](#), [DialogBox GraphicButton](#)

Version

6.0 Original

AlwaysOnTop

Syntax

AlwaysOnTop flag

Description

The AlwaysOnTop command controls whether or not other windows can be displayed on top of the session window.

Parameters

The following table describes the parameters of the AlwaysOnTop command:

Parameter	Description
<i>flag</i>	Can be set to one the following: TRUE, ON, YES or any whole number other than 0 - The session window always stays on top of all other windows. FALSE, OFF, NO, or 0 - Other windows can be on top of the session window.

Example

This example turns on the AlwaysOnTop feature, which does not allow other application windows to cover the wIntegrate window.

```
AlwaysOnTop TRUE
```

And

Syntax

expr1 **And** *expr2*

Description

The And Boolean operator combines a set of expressions by placing it between them as follows:

expr1 And *expr2*

If *expr1* or *expr2* is false, the combined expression is false. Both expressions must be true for a true condition.

Parameters

The following table describes the parameters of the And command:

Parameter	Description
<i>expr1</i>	Expression evaluated to true or false
<i>expr2</i>	Expression evaluated to true or false

Example

Checking that two conditions must be true

```
If season = "winter" And weather = "sunny" Then Forecast_wrong = TRUE
```

Related Script Commands

[If](#), [Loop](#), [Or](#), [Until](#), [While](#)

AppFolder

Syntax

AppFolder *FolderName*, [*DefaultPath*], [*AddToFolderList*], [*overwrite*]

Description

This function returns the full path to the folder where files of the type given by *FolderName* are loaded from and/or saved to. If *FolderName* specifies an unknown folder name the path to the User directory is returned unless a default folder path has been specified in *DefaultPath*.

If the *FolderName* is specified as "?" a list of all currently defined folder names is returned.

Parameters

The following table describes the parameters of the AppFolder command:

Parameter	Description
<i>FolderName</i>	The name of the folder to retrieve. Use "?" to return a list of all folder names.
<i>DefaultPath</i>	The default path for the folder if the folder name does not exist. This can be a full path or just a path relative to the user directory. If the <i>FolderName</i> is "?" set this value to true to return the current paths for the folders as well as their names.
<i>AddToFolderList</i>	Set to true to add the name of this folder to the list of folders. If the name does not already exist the Default Path parameter will be used for the initial location of the folder.
<i>overwrite</i>	Set to true to overwrite the current path to the folder with the value specified in the <i>DefaultPath</i> parameter.

Return Value

The return value is the full path to the folder where items for the requested folder name are loaded from and/or saved to.

Example

Display the location the session files are saved in by default.

```
MessageBox AppFolder("Session")
```

Set up the scheduler folder

```
sched_folder = AppFolder("Scheduler","Schedule",true)
File CreateDir sched_folder,err,true
```

Get the path for MyFolder

```
* Get the path to the folder name MyFolder if it exists otherwise
* return "C:\My Documents". Don't add it to the list of folders if
* it isn't already there.
my_folder = AppFolder("MyFolder", "C:\My Documents")
```

Version

4.1.1 Original

4.2.1 DefaultPath and AddToFolderList added

5.0 "?" value to return all folder names. overwrite parameter.

Asc

Syntax

Asc char

Description

The Asc function returns the ASCII (American Standard Code for Information Interchange) value of a character.

As of version 5.2 this function will now return the Unicode value of the character.

Parameters

The following table describes the parameters of the Asc command:

Parameter	Description
<i>char</i>	The character to return the ASCII value.

Return Value

The number representing the character

Example

The following example prints 65 in the status bar.

```
* Display the ASCII number for the letter "A" in the status bar
Print Asc("A")
```

Related Script Commands

[Char](#)

Version

5.2 Support for Unicode characters

AsciiToBS

Syntax

AsciiToBS *string*

Description

The AsciiToBS function converts an ASCII (American Standard Code for Information Inter-change) string to backslash format. For example, the ASCII value of 10 represents a line feed or \n in backslash format. Tip You may want to use this as a way of using text containing control codes.

Parameters

The following table describes the parameters of the AsciiToBS command:

Parameter	Description
<i>string</i>	The string to convert

Return Value

A copy of the string converted to the back slash format

Related Script Commands

[AsciiToFT](#), [AsciiToHex](#), [BSToAscii](#), [FTToAscii](#), [HexToAscii](#)

AsciiToFT

Syntax

AsciiToFT *string*

Description

The AsciiToFT function converts an ASCII (American Standard Code for Information Inter-change) format string to its FT (file transfer) equivalent. The FT format represents control codes and other special characters as 1 or 2 7-bit values that wIntegrate uses when transferring data and control codes to and from the host.

Parameters

The following table describes the parameters of the AsciiToFT command:

Parameter	Description
<i>string</i>	The string to convert

Return Value

A copy of the string after being convert to FT format

Related Script Commands

[AsciiToBS](#), [AsciiToHex](#), [BSToAscii](#), [FTToAscii](#), [HexToAscii](#)

AsciiToHex

Syntax

AsciiToHex *string*

Description

The **AsciiToHex** function converts an ASCII (American Standard Code for Information Inter-change) string to a string of 2 digit hex numbers representing each character in the string. In Hex format, every byte is a 2-byte hexadecimal version, so a particular byte position can be located directly.

Parameters

The following table describes the parameters of the **AsciiToHex** command:

Parameter	Description
<i>string</i>	The string to convert

Return Value

A copy of the string converted to hex format

Related Script Commands

[AsciiToBS](#), [AsciiToFT](#), [BStoAscii](#), [FTToAscii](#), [HexToAscii](#)

Attribute

Syntax

Attribute Options, [x], [y], [page]

Description

This function returns the current screen attribute or the attribute at the specified location.

Parameters

The following table describes the parameters of the Attribute command:

Parameter	Description
<i>Options</i>	Specifies the value to return. See below.
<i>x</i>	x co-ordinate to read attribute from. Default current position of cursor
<i>y</i>	y co-ordinate to read attribute from. Default current position of cursor
<i>page</i>	Back page number to check. Default 0

Values for Options

The following option values determine the value that is returned by this function.

Value	Description
<i>1</i>	Effect (returns a value from the effect table, see below)
<i>2</i>	Font bank (returns font number 0-3)
<i>4</i>	Foreground color (returns color number 0-15)
<i>8</i>	Background color (returns color number 0-15)
<i>12</i>	Foreground color + 16 * background colour (returns combined color number 0-255)

Return Value

The return value is dependent on the option and is in the range specified in the option description above. The number returned for the effect is made up of the following bits.

Value	Description
<i>0</i>	Normal
<i>1</i>	Dim
<i>2</i>	Reverse
<i>4</i>	Flash
<i>8</i>	Underline
<i>16</i>	Bold
<i>32</i>	Secret

Example

Display the foreground color to be used for the next character to be written to the screen

```
MessageBox Attribute(4)
```

Check if the start of the bottom line of the screen is displayed in red

```
If Attribute(4,0,24) = COL_Red Then ...
```

Check if the cursor is over a reversed field

```
If Attribute(1,,,0) = 2 Then ...
```

Version

4.0.3 Original

AutoWrapOn

Syntax

AutoWrapOn

Description

This function checks if auto wrap mode is currently on.

Parameters

None

Return Value

True (1) if autowrap is on, otherwise False (0).

Related Script Commands

[Display Command](#)

Version

5.0 Original

BaseBar

Syntax

BaseBar

Description

The BaseBar function returns the name of the current base bar.

Parameters

None

Return Value

The current name of the base bar or "" if there is no base bar

Related Script Commands

[IconBar](#)

Blue

Syntax

Blue *color*

Description

The Blue function returns the blue intensity of a specified RGB (Red Green Blue) color. On a color monitor, colors are displayed as varying intensities of red, green, and blue dots. Intensity is the amount of red, green, or blue on a scale of 0 through 255.

Parameters

The following table describes the parameters of the Blue command:

Parameter	Description
<i>color</i>	The color name or number.

Return Value

The amount of blue in the color (0 to 255)

Example

This returns 128 as magenta contains 50% blue

```
Print Blue( RGB_Magenta )
```

In the next example, a message box displays showing that the color intensity for blue is 50.

```
colortest=RGB(50,50,50)  
MessageBox "Blue intensity = ": Blue(colortest)
```

Related Script Commands

[ChooseColor](#), [Green](#), [Red](#), [RGB](#)

BSToAscii

Syntax

BSToAscii *string*

Description

The BSToAscii function converts a backslash format string to ASCII (American Standard Code for Information Interchange) format. For example, the ASCII value of 10 represents a line feed or \n in backslash format.

Parameters

The following table describes the parameters of the BSToAscii command:

Parameter	Description
<i>string</i>	The string to convert

Return Value

A copy of the string converted from back slash format to ASCII text

Example

Get contents of key F1 as ascii text

```
key_val = Get(F1) ;* Gets value of F1 key (in BS format)
asc_val = BSToAscii(key_val) ;* Converts to ASCII value
```

Related Script Commands

[AsciiToBS](#), [AsciiToFT](#), [AsciiToHex](#), [FTToAscii](#), [HexToAscii](#)

Capture AddData

Syntax

Capture AddData *name, data*

Synonyms

CP AD

Description

The Capture AddData command adds data you specify to a capture session.

Parameters

The following table describes the parameters of the Capture AddData command:

Parameter	Description
<i>name</i>	The name of the capture session.
<i>data</i>	The data to add to the capture session. Note - data must be in backslash format.

Example

* To add a call to pause script during recording of host data

```
Capture AddData democap, "\e\001Script 'pause'\r"
```

Related Script Commands

[Capture Delete](#), [CaptureInfo](#), [Capture Off](#), [Capture On](#), [CaptureText](#)

Capture Continue

Syntax

Capture Continue *name*

Synonyms

CP CN

Description

The Capture Continue command continues the recording of data after a Capture Pause command has been issued.

Parameters

The following table describes the parameters of the Capture Continue command:

Parameter	Description
<i>name</i>	The name for this paused capture session

Version

4.0.3 Original

Capture Delete

Syntax

Capture Delete *name*

Synonyms

CP DL

Description

The Capture Delete command deletes a capture session.

Parameters

The following table describes the parameters of the Capture Delete command:

Parameter	Description
<i>name</i>	The name of the capture session to delete.

Example

This example is part of the PRESTORE.WIS demonstration program.

```
Sub OnPClick(pno)
id = "P":pno
cname = "Prestore_":id
If Extract(Prestore_State,pno) Then
* Recording in progress
Capture Off $cname
Rep Prestore_text, pno, CaptureText($cname, DF_BS)
Capture Delete $cname
DialogBox SetText Prestore,$id,id
Rep Prestore_State,pno,0
Else If Prestore.Rec Then
Capture On $cname, "", "Keystrokes"
DialogBox SetText Prestore,$id,"R":pno
Rep Prestore_State,pno,1
Else
Type Extract(Prestore_text, pno)
EndIf
EndSub
```

Related Script Commands

[Capture AddData](#), [CaptureInfo](#), [Capture Off](#), [Capture On](#), [CaptureText](#)

Capture Off

Syntax

Capture Off *name*

Synonyms

CP OF

Description

The Capture Off command turns off a capture session.

Parameters

The following table describes the parameters of the Capture Off command:

Parameter	Description
<i>name</i>	The name of the capture session to turn off.

Example

This example is part of the PRESTORE.WIS demonstration program.

```
Sub OnPClick(pno)
id = "P":pno
cname = "Prestore_":id
If Extract(Prestore_State,pno) Then
* Recording in progress
Capture Off $cname
Rep Prestore_text, pno, CaptureText($cname, DF_BS)
Capture Delete $cname
DialogBox SetText Prestore,$id,id
Rep Prestore_State,pno,0
Else If Prestore.Rec Then
Capture On $cname, "", "Keystrokes"
DialogBox SetText Prestore,$id,"R":pno
Rep Prestore_State,pno,1
Else
Type Extract(Prestore_text, pno)
EndIf
EndSub
```

Related Script Commands

[Capture AddData](#), [Capture Delete](#), [CaptureInfo](#), [Capture On](#), [CaptureText](#)

Capture On

Syntax

Capture On *name*, [*to*], [*format*], [*buffer_size*], [*options*], *columns*

Synonyms

CP ON

Description

The Capture On command starts a capture with a name that you specify.

Note: If the name you specify is already an active capture session, the capture session restarts with the new parameters.

Parameters

The following table describes the parameters of the Capture On command:

Parameter	Description
<i>name</i>	The name for this capture session.
<i>to</i>	The destination for the captured data. This can be "Memory", "Printer", or the full path of a file. The default is "Memory".
<i>format</i>	The format of the data capture. See below
<i>buffer_size</i>	The size of the memory buffer.
<i>options</i>	For information, see the following Options table.
<i>columns</i>	Comma separated list of the column widths for use with the table format

Values for format

Value	Description
"RawData"	Captures all host characters.
"Keystrokes"	Captures keystrokes.

Value	Description
<i>"Screens"</i>	Captures each screen as the next appears.
<i>"ControlCodes"</i>	Captures host data control codes.
<i>"Session"</i>	Captures the data sent to and received from the host. Each piece of data is prefixed by a "<" if sent to the host and a ">" if received from the host.
<i>"Table"</i>	Captures each line with tabs separating columns. The column widths are specified in the columns parameter.

Values for options

You can specify more than one option by separating them with a pipe (|):

Value	Description
<i>CAP_APPEND</i>	Appends data to a memory or file capture.
<i>CAP_AUTOOFF</i>	Automatically turns off the capture when the buffer is full.
<i>CAP_CharMapOff</i>	Stops the characters from being remapped during capturing.
<i>CAP_DIRECT</i>	This option sends captured data directly to the printer without formatting.
<i>CAP_KEEP</i>	Retains the capture in memory when capture off is executed. Use Capture Delete when the capture is no longer required.
<i>CAP_SHIFT</i>	Retains at least half the buffer if the buffer is full.

Value	Description
<i>CAP_ReadDirect</i>	Reads the capture direct settings from the "Printer Setup" dialog box available on the File menu. When this flag is set the print direct flag and use character mapping flags are read and used. Turning off character mapping in this dialog box also stops the terminal commands i.e. both the CAP_CharMapOff and CAP_TermCommandOff flags are set.
<i>CAP_TermCommandOff</i>	Stops Terminal Commands, etc., in the WIT file being executed during a capture. Once a capture session has been started using CAP_TermCommandOff, it can only be stopped by appending a “,P” to the terminal command line. See example below

Example

This example is part of the PRESTORE.WIS demonstration program.

```
Sub OnPClick(pno)
id = "P":pno
cname = "Prestore_":id
If Extract(Prestore_State,pno) Then
    * Recording in progress
    Capture Off $cname
    Rep Prestore_text, pno, CaptureText($cname, DF_BS)
    Capture Delete $cname
    DialogBox SetText Prestore,$id,id
    Rep Prestore_State,pno,0
Else If Prestore.Rec Then
    Capture On $cname, "","Keystrokes"
    DialogBox SetText Prestore,$id,"R":pno
    Rep Prestore_State,pno,1
Else
    Type Extract(Prestore_text, pno)
EndIf

EndSub
```

Shows the use of CAP_TermCommandOff

```
* For example, in TVI955.WIT:
TerminalCommand = "\e'", "IB 145Sending data to printer...146;SN OFF;CP
On LocalPrint,146Printer146,,,CAP_TermCommandOff"
TerminalCommand = "\ea", "SN ON;IB;If Capture-Info(LocalPrint) Then CP
Off
LocalPrint",P
```

Capture the columns from a report

```
Capture On MyRecord, FileName, "Table",,, "11,20,10,9,7"
```

Related Script Commands

[Capture AddData](#), [Capture Delete](#), [CaptureInfo](#), [Capture Off](#), [CaptureText](#)

Version

4.0.3 Table format added

Capture Pause

Syntax

Capture Pause *name*

Synonyms

CP PS

Description

The Capture Pause command pause the recording of data until a Capture Continue command is issued.

Parameters

The following table describes the parameters of the Capture Pause command:

Parameter	Description
<i>name</i>	The name for this capture session

Example

Pause during capture to display percentage complete

```
* The following uses the new commands in a host program to
* Display a percentage complete on the status bar.
* HostPrt is the name of the Capture used by WIN.PRINTON
CALL WIN.PRINTON(TRUE)
LOOP
WHILE MORE.LINES DO
GOSUB 1000;* Print line
IF PC # OLD.PC THEN
CALL WIN.COMSUB("Capture Pause HostPrt")
CALL WIN.STATLINE(PC:"% complete")
CALL WIN.COMSUB("Capture Continue HostPrt")
OLD.PC = PC
END
REPEAT
CALL WIN.PRINTOFF(TRUE)
```

Version

4.0.3 Original

CaptureInfo

Syntax

CaptureInfo *name*, [*option*]

Description

The CaptureInfo command returns information on a capture session.

Parameters

The following table describes the parameters of the CaptureInfo command:

Parameter	Description
<i>name</i>	The name of the capture session specified in Capture On.
<i>option</i>	The type of information you want to return:

Values for option

Value	Description
0	Returns true if the capture is running. This is the default if you do not specify an option.
1	Returns the number of bytes in the capture buffer.
2	Returns true if the capture is active and currently paused
3	Returns true if the capture is active, and either running or paused. Returns false if capture is inactive

Return Value

The information as specified by the option parameter

Example

This example prints a message if the capture is still running.

```
If CaptureInfo(capture_1) Then  
Print "Capture 1 is still recording"
```

Related Script Commands

[Capture AddData](#), [Capture Delete](#), [Capture Off](#), [Capture On](#), [CaptureText](#)

Version

4.0.3 Capture pause status options

CaptureText

Syntax

CaptureText *name*, [*option*]

Description

The CaptureText function returns converted text from the capture buffer.

Parameters

The following table describes the parameters of the CaptureText command:

Parameter	Description
<i>name</i>	the name of the capture, specified in Capture On.
<i>option</i>	Allows you to choose a conversion option for data. See table below the default is 0 (no conversion).

Values for option

Value	Description
<i>DF_BS</i>	Returns data in BackSlash format.
<i>DF_Hex</i>	Returns data in Hex format.
<i>DF_FT</i>	Returns data in FT format.
<i>DF_NoZero</i>	Strips character 0's from data.

Return Value

The text that has currently be captured. The format the text is returned in is specified by the option parameter.

Example

This example is part of the PRESTORE.WIS demonstration program.

```
Sub OnPClick(pno)
id = "P":pno
cname = "Prestore_":id
If Extract(Prestore_State,pno) Then
* Recording in progress
Capture Off $cname
Rep Prestore_text, pno, CaptureText($cname, DF_BS)
Capture Delete $cname
DialogBox SetText Prestore,$id,id
Rep Prestore_State,pno,0
Capture On $cname, "", "Keystrokes"
DialogBox SetText Prestore,$id,"R":pno
Rep Prestore_State,pno,1
Else
Type Extract(Prestore_text, pno)
EndIf
EndSub
```

Related Script Commands

[Capture AddData](#), [Capture Delete](#), [CaptureInfo](#), [Capture Off](#), [Capture On](#)

Chain

Syntax

Chain *scriptfile*, [*commandline*]

Synonyms

CH

Description

The Chain command terminates the current script and executes another script. Chain performs a function similar to the Script command, except that control is not returned to the original program.

Parameters

The following table describes the parameters of the Chain command:

Parameter	Description
<i>scriptfile</i>	The full path name of the script file. If the file is in the "wIntegrate" directory or a subdirectory, you do not need the full path name.
<i>commandline</i>	The command line to pass to the script. When the new script starts, the global variable CommandLine is set to the value of commandline or to Null if commandline is omitted.

Example

The following example terminates the current script and executes the script "dispbox.wis":

```
Chain "C:\wintegrate\example\script\dispbox.wis"
Since the file is in a subdirectory of the wIntegrate directory, this
line can also be written as fol-ows:
Chain "example\script\dispbox.wis"
```

Related Script Commands

[Execute](#), [Global](#), [Script](#)

Change

Syntax

Change *string, from_string, to_string*

Description

The Change command replaces all occurrences of a string with a new string.

Parameters

The following table describes the parameters of the Change command:

Parameter	Description
<i>string</i>	Specifies the original text string.
<i>from_string</i>	Specifies the text string to replace with <i>to_string</i> .
<i>to_string</i>	The text string with which to replace <i>from_string</i> .

Return Value

Returns a copy of the string with the relevant changes

Example

The following example produces a message box with the displayed text depending on the internal time.

```
* Message box greeting depending on time
var1 = "Good Morning!"
var2 = Change(var1, "Morning!", "Afternoon!")
* (43200 is the number of seconds from midnight to noon in internal
time.)
If Time() < 43200 Then
    MessageBox " " : var1
Else
    MessageBox " " : var2
EndIf
```

Related Script Commands

[Convert](#)

ChangeUserSession

Syntax

ChangeUserSession *name*, *result_var*, [*create*], [*allow_login*]

Description

This command will change the user in the Thin client version of wIntegrate.

It is used so that a single login can be used for all clients that validates the user via a wIntegrate script. If the user is valid the script would then call this command so that the user has his own settings and files.

This command will only operate if it has been enabled in the Modify User dialog of the user maintenance program.

New users created by this command can only be logged in by this command unless specifically allowed to log in directly from the Modify User dialog of the user maintenance program.

If run on the local version of wIntegrate it will always return error code 1.

Parameters

The following table describes the parameters of the ChangeUserSession command:

Parameter	Description
<i>name</i>	The name to change this session user to.
<i>result_var</i>	A script variable that will return the result from this command. Either 0 or an error code. See the table below.
<i>create</i>	Set to true to create a new user if the specified user does not already exist. Defaults to false.
<i>allow_login</i>	Set to true to allow a new user created by this command to login directly to the server with this name. Defaults to false.

Values for result_var

The values returned in the script variable are:

Value	Description
0	The user was changed successfully
1	Permission to use this command was denied
2	The name specified is the same as the name of the current user
3	The name is either invalid or login via this name has been denied
4	Licensing does not allow any more users
5	Unable to create the folders for a new user
6	Unable to run this command as a File Another command is in progress
7	Unable to allocate the memory required to change the user
8	Unable to create a new user
9	The name contains a reserved word or invalid characters

Example

Changes to user name "Fred"

```
ChangeUserSession "fred", err  
If err = 0 Then MessageBox "User is now fred"
```

Version

6.0 Original version

Char

Syntax

Char *number*

Description

The Char function returns the character for an ASCII code number. The Char function changes a numeric expression to its ASCII (American Standard Code for Information Interchange) character string equivalent.

As of version 5.2 this function will also convert to Unicode characters.

Parameters

The following table describes the parameters of the Char command:

Parameter	Description
<i>number</i>	The number to return as a character. number must evaluate to a positive number from 0 to 65535 (the range of Unicode character codes).

Return Value

The character for the specified code

Example

The following example prints "A" in the status bar.

```
Print Char(65)
```

Related Script Commands

[Asc](#)

Version

5.2 Unicode character support

CheckServer

Syntax

CheckServer

Description

The CheckServer command checks if the host side of the wIntegrate server is running, opening it if it is not. If the CheckServer command fails it set the global variable Server_Error to 1 if it couldn't start the host server or 2 if the host server is busy.

All other server library routines automatically call this routine before they proceed.

The CheckServer command is a Server Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

None

Version

4.0 Original

ChooseColor

Syntax

ChooseColor *default*, [*options*], [*custom_var*]

Description

The ChooseColor function displays the Color dialog box that lets the user choose a color, and then returns the RGB value of the color selected.

You can also use the alternative spelling of this function: ChooseColour.

Parameters

The following table describes the parameters of the ChooseColor command:

Parameter	Description
<i>default</i>	Specifies the default color to display when opening the dialog box.
<i>options</i>	See below
<i>custom_var</i>	Variable name from which to get/set list of custom colors. custom_var is an array of up to 16 colors which display in the Custom color section of the dialog box. Any colors which are not defined display as black but can still be defined by the user.

Values for options

Value	Description
<i>1</i>	Display full dialog including the custom color selection.
<i>2</i>	Do not allow user to modify custom colors.

Return Value

The RGB color number for the selected color or "" if cancel was pressed.

Example

The following example displays the Color dialog box, does not display the customer color selection, and allows the user to define colors.

```
* Set default color to red
def_col = RGB_Red
* Fill custom colors with standard color for which there
* are constants already defined (could have used the RGB function).
cust_cols = ""
Ins cust_cols,-1,RGB_Black
Ins cust_cols,-1,RGB_Green
Ins cust_cols,-1,RGB_Cyan
Ins cust_cols,-1,RGB_Red
Ins cust_cols,-1,RGB_Magenta
Ins cust_cols,-1,RGB_Brown
Ins cust_cols,-1,RGB_LightGrey
Ins cust_cols,-1,RGB_LightBlue
Ins cust_cols,-1,RGB_LightGreen
Ins cust_cols,-1,RGB_LightCyan
Ins cust_cols,-1,RGB_LightRed
Ins cust_cols,-1,RGB_LightMagenta
Ins cust_cols,-1,RGB_Yellow
Ins cust_cols,-1,RGB_White

* Display the dialog box
val = ChooseColor(def_col, 0, cust_cols)
MessageBox "Result = ":val:"
(":red(val):",":green(val):",":blue(val):")
MessageBox cust_cols, "Custom Colours"
```

Related Script Commands

[Blue](#), [Green](#), [Red](#), [RGB](#)

ChooseFont

Syntax

ChooseFont [*FontName*], [*FontSize*], [*FontStyle*], [*FontColor*], *Options*, [*PrinterName*]

Description

This function displays a standard Windows font selection dialog that allows the user to select a font.

Parameters

The following table describes the parameters of the ChooseFont command:

Parameter	Description
<i>FontName</i>	Default font name
<i>FontSize</i>	The point size of the font
<i>FontStyle</i>	The style for the font. See table below.
<i>FontColor</i>	The color for the font. The color information is only used and returned if the CF_EFFECT flag is included in the options.
<i>Options</i>	See table below.
<i>PrinterName</i>	The name of the printer to display the fonts for. The options must include the CF_PRINTERFONTS flag for this parameter to be used. If the printer name is set to "" or omitted the fonts for the default printer are shown.

Values for FontStyle

The font style can be one or more of the following combined with the "|" operator.

Value	Description
<i>FONT_Bold</i>	Sets a bold font.
<i>FONT_Italic</i>	Sets an italic font.

Value	Description
<i>FONT_Underline</i>	Underlines the text.
<i>FONT_StrikeOut</i>	Places a line through the text.

Values for Options

Combine one or more the options using the "|" symbol.

Value	Description
<i>CF_TTONLY</i>	Show True type fonts only
<i>CF_EFFECTS</i>	Show underline/strikeout and colors
<i>CF_FIXEDPITCHONLY</i>	Show only fixed pitch fonts
<i>CF_FORCEFONTEXIST</i>	User must pick a font that exists
<i>CF_SCREENFONTS</i>	Show fonts for the screen display
<i>CF_PRINTERFONTS</i>	Show fonts for the printer

Return Value

The font selected as name,font size and style or "" if cancel was pressed. If the CF_EFFECTS flag was included as an option a comma and then the color selected for the font is appended to the return value.

Example

Select a font for the default printer

```
* We set the default to "Courier New", 8pt italic
font_info = ChooseFont("Courier New", 8, FONT_Italic, ,
CF_PRINTERFONTS | CF_FORCEFONTEXIST)
If font_info Then
    font_name = Field(font_info,"",1)
    font_size = Field(font_info,"",2)
    font_style = Field(font_info,"",3)
EndIf
```

Version

4.0.2 Original

5.1 Added printer parameter

Clipboard Read

Syntax

Clipboard Read *data_var, err_var*

Synonyms

CB RD

Description

The Clipboard Read command reads text from the clipboard to a script variable.

Parameters

The following table describes the parameters of the Clipboard Read command:

Parameter	Description
<i>data_var</i>	The name of the variable to receive the data.
<i>err_var</i>	The name of the variable to store an error condition. See the following table.

Values for err_var

Value	Description
1	Unable to connect to the Clipboard.
2	Unable to allocate memory to transfer the data.
3	Data in the Clipboard is not text.
4	More than the 30 K maximum amount of text is on the Clipboard.

Example

* Send data from the Clipboard to the host computer

```
Clipboard Read paste_data, err
If err=0, Then
Enter "Pasted data" : paste_data
EndIf
```

Related Script Commands

[Clipboard Write](#)

Clipboard Write

Syntax

Clipboard Write *text*, *err_var*

Synonyms

CB WR

Description

The Clipboard Write command writes text to the clipboard.

Parameters

The following table describes the parameters of the Clipboard Write command:

Parameter	Description
<i>text</i>	The text to send to the Clipboard.
<i>err_var</i>	The name of the variable to store an error condition. See the following table.

Values for err_var

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

Example

Write test text to the clipboard

```
Clipboard Write "Testing, 123...", err
If err Then
    MsgBox "Unable to write to Clipboard"
EndIf
```

Related Script Commands

[Clipboard Read](#)

CloseServer

Syntax

CloseServer

Description

The CloseServer command closes the host side of the server. You must use this command to close the server after specifying "Library 'wintsys\lib\server'" and using the server library commands.

The CloseServer command is a Server Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

None

Example

This example is part of the PRODM2.WIS demonstration program.

```
* Cancel Button is used for two purposes, Void the current entry
* and exit the program.
Sub OnCancel()
If ProdM_Mode Then
ClearEntry
Else
Library 'wintsys\lib\server'
CloseServer
DialogBox End ProdM
DialogBox Delete ProdM
Destroy ProdM_Mode
EndIf
EndSub
```

Color

Syntax

Color [*foreground*], [*background*]

Synonyms

CL

Description

The Color command sets the foreground and background colors of subsequent text display on your terminal screen. If foreground or background is omitted, the current value is unchanged.

This command can also be spelled Colour.

Parameters

The following table describes the parameters of the Color command:

Parameter	Description
<i>foreground</i>	The foreground color or number. See the following Colors table for more information.
<i>background</i>	The background color or number. See the following Colors table for the foreground parameter for more information.

Values for foreground

The following constants can be used to make the script more readable

Value	Description
<i>COL_None</i>	(-1) Set screen to use monochrome effects
<i>COL_Black</i>	(0) Black
<i>COL_Blue</i>	(1) Blue
<i>COL_Green</i>	(2) Green

Value	Description
<i>COL_Cyan</i>	(3) Cyan
<i>COL_Red</i>	(4) Red
<i>COL_Magenta</i>	(5) Magenta
<i>COL_Brown</i>	(6) Brown
<i>COL_LightGray</i>	(7) Light Gray
<i>COL_LightGrey</i>	(7) Light Grey
<i>COL_Gray</i>	(8) Dark grey
<i>COL_Grey</i>	(8) Dark gray
<i>COL_LightBlue</i>	(9) Light Blue
<i>COL_LightGreen</i>	(10) Light Green
<i>COL_LightCyan</i>	(11) Light Cyan
<i>COL_LightRed</i>	(12) Light Red
<i>COL_LightMagenta</i>	(13) Light Magenta
<i>COL_Yellow</i>	(14) Yellow
<i>COL_White</i>	(15) White

Example

The following example sets the foreground color to white and background color to blue.

```
* To set the text to white on blue
Color COL_White, COL_Blue
```

The next example sets the foreground color to red (color number 4) and does not change the background color:

```
* To change the foreground color to COL_Red
Color 4
```

CommandBar Add

Syntax

CommandBar Add *name, dock_pos, visible, x, y, w, d, flags, script*

Synonyms

COMMANDBAR AD

Description

This command adds a new command bar (also known as a toolbar) to the list of command bars.

To create a command bar first define it either using the DialogBox script commands or the iconlib script library and then use CommandBar Add to add it to the list of toolbars.

Parameters

The following table describes the parameters of the CommandBar Add command:

Parameter	Description
<i>name</i>	The name of the command bar. This must match the name of the dialog box or iconbar which was used to create it.
<i>dock_pos</i>	The position on the screen to show the commandbar. 0 = top, 1 = bottom. Default 0.
<i>visible</i>	True to show the command bar as it is added
<i>x</i>	Column position in pixels to insert the toolbar into the docking position
<i>y</i>	Row position in pixels to insert the toolbar into the docking position
<i>w</i>	Reserved for future expansion.
<i>d</i>	Reserver for future expansion.
<i>flags</i>	Additional flags. See Below. Default 0

Parameter	Description
<i>script</i>	Script used to create this commandbar. This is used so that the command bar can be restored after the wIntegrate session has been closed down.

Values for flags

Value	Description
<i>0</i>	Normal toolbar
<i>1</i>	The Menubar (defined using the Menu script commands)
<i>2</i>	wIntegrate 3.x Iconbar (only one iconbar can be shown at once).
<i>3</i>	wIntegrate 3.x Basebar (only one basebar can be shown at once).

CommandBar OnRightClick

Syntax

CommandBar OnRightClick *command*

Synonyms

COMMANDBAR OR

Description

This command sets the script command to run when the right mouse button is clicked on the background of the toolbar area. This is set to run the "wintsys\script\cbmenu.wis" script which allows the selection of toolbars when wIntegrate is initially installed.

Parameters

The following table describes the parameters of the CommandBar OnRightClick command:

Parameter	Description
<i>command</i>	The script command to run.

Version

4.0 Original

CommandBar Remove

Syntax

CommandBar Remove *name*

Synonyms

COMMANDBAR RM

Description

This command removes a commandbar from memory.

Parameters

The following table describes the parameters of the CommandBar Remove command:

Parameter	Description
<i>name</i>	The name of the commandbar.

Version

4.0 Original

CommandBar Replace

Syntax

CommandBar Replace *name, new_name*

Synonyms

COMMANDBAR RP

Description

This command replace the command bar with a new one based on the dialog box named *new_name* and renames the command bar to *new_name* .

Parameters

The following table describes the parameters of the CommandBar Replace command:

Parameter	Description
<i>name</i>	The name of the command bar.
<i>new_name</i>	The name of the dialog box to replace the command bar with. This will also be the new name of the command bar.

Version

4.0 Original

CommandBar SetScript

Syntax

CommandBar SetScript *name, script*

Synonyms

COMMANDBAR SS

Description

This command sets the file name of the script that is used to create this command bar.

Parameters

The following table describes the parameters of the CommandBar SetScript command:

Parameter	Description
<i>name</i>	The name of the command bar.
<i>script</i>	The script filename used to create this command bar. If only the leaf name of the script is used it is assumed to be in the wintsys\iconbar directory

Version

4.0 Original

CommandBar Show

Syntax

CommandBar Show *name, show*

Synonyms

COMMANDBAR SH

Description

This command shows or hides a command bar.

Parameters

The following table describes the parameters of the CommandBar Show command:

Parameter	Description
<i>name</i>	The name of the command bar.
<i>show</i>	True shows the command bar, false hides it.

Version

4.0 Original

CommandBarInfo

Syntax

CommandBarInfo *name, flags*

Description

This function returns information about the specified command bar according to the value in the flags parameter. The delimiter between the different pieces of information returned is a comma (unless tab is specified in the flags).

Parameters

The following table describes the parameters of the CommandBarInfo command:

Parameter	Description
<i>name</i>	The name of the command bar
<i>flags</i>	Flags specifying the values to be returned. See Below

Values for flags

The flags parameters can be one or more of the following added together. The default value is 254.

Value	Description
<i>1</i>	The name of the command bar
<i>2</i>	The docking position (0 = top, 1 = bottom)
<i>4</i>	shown or hidden
<i>8</i>	x,y pixel positions
<i>16</i>	w,d reserved values
<i>32</i>	Type (0 = normal, 1 = menu, 2= iconbar, 3=basebar)
<i>64</i>	script used to create the commandbar
<i>1024</i>	Use tab instead of comma as the delimiter

Value	Description
2048	Put quotes around the name and script name

Return Value

The information specified in flags

Version

4.0 Original

CommandBarList

Syntax

CommandBarList [*flags*]

Description

This function returns a cr separated list of information about all the command bars in memory according to the value in the flags parameter. The delimiter between the different pieces of information returned is a tab character.

Parameters

The following table describes the parameters of the CommandBarList command:

Parameter	Description
<i>flags</i>	Flags specifying the values to be returned. See Below

Values for flags

The flags parameters can be one or more of the following added together. The default value is 254.

Value	Description
<i>1</i>	The name of the command bar
<i>2</i>	The docking position (0 = top, 1 = bottom)
<i>4</i>	shown or hidden
<i>8</i>	x,y pixel positions
<i>16</i>	w,d reserved values
<i>32</i>	Type (0 = normal, 1 = menu, 2= iconbar, 3=basebar)
<i>64</i>	script used to create the commandbar
<i>2048</i>	Put quotes around the name and script name

Return Value

cr separated list of the information on the command bars specified by the flags parameter.

Version

4.0 Original

ConfigFile Read

Syntax

ConfigFile Read *var, entry, section, file, [default], [max_size]*

Synonyms

CN RD

Description

The ConfigFile Read command reads a value for an entry from the specified section in a configuration file.

Parameters

The following table describes the parameters of the ConfigFile Read command:

Parameter	Description
<i>var</i>	Name of variable into which to read the entry
<i>entry</i>	Entry name in the configuration file.
<i>section</i>	Section name in the configuration file.
<i>file</i>	Configuration file name.
<i>default</i>	Text to return if entry is not found. ("" if not specified)
<i>max_size</i>	Maximum size of entry to return (default is 255 characters). If the size of the return value is greater than max_size, it is truncated.

Example

The configuration file c:\wintuser.ini has the following section:

```
[User]
Surname=Robertshaw
First Name=David
```

Then the following example script reads the Surname and assigns it to the variable username.

```
* To get the surname (in this case Robertshaw) or Unknown if
* the entry does not exist
ConfigFile Read username, "Surname", "User", "c:\wintuser.ini",
"Unknown"
```

Related Script Commands

[ConfigFile ReadInteger](#), [ConfigFile ReadSection](#), [ConfigFile Write](#)

ConfigFile ReadInteger

Syntax

ConfigFile ReadInteger *var, entry, section, file, [default]*

Synonyms

CN RI

Description

The ConfigFile ReadInteger command reads an integer value for an entry from the specified section in a configuration file. This strips any nonnumeric character from the end of the value.

Parameters

The following table describes the parameters of the ConfigFile ReadInteger command:

Parameter	Description
<i>var</i>	Name of variable into which to read the entry.
<i>entry</i>	Entry name in the configuration file.
<i>section</i>	Section name in the configuration file.
<i>file</i>	Configuration file name.
<i>default</i>	Number to return if entry is not found. (0 if not specified).

Example

The configuration file c:\wintuser.ini has the following section:

```
[Statistics]
Logon Count = 1
* To get the logon count (in this case 1) or 0 if the entry does not
exist
ConfigFile ReadInteger logcount, "Logon Count", "Statistics",
"c:\wintuser.ini", 0
```

Related Script Commands

[ConfigFile Read](#), [ConfigFile ReadSection](#), [ConfigFile Write](#)

ConfigFile ReadSection

Syntax

ConfigFile ReadSection *var, section, file, [max_size]*

Synonyms

CN RS

Description

The ConfigFile ReadSection command reads a carriage return separated list of all the entries for the specified section of a configuration file.

Parameters

The following table describes the parameters of the ConfigFile ReadSection command:

Parameter	Description
<i>var</i>	Name of variable into which to read the section.
<i>section</i>	Section name of the configuration file.
<i>file</i>	Configuration file name of the configuration file.
<i>max_size</i>	Maximum size of value of entry (default 2048 characters). If the size of the return value is greater than max_size, it is truncated.

Example

The following example reads a section in a configuration file. The configuration file c:\wintuser.ini has the following section:

```
[User]
Surname=Robertshaw
First Name=David
* Get list of entries in user section.
* In this case = "Surname":cr:"First Name"
ConfigFile ReadSection user_entries, "User", "c:\wintuser.ini"
```

Related Script Commands

[ConfigFile Read](#), [ConfigFile ReadInteger](#), [ConfigFile Write](#)

ConfigFile Remove

Syntax

ConfigFile Remove *entry, section, file, ok_var*

Synonyms

CN RM

Description

This command removes an entry or an entire section of entries from a file written using ConfigFile Write.

Parameters

The following table describes the parameters of the ConfigFile Remove command:

Parameter	Description
<i>entry</i>	The name of the entry to remove from the file. Set this to "" to remove the entire section from the file
<i>section</i>	The section in which the entry to be removed. If entry is "" then this entire section is removed
<i>file</i>	The filename of the configuration file
<i>ok_var</i>	Variable name to set to true if write succeeds. If omitted, no errors are reported.

Example

Remove the list of groups from the dialer.inf file

```
ConfigFile Remove "", "groups", Dialer.FileName
```

Version

4.1 Original

ConfigFile Write

Syntax

ConfigFile Write *value, entry, section, file, [ok_var]*

Synonyms

CN WR

Description

The ConfigFile Write command writes the value for an entry in the specified section in a configuration file. If the configuration file does not exist, it is created (if the directory exists). If the section or entry is not currently in the configuration file, it is added.

Parameters

The following table describes the parameters of the ConfigFile Write command:

Parameter	Description
<i>value</i>	Entry value to write in the configuration file.
<i>entry</i>	Entry name of the configuration file.
<i>section</i>	Section name of the configuration file.
<i>file</i>	Configuration file name.
<i>ok_var</i>	Variable name to set to true if write succeeds. If omitted, no errors are reported.

Example

The following example creates the following entry in the "c:\wintuser.ini" configuration file:

```
[User]
Surname=Robertshaw
First Name=David
ConfigFile Write "Robertshaw", "Surname", "User", "c:\wintuser.ini",
ok
If ok Then
  fname = "David"
```

```
ConfigFile Write fname, "First Name", "User", "c:\wintuser.ini"  
EndIf
```

Related Script Commands

[ConfigFile Read](#), [ConfigFile ReadInteger](#), [ConfigFile ReadSection](#)

Configure

Syntax

Configure

Synonyms

CF

Description

The Configure command places the current script into configuration mode. When a script is in configuration mode, session variables can be stored without the Store command. When an the Update command is executed, configure mode stops and the values are updated.

See the "Reference - Script Menu Options" in this manual for details of the menu options and variables available.

Parameters

None

Example

This example is taken from the MACIMP.WIS demonstration program.

```
Dialog RunImportFile
Configure
Account=" "
File="WIN.PROGS"
Items="MACHINE.TYPES"
Fields="*"
DosFile="C:\WINTEG\MACTYPES.TXT"
Format="ASC"
Inform="0"
Timeout="5"
.
.
.
Update
```

Related Script Commands

[Set](#), [Store](#), [Update](#)

Convert

Syntax

Convert *var_name, from_chars, to_chars*

Synonyms

CV

Description

The Convert command changes all occurrences of a substring in a variable to another substring.

Parameters

The following table describes the parameters of the Convert command:

Parameter	Description
<i>var_name</i>	The name of the variable to search.
<i>from_chars</i>	The original characters to convert. If the from_chars characters are longer than the to_chars, the extra characters are stripped from the variable.
<i>to_chars</i>	The replacement characters.

Example

The following example is taken from the PARSE.WIS demonstration program.

```
Convert line, tab, ";"* Remove tabs for this test
```

Related Script Commands

[Change](#)

Count

Syntax

Count *string, character*

Description

The Count function returns the number of occurrences of a single character in a string.

Parameters

The following table describes the parameters of the Count command:

Parameter	Description
<i>string</i>	The text string to search.
<i>character</i>	The character that you specify to count.

Return Value

The number of occurrences of the substring

Example

The following example is taken from the PARSE.WIS demonstration program.

```
text = ScreenText(x1,y1,x2,y2,type,b1,b2)
Convert text, lf, "";* Remove linefeeds from returned text
nolines = Count(text, cr) + 1
```

CreateIconBar

Syntax

CreateIconBar *name, no_of_rows*

Description

The CreateIconBar command starts the creation of an icon bar. You must use EndCreateIconBar command to stop creation mode.

The CreateIconBar command is a Icon Library command. For more information, see "Appendix A - Scripting Libraries and Modules".

Parameters

The following table describes the parameters of the CreateIconBar command:

Parameter	Description
<i>name</i>	The name of the new icon bar.
<i>no_of_rows</i>	The number of rows for the icon bar. For instance, the wIntegrate general icon bar is one row deep; the technical icon bar is two rows deep.

Example

The following example is from the C:\winteg\wintsys\iconbar\bar_gen.wis script.

```
* General icon bar
* Take pos variable from command line which executed the script
* I = IconBar (default, at top)
* B = BaseBar (at bottom)
pos = CommandLine
If IsShown(General_bar) Then EndScript
If Not(IsDialog(General_bar)) Then
Library 'wintsys\lib\iconlib'
CreateIconBar General_bar,1
AddIcon 'image\i_open.bmp',"Show FileOpen"
AddIcon 'image\i_save.bmp',"Show FileSaveAs"
AddIcon 'image\i_copy.bmp',"Invoke EditCopy"
AddIcon 'image\i_paste.bmp',"Invoke EditPaste"
AddGap
```

```
.  
.  
EndCreate  
EndIf  
If pos = "B" Then  
DialogBox Basebar General_bar  
Else  
DialogBox Iconbar General_bar  
EndIf
```

Related Script Commands

[AddGap](#), [AddIcon](#), [AddLargeIcon](#), [AddWideIcon](#), [EndCreateIconBar](#), [NextRow](#)

CreateToolBar

Syntax

CreateToolBar *name, no_of_columns, no_of_rows, size*

Description

The CreateToolBar command starts creation of a tool bar. The CreateToolBar command is a Tool Library command. For more information, see "Appendix A - Scripting Libraries and Modules".

Parameters

The following table describes the parameters of the CreateToolBar command:

Parameter	Description
<i>name</i>	The name for the new tool bar.
<i>no_of_columns</i>	The number of columns for the tool bar.
<i>no_of_rows</i>	The number of rows for the tool bar.
<i>size</i>	The size of the icons to use. Specify the 16x16 default size with "" (null), or the large (32x32) with "L".

Example

The following example demonstrates the use of the CreateToolBar command:

```
Library 'wintsys\lib\toollib'  
CreateToolBar BigTool_bar,1,6,"L"  
AddTool 'icon\excel.ico','Dialog RunProgram;Set Filename =  
"excel.exe";Set  
Arguments = "";Invoke'  
AddTool 'icon\word.ico','Dialog RunProgram;Set Filename =  
"winword.exe";Set  
Arguments = "";Invoke'
```

Related Script Commands

[AddTool](#), [AddToolTip](#), [ToolSpace](#)

Cursor

Syntax

Cursor *param*, [*option*]

Description

The Cursor function returns cursor related information, depending on the option.

Parameters

The following table describes the parameters of the Cursor command:

Parameter	Description
<i>param</i>	The data returned depends on the option. See the following table for more information on available options.
<i>option</i>	Sub option. See below for use with param = 3

Values for param

Value	Description
<i>0 or Get_X</i>	The cursor column position coordinate.
<i>1 or Get_Y</i>	The cursor row position coordinate.
<i>2</i>	Returns "True" if there is a scroll region.
<i>3</i>	Gives information about a scroll region or the screen if there is no scroll region.
<i>4</i>	Returns the font width in pixels.
<i>5</i>	Returns the font depth in pixels.
<i>6</i>	Gets or sets the current cursor position attributes. The return value from this function is passed as a parameter to a later call to this function.

Value	Description
7	Returns the row depth in pixels. This is the total depth used by a line of text and consists of the font depth plus any extra line space.

Values for option

The following options are available when param is set to 3

Value	Description
1	Returns the left column coordinate.
2	Returns the top row coordinate.
3	Returns the right column coordinate.
4	Returns the bottom row coordinate.
5	Returns the width of the scroll region.
6	Returns the depth of the scroll region.

Return Value

The information required as specified by the param and option parameters

Example

The following example is taken from the DISPBOX.WIS demonstration program.

```
ScreenWidth = Cursor(3,5)
ScreenDepth = Cursor(3,6)
```

CursorShape

Syntax

CursorShape *newshape*

Synonyms

CS

Description

The CursorShape command changes the current shape of the mouse cursor when it is over the terminal window.

The wIntegrate cursors can be overridden by creating a Dynamic Link library called wcursor.dll.

Parameters

The following table describes the parameters of the CursorShape command:

Parameter	Description
<i>newshape</i>	The new shape to use for the cursor

Values for newshape

Value	Description
<i>1</i>	Arrow - standard mouse pointer.
<i>2</i>	Upward pointing arrow.
<i>3</i>	Cross hair cursor.
<i>4</i>	Standard windows I-beam cursor.
<i>5</i>	Hourglass cursor Cursor Shapes.
<i>"Hand"</i>	Cursor hand shape

Example

The following example changes the cursor from the default block cursor to a cross-hair cursor.

```
* Turn cursor into a cross hair cursor  
CursorShape 3
```

Date

Syntax

Date

Description

The Date function returns the current internal date. Internal format is the number of days after December 31, 1967. This format is used by most host systems.

Parameters

None

Return Value

Today's internal date

Example

The following example displays a message box with the internal date.

```
* Give the internal date in a message box  
MessageBox "The internal date is " : Date()
```

Related Script Commands

[Iconv](#), [Oconv](#), [Stamp](#), [Time](#)

DCount

Syntax

DCount *Value, delimiter*

Description

This function returns the number of fields separated by the single character delimiter in a string.

Parameters

The following table describes the parameters of the DCount command:

Parameter	Description
<i>Value</i>	The string to count the fields in
<i>delimiter</i>	The single character that delimits the fields

Return Value

The return value is the number of delimited fields in the value. This is zero if the value or the delimiter are empty.

Example

Check the number of lines inserted into a variable

```
Ins val,-1,"Line 1"  
Ins val,-1,"Line 2"  
Ins val,-1,"Line 3"  
  
If Dcount(Val,cr) = 3 Then  
    MessageBox "There are three lines inserted"  
Else  
    MessageBox "Incorrect number of lines:"&Dcount(val,cr)  
EndIf
```

Version

4.1.1 Original

DDE Advise

Syntax

DDE Advise *name, server_item, global_var_name*

Synonyms

DDE AD

Description

The DDE Advise command establishes a hot link between wIntegrate and the DDE server.

Parameters

The following table describes the parameters of the DDE Advise command:

Parameter	Description
<i>name</i>	The name that you gave to the DDE conversation in DDE Initiate.
<i>server_item</i>	This name is dependent on the other Windows application
<i>global_var_name</i>	Any wIntegrate variable defined as global.

Example

The following example is taken from a host demonstration program that calls script commands.

It uses DDE to link wIntegrate and Visual Basic.
You can run this program from wIntegrate at the database prompt by entering: WIN.FRUIT

```
* WIN.FRUIT
* Example of a DDE Link to a visual basic program
* Compile for: GENERIC AP MD ME PI PR SQ UD UL UP UV UC IN
* Copyright (c) 1991-93. Impact Business Systems
*
PROMPT ' '
*
OPEN ' ','FRUIT' TO F.FRUIT ELSE
```

```
PRINT "This program needs a small file called 'FRUIT'"
PRINT "Please create this file and re-run the program."
PRINT
STOP
END
*
DIM NAME(3)
NAME(1) = 'USA'
NAME(2) = 'CANADA'
NAME(3) = 'UK'
*
PRINT @(-1) : "WIN.FRUIT" : @(30,0) : "Fruit server" : @(71,0) :
OCONV(DATE(),
"D2")
PRINT @(0,2):"This program uses wIntegrate as a DDE server to a Visual
Basic pro-gram"
PRINT "fruity.exe. Data is held in a simple host file called FRUIT
which con-tains"
PRINT "the number of apples, oranges and bananas eaten in three
countries."
PRINT "You can drag and drop fruit onto countries, and retrieve/update
the host."
PRINT
PRINT "This program requires Visual Basic 2.0 support DLLs"
*
GOSUB 300;* Start up fruity
*
PRINT @(24,18):"Enter EXIT to stop this program" :
*
CALL WIN.COLOR("Yellow","Blue")
CALL WIN.BOX(20,9,60,16,1)
PRINT @(23,9):"Data sent from Visual Basic program":
CALL WIN.COMSUB("Screen ScrollRegion 21,10,59,15")
*
CALL WIN.COMSUB('DDE Initiate fruit,"Fruity","Fruit"')
CALL WIN.COMSUB('DDE Advise fruit,"action",fruit_action')
CALL WIN.COMSUB("DDE HostLink fruit, '%d':cr")
*
* Wait for action to change
LOOP
INPUT ACTION
ACTION = OCONV(ACTION, "MCU")
*
BEGIN CASE
CASE ACTION = "READ" ; GOSUB 100
CASE ACTION = "WRITE"; GOSUB 200
END CASE
*
```

```
UNTIL ACTION = "EXIT" OR ACTION = "CANCEL" DO
REPEAT
*
CALL WIN.COMSUB("Screen ScrollRegion")
CALL WIN.COLOR("Off","")
PRINT @(0,20):
*
UNTIL ACTION = "EXIT" OR ACTION = "CANCEL" DO
REPEAT
*
CALL WIN.COMSUB("Screen ScrollRegion")
CALL WIN.COLOR("Off","")
PRINT @(0,20):
IF ACTION = "EXIT" THEN
CALL WIN.COMSUB('DDE Unadvise fruit,"action"')
END
CALL WIN.COMSUB("DDE Terminate fruit")
*
.
.
.
```

Related Script Commands

[DDEChanged](#), [DDE Execute](#), [DDE HostLink](#), [DDE Initiate](#), [DDE Poke](#),
[DDERequest](#), [DDE Terminate](#), [DDE Timeout](#), [DDE Unadvise](#)

DDE Execute

Syntax

DDE Execute *name, command, [status_var]*

Synonyms

DDE EX

Description

The DDE Execute command sends a command to run on the server. DDE runs the command on the linked server. The value of command is dependent on the server application. If the response variable is included on the command line, the script monitor does not appear. If the DDE Execute fails instead, the Response variable is set to a value.

Parameters

The following table describes the parameters of the DDE Execute command:

Parameter	Description
<i>name</i>	The name of the DDE conversation designated in DDE Initiate.
<i>command</i>	The server command to execute.
<i>status_var</i>	The status of the result of server command. See the following Status table.

Values for status_var

Value	Description
0	Command completed successfully
2	Server is busy
4	Server communication failure
5	Server communication failure
6	Server command timed out

Value	Description
7	Invalid parameter
8	Free memory is low
9	Memory allocation error
10	Server command failed
11	Conversation not established with Server
13	Server communication failure
14	Server communication failure
15	The Server is no longer active
16	Server communication failure
18	Invalid parameter

Example

The following example is taken from the COPYMENU.WIS demonstration program.

```
If IsApp("excel.exe") Then
Dialog EditCopyTo
Set Format = "Table"
Set CopyTo = "Clipboard"
Invoke EditCopySpecial
DDE Initiate excel_link,"excel","system", ok
If ok Then
DDE Execute excel_link,['PASTE()']
DDE Terminate excel_link
EndIf
EndIf
EndSub
```

Related Script Commands

[DDE Advise](#), [DDE Changed](#), [DDE HostLink](#), [DDE Initiate](#), [DDE Poke](#), [DDE Request](#),
[DDE Terminate](#), [DDE Timeout](#), [DDE Unadvise](#)

DDE HostLink

Syntax

DDE HostLink *name*, [*return_data*]

Synonyms

DDE HL

Description

The DDE HostLink command sends data to the host when DDE Advise changes. The first time this command is used, any changes since the DDE Advise command are sent to the host.

Parameters

The following table describes the parameters of the DDE HostLink command:

Parameter	Description
<i>name</i>	The name of the DDE conversation, designated in DDE Initiate.
<i>return_data</i>	The format of the data to send to the host. This parameter can contain any characters combined with these special sequences: %d - Change to the new value of the data. %i - Change to the server item name. %v - Change to the global variable name linked to the server item in DDE Advise.

Example

The following example is taken from a host demonstration program that calls script commands. It uses DDE to link wIntegrate and Visual Basic.

You can run this program from wIntegrate at the database prompt by entering: WIN.FRUIT

```
* WIN.FRUIT
* Example of a DDE Link to a visual basic program
* Compile for: GENERIC AP MD ME PI PR SQ UD UL UP UV UC IN
* Copyright (c) 1991-93. Impact Business Systems
```

```
*
PROMPT ' '
*
OPEN '','FRUIT' TO F.FRUIT ELSE
PRINT "This program needs a small file called 'FRUIT'"
PRINT "Please create this file and re-run the program."
PRINT
STOP
END
*
DIM NAME(3)
NAME(1) = 'USA'
NAME(2) = 'CANADA'
NAME(3) = 'UK'
*
.
.
.
*
GOSUB 300;* Start up fruity
*
PRINT @(24,18):"Enter EXIT to stop this program" :
*
CALL WIN.COLOR("Yellow","Blue")
CALL WIN.BOX(20,9,60,16,1)
PRINT @(23,9):"Data sent from Visual Basic program":
CALL WIN.COMSUB("Screen ScrollRegion 21,10,59,15")
*
CALL WIN.COMSUB('DDE Initiate fruit,"Fruity","Fruit"')
CALL WIN.COMSUB('DDE Advise fruit,"action",fruit_action')
CALL WIN.COMSUB("DDE HostLink fruit, '%d':cr")
```

Related Script Commands

[DDE Advise](#), [DDEChanged](#), [DDE Execute](#), [DDE Initiate](#), [DDE Poke](#), [DDERequest](#), [DDE Terminate](#), [DDE Timeout](#), [DDE Unadvise](#)

DDE Initiate

Syntax

DDE Initiate *name, server_name, server_topic, [ok_var]*

Synonyms

DDE IN

Description

The DDE Initiate command starts a "conversation" with another Windows application. This command must be used before any other DDE commands, because the name of the conversation is set in this command, then used by other commands. To end the DDE conversation, use DDE Terminate. The DDE Poke command changes the value of a specified server item.

Parameters

The following table describes the parameters of the DDE Initiate command:

Parameter	Description
<i>name</i>	The name you give to the DDE conversation. This name is used to refer to the conversation in other DDE commands.
<i>server_name</i>	The name of the other application to communicate with.
<i>server_topic</i>	This is dependent on the other application; common topics are a file-name and system.
<i>ok_var</i>	This variable returns "True" if the conversation is started.

Example

The following example is taken from the COPYMENU.WIS demonstration program.

```
If IsApp("excel.exe") Then
Dialog EditCopyTo
Set Format = "Table"
Set CopyTo = "Clipboard"
```

```
Invoke EditCopySpecial
DDE Initiate excel_link,"excel","system", ok
If ok Then
DDE Execute excel_link,['PASTE()']
DDE Terminate excel_link
EndIf
EndIf
EndSub
```

The following example is taken from the DDE_SCR.WIS demonstration program.

```
* DDE_SCR
* Script to create DDE links to spreadsheet
Library "wintsys\lib\server"
If Not(OK) Then
MessageBox "Excel not started, or SHEET1.XLS not open"
EndScript
EndIf
DDE Poke SpreadSheet, "r2c2", "Data sent by wIntegrate"
GetAndSend "WIN.PROGS", "MACHINE.TYPE", 1, SpreadSheet, "r4c2"
GetAndSend "MD", "WHO", 1, SpreadSheet, "r5c2"
CloseServer
EndScript
Sub GetAndSend(HostFile, HostItem, HostAtt, $LinkName, LinkItem)
HReadv Value, HostFile, HostItem, HostAtt
If Server_Error Then
MessageBox ServerErrMsg(Server_Error) : " " : HostFile : ", " :
HostItem
Else
DDE Poke $LinkName, LinkItem, Value
EndIf
EndSub
```

Related Script Commands

[DDE Advise](#), [DDEChanged](#), [DDE Execute](#), [DDE HostLink](#), [DDE Poke](#),
[DDERequest](#), [DDE Terminate](#), [DDE Timeout](#), [DDE Unadvise](#), [DDE Poke](#), [DDE](#)
[Advise](#), [DDEChanged](#), [DDE Execute](#), [DDE HostLink](#), [DDE Initiate](#), [DDERequest](#),
[DDE Terminate](#), [DDE Timeout](#), [DDE Unadvise](#)

DDE Poke

Syntax

DDE Poke *name, server_item, data*

Synonyms

DDE PK

Description

The DDE Poke command changes the value of a specified server item.

Parameters

The following table describes the parameters of the DDE Poke command:

Parameter	Description
<i>name</i>	The name of the DDE conversation, designated in DDE Initiate.
<i>server_item</i>	The name of the server item to change. For instance, a server item can be a cell in an Excel table.
<i>data</i>	The new data to replace the server item.

Example

The following example is taken from the DDE_SCR.WIS demonstration program.

```
* DDE_SCR
* Script to create DDE links to spreadsheet
Library "wintsys\lib\server"
DDE Initiate SpreadSheet, "excel", "sheet1", OK
If Not(OK) Then
  MessageBox "Excel not started, or SHEET1.XLS not open"
EndScript
EndIf
DDE Poke SpreadSheet, "r2c2", "Data sent by wIntegrate"
GetAndSend "WIN.PROGS", "MACHINE.TYPE", 1, SpreadSheet, "r4c2"
GetAndSend "MD", "WHO", 1, SpreadSheet, "r5c2"
DDE Terminate SpreadSheet
CloseServer
```

```
EndScript

Sub GetAndSend(HostFile, HostItem, HostAtt, $LinkName, LinkItem)
HReadv Value, HostFile, HostItem, HostAtt
If Server_Error Then
  MsgBox ServerErrMsg(Server_Error) : " " : HostFile : ", " :
  HostItem
Else
  DDE Poke $LinkName, LinkItem, Value
EndIf
EndSub
```

Related Script Commands

[DDE Advise](#), [DDEChanged](#), [DDE Execute](#), [DDE HostLink](#), [DDE Initiate](#),
[DDERequest](#), [DDE Terminate](#), [DDE Timeout](#), [DDE Unadvise](#)

DDE Terminate

Syntax

DDE Terminate *name*

Synonyms

DDE TR

Description

The DDE Terminate command stops a connection started by DDE Initiate. You can specify "All" as the name parameter to end all current DDE conversations.

Parameters

The following table describes the parameters of the DDE Terminate command:

Parameter	Description
<i>name</i>	The name of the DDE conversation designated in DDE Initiate.

Example

The following example is a part of the COPYMENU.WIS demonstration program.

```
If IsApp("excel.exe") Then
Dialog EditCopyTo
Set Format = "Table"
Set CopyTo = "Clipboard"
Invoke EditCopySpecial
DDE Initiate excel_link,"excel","system", ok
If ok Then
DDE Execute excel_link,['PASTE()']
DDE Terminate excel_link
EndIf
EndIf
EndSub
```

Related Script Commands

[DDE Advise](#), [DDEChanged](#), [DDE Execute](#), [DDE HostLink](#), [DDE Initiate](#), [DDE Poke](#), [DDERequest](#), [DDE Timeout](#), [DDE Unadvise](#)

DDE Timeout

Syntax

DDE Timeout *name, timeout*

Synonyms

DDE TM

Description

The DDE Timeout command sets a timeout for a DDE connection.

Parameters

The following table describes the parameters of the DDE Timeout command:

Parameter	Description
<i>name</i>	The name of the DDE conversation, designated in DDE Initiate.
<i>timeout</i>	The number of centiseconds to wait for a response from the host for actions such as DDE Execute, DDERequest, DDE Poke, etc. For instance, the value for five seconds can be "500" or "5s".

Example

In the following example, the timeout is set for 30 seconds:

```
DDE Timeout Conv1 "30s"
```

Related Script Commands

[DDE Advise](#), [DDEChanged](#), [DDE Execute](#), [DDE HostLink](#), [DDE Initiate](#), [DDE Poke](#), [DDERequest](#), [DDE Terminate](#), [DDE Unadvise](#), [DDE Unadvise](#), [DDE Advise](#), [DDEChanged](#), [DDE Execute](#), [DDE HostLink](#), [DDE Initiate](#), [DDE Poke](#), [DDERequest](#), [DDE Terminate](#), [DDE Timeout](#)

DDE Unadvise

Syntax

DDE Unadvise *name, server_item*

Synonyms

DDE UN

Description

This command closes a server link opened by DDE Advise.

Parameters

The following table describes the parameters of the DDE Unadvise command:

Parameter	Description
<i>name</i>	The name of the DDE conversation, designated in DDE Initiate.
<i>server_item</i>	The server item name that was monitored by DDE Advise.

Example

The following example is taken from a host demonstration program that calls script commands.

It uses DDE to link wIntegrate and Visual Basic.
You can run this program from wIntegrate at the database prompt by
entering: WIN.FRUIT

```
* WIN.FRUIT
* Example of a DDE Link to a visual basic program
* Compile for: GENERIC AP MD ME PI PR SQ UD UL UP UV UC IN
* Copyright (c) 1991-93. Impact Business Systems
*
PROMPT 145146
*
OPEN 145146,146FRUIT146 TO F.FRUIT ELSE
PRINT "This program needs a small file called 145FRUIT146"
PRINT "Please create this file and re-run the program."
PRINT
```

```
STOP
END
*
DIM NAME(3)
NAME(1) = 145USA146
NAME(2) = 145CANADA146
NAME(3) = 145UK146
*
.
.
.
*
GOSUB 300;* Start up fruity.exe
IF NOT(RUNNING) THEN STOP
*
PRINT @(24,18):"Enter EXIT to stop this program" :
*
CALL WIN.COLOR("Yellow","Blue")
CALL WIN.BOX(20,9,60,16,1)
PRINT @(23,9):"Data sent from Visual Basic program":
CALL WIN.COMSUB("Screen ScrollRegion 21,10,59,15")
*
CALL WIN.COMSUB(145DDE Initiate fruit,"Fruity","Fru"148146)
CALL WIN.COMSUB(145DDE Advise fruit,"action",fruit_action146)
CALL WIN.COMSUB("DDE HostLink fruit, 145%d146:cr")
*
* Wait for action to change
LOOP
INPUT ACTION
ACTION = OCONV(ACTION, "MCU")
*
BEGIN CASE
CASE ACTION = "READ" ; GOSUB 100
CASE ACTION = "WRITE"; GOSUB 200
END CASE
*
UNTIL ACTION = "EXIT" OR ACTION = "CANCEL" DO
REPEAT
*
CALL WIN.COMSUB("Screen ScrollRegion")
CALL WIN.COLOR("Off","")
PRINT @(0,20):
IF ACTION = "EXIT" THEN
CALL WIN.COMSUB(145DDE Unadvise fruit,"action"146)
END
CALL WIN.COMSUB("DDE Terminate fruit")
*
STOP
```

Related Script Commands

[DDE Advise](#), [DDEChanged](#), [DDE Execute](#), [DDE HostLink](#), [DDE Initiate](#), [DDE Poke](#), [DDERequest](#), [DDE Terminate](#), [DDE Timeout](#)

DDEChanged

Syntax

DDEChanged *name*, [*server_item*]

Description

The DDEChanged function determines if a DDE server item has changed since it was last called. DDEChanged returns "True" if item has changed.

Parameters

The following table describes the parameters of the DDEChanged command:

Parameter	Description
<i>name</i>	The name of the DDE conversation, designated in DDE Initiate.
<i>server_item</i>	The name of the server item to check. For instance, a server item can be a cell in an Excel table.

Return Value

Returns True (1) if the server item has changed, otherwise False (0).

Example

The following example obtains new Excel data if the cell has changed.

```
IF DDEChanged(excel_link, "R12C4") Then  
  new_data = DDERequest(excel_link, "R12C4")  
EndIf
```

Related Script Commands

[DDE Advise](#), [DDE Execute](#), [DDE HostLink](#), [DDE Initiate](#), [DDE Poke](#), [DDERequest](#), [DDE Terminate](#), [DDE Timeout](#), [DDE Unadvise](#)

DDERequest

Syntax

DDERequest *name, server_item*

Description

The DDERequest function retrieves data from an open DDE connection.

Parameters

The following table describes the parameters of the DDERequest command:

Parameter	Description
<i>name</i>	The name of the DDE conversation, designated in DDE Initiate.
<i>server_item</i>	The name of the server item to check. For instance, a server item can be a cell in an Excel table.

Return Value

Data retrieved from the DDE connection

Example

The following example obtains new Excel data if the cell has changed.

```
IF DDEChanged(excel_link, "R12C4") Then  
  new_data = DDERequest(excel_link, "R12C4")  
EndIf
```

Related Script Commands

[DDE Advise](#), [DDEChanged](#), [DDE Execute](#), [DDE HostLink](#), [DDE Initiate](#), [DDE Poke](#), [DDE Terminate](#), [DDE Timeout](#), [DDE Unadvise](#)

Debug

Syntax

Debug *option*

Description

The Debug command calls the Script Monitor to use for debugging purposes. For more information on the Script Monitor, see "Appendix C - The Script Monitor".

Parameters

The following table describes the parameters of the Debug command:

Parameter	Description
<i>option</i>	See below.

Values for option

Value	Description
<i>0</i>	Turns debugging off.
<i>1</i>	Calls the Script Monitor for the current script.
<i>2</i>	Calls the Script Monitor for any script that is started.
<i>3</i>	Calls the Script Monitor for the current and future scripts.
<i>4</i>	Break all running scripts except this script.
<i>5</i>	Break all running scripts including this script.
<i>8</i>	Prevent break key from affecting this script

Version

5.0 Option 4 and 8

Define

Syntax

Define *func_name*([*arg1*], [*arg2*], ... [*argN*])

Description

The Define command begins the definition of a user-written function. A user-written function must be placed after an EndScript command is used to exit the script. You must end your user-written function with the Return command.

Note Variables that only exist within this function should be created with the Local command.

Parameters

The following table describes the parameters of the Define command:

Parameter	Description
<i>func_name</i>	The name of the function
<i>arg1, arg2 ... argN</i>	Variables which are initialized with the values passed to the function. If the argument begins with \$, then the variable is set with a name, so that an argument can be passed by reference and the original variable can be changed.

Example

The following example determines the higher of two numbers using a user-written function.

```
var = 3
a = max(var,7)
EndScript
Define max(a,b)
Local result
If a > b Then
result = a
Else
result = b
EndIf
```

Return result

The next example is a part of the PRODM2.WIS demonstration program.

```
* NotCancel, checks if Cancel button has been clicked
* If it has it overrides the validation and returns False
Define NotCancel()
If DialogNextEvent(ProdM) = "C;Cancel" Then
DB InputOK ProdM, True
return False
EndIf
return True
```

Related Script Commands

[EndScript](#), [EndSub](#), [Local](#), [Return](#), [Sub](#)

Del

Syntax

Del *var_name*, *line*, [*tab*]

Description

The Del command deletes the data at a specified line or tab position in a variable. If the tab parameter does not exist, the entire line is deleted.

Parameters

The following table describes the parameters of the Del command:

Parameter	Description
<i>var_name</i>	The name of the variable whose value is to be modified
<i>line</i>	The line number to delete
<i>tab</i>	The tab number to delete. If this is omitted the whole line is deleted

Example

This example is taken from the PARSE.WIS demonstration program.

```
j = noline$
maxval = 1
maxlen = 3
Loop
line = Trim(Extract(text,j))
If Extract(line,1,2) > maxval Then maxval = Extract(line,1,2)
If Length(Extract(line,1,1)) > maxlen Then maxlen =
Length(Extract(line,1,1))
Convert line, tab, "" ;* Remove tabs for this test
If line = "" Then
Del text, j
noline$ -= 1
EndIf
```

Related Script Commands

[Extract](#), [Ins](#), [Rep](#)

Destroy

Syntax

Destroy *var_name*, [*var_name2*], ..., [*var_name5*]

Synonyms

DS

Description

The Destroy command removes up to five global variables from memory.

Tip You can use the Global command to create up to five global variables.

Parameters

The following table describes the parameters of the Destroy command:

Parameter	Description
<i>var_name</i>	The name of the first global variable to delete.
<i>var_name2</i>	The second global variables to delete
...	The third and fourth variable names
<i>var_name5</i>	The fifth variable to destroy. A maximum of five variables can be destroyed in one Destroy command

Example

The following example is part of the PRODM2.WIS demonstration program.

```
* Cancel Button is used for two purposes, Void the current entry
* and exit the program.
Sub OnCancel()
If ProdM_Mode Then
ClearEntry
Else
Library 'wintsys\lib\server'
CloseServer
DialogBox End ProdM
DialogBox Delete ProdM
```

```
* Tidy up global variables  
Destroy ProdM_Mode  
EndIf  
EndSub
```

Related Script Commands

[Global](#)

Dialog

Syntax

Dialog *name*

Synonyms

DL

Description

The Dialog command sets the menu command for future Configure, Invoke, Set, Show, and Store commands and the Get function.

See the "Reference - Script Menu Options" in this manual for details of the menu options and variables available.

Parameters

The following table describes the parameters of the Dialog command:

Parameter	Description
<i>name</i>	The name of the dialog. See Appendix for details

Example

The following example demonstrates the use of the Dialog command:

```
* Copy Text to Notepad
Sub CopyTextNotepad()
If Not(IsApp("notepad.exe")) Then
Dialog RunProgram
Set Filename = 'notepad.exe'
Set Arguments = ''
Invoke
EndIf
```

Related Script Commands

[Configure](#), [Get](#), [Invoke](#), [Set](#), [Show](#), [Store](#)

DialogBox AddToList

Syntax

DialogBox AddToList *name*, *var_name*, *string*, [*position*]

Synonyms

DB AL

Description

The DialogBox AddToList command adds a string to a list box or combo box.

Parameters

The following table describes the parameters of the DialogBox AddToList command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox Create or DialogBox New.
<i>var_name</i>	The name of the control, which must be a list box or combo box.
<i>string</i>	The string to be added.
<i>position</i>	The position to insert the string. If you do not specify a position, the string is added to the end of the list.

Related Script Commands

[DialogBox ComboBox](#), [DialogBox DeleteFromList](#), [DialogIndex](#), [DialogList](#), [DialogListFind](#), [DialogListFind](#), [DialogBox Select](#)

DialogBox Animate

Syntax

DialogBox Animate *dlg_name, ctrl_name, x, y, width, depth, [style]*

Synonyms

DB AN

Description

The DialogBox Animate control plays back an AVI (Audio Video Interleave - Microsoft's Video for Windows standard) file in the control. The AVI file must meet the following requirements:

There must be exactly one video stream and it must have at least one frame.

There can be at most two streams in the file (typically the other stream, if present, is an audio stream, although the animation control ignores audio information).

The clip must either be uncompressed or compressed with RLE8 compression.

No palette changes are allowed in the video stream.

See the "Reference - Script Controls" chapter for details of the styles, properties, methods and events for this control.

Parameters

The following table describes the parameters of the DialogBox Animate command:

Parameter	Description
<i>dlg_name</i>	The name of the dialog.
<i>ctrl_name</i>	The name of the control
<i>x</i>	Specifies the column position for displaying the control in dialog box units.
<i>y</i>	Specifies the row position for displaying the control in dialog box units.

Parameter	Description
<i>width</i>	Specifies the width to display the control in dialog box units.
<i>depth</i>	Specifies the depth or height to display the control in dialog box units.
<i>style</i>	The style of the control. See below. The default is WS_VISIBLE.

Values for style

The style can be no style or a combination of the following:

Value	Description
<i>ACS_AUTOPLAY</i>	Automatically start playing the control.
<i>ACS_CENTER</i>	Center animation in control (otherwise control is expanded to fit the animation).
<i>ACS_TRANSPARENT</i>	Background of animation is transparent.
<i>WS_DISABLED</i>	Disables the animation.
<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).

DialogBox AxControl

Syntax

DialogBox AxControl *dlg_name, ctrl_name, prog_id, x, y, width, depth, [style]*

Synonyms

DB AX

Description

The DialogBox AxControl creates an Active X control.

Note: Available in wIntegrate 98 or later releases.

Tip: You can also create an activeX control by specifying the class ID in {} in the class section of a DialogBox Control statement.

Note: For information on how to process events from an activeX control, see DialogBox OnEvent.

See the "Reference - Script Controls" chapter for details of the Active X controls bundled with wIntegrate.

Parameters

The following table describes the parameters of the DialogBox AxControl command:

Parameter	Description
<i>dlg_name</i>	The name of the dialog.
<i>ctrl_name</i>	The name of the control
<i>prog_id</i>	The Program ID or Class ID of the Active X control.
<i>x</i>	Specifies the column position for displaying the control in dialog box units.
<i>y</i>	Specifies the row position for displaying the control in dialog box units.

Parameter	Description
<i>width</i>	Specifies the width to display the control in dialog box units.
<i>depth</i>	Specifies the depth or height to display the control in dialog box units.
<i>style</i>	Style of the control. See below.

Values for style

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

[DialogBox Control](#), [DialogBox OnEvent](#)

DialogBox Background

Syntax

DialogBox Background *name, back_color*

Synonyms

DB BG

Description

This command sets the default background color for a dialog box and its controls.

Parameters

The following table describes the parameters of the DialogBox Background command:

Parameter	Description
<i>name</i>	The name of the dialog box
<i>back_color</i>	The new background color for the dialog box.

DialogBox BaseBar

Syntax

DialogBox BaseBar *name*

Synonyms

DB BB

Description

The DialogBox BaseBar command registers a dialog box as the base bar. If a base bar is already shown, it will change to the one you name here.

This command is kept for backward compatibility with older versions of wIntegrate.

Newer versions should use the CommandBar script commands for defining iconbars.

If you choose Preferences from the Setup Menu, you can select multiple toolbars rather than turning on/off one base bar.

Parameters

The following table describes the parameters of the DialogBox BaseBar command:

Parameter	Description
<i>name</i>	The name of the dialog box to make into a base bar

Related Script Commands

[DialogBox IconBar](#)

DialogBox Caption

Syntax

DialogBox Caption *name, text*

Synonyms

DB CP

Description

The DialogBox Caption command specifies the title for a dialog box. The dialog box style must be one that includes a caption.

Parameters

The following table describes the parameters of the DialogBox Caption command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox Create or DialogBox New.
<i>text</i>	The text for the caption.

Example

The following example creates the caption "Customer Lookup":

```
DialogBox Create Cust 16, 2, 236, 120  
Caption "Customer Lookup"
```

Related Script Commands

[DialogBox Create](#), [DialogBox Style](#)

DialogBox CheckBox

Syntax

DialogBox CheckBox [*name*], *text*, *var_name*, *x*, *y*, *width*, *depth*, [*style*]

Synonyms

DB CH

Description

The DialogBox CheckBox command 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.

Tip: When you don't want the user to be able to select more than one of several options, use radio buttons in a group box.

See the "Reference - Script Controls" chapter for details of the styles, properties, methods and events for this control.

Parameters

The following table describes the parameters of the DialogBox CheckBox command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New. (If you used DialogBox Create, you do not need this parameter.)
<i>text</i>	Specifies the text for the check box caption.
<i>var_name</i>	The name you specify for the check box.
<i>x</i>	Specifies the check box column position.
<i>y</i>	Specifies the check box row position.
<i>width</i>	Defines the check box width. The default width is 40.
<i>depth</i>	Defines the check box depth. The default is 14.

Parameter	Description
<i>style</i>	Style for the control. See the chapter "Reference - Script Controls" for details.

Example

The following example is taken from the PRESTORE.WIS demonstration program.

```
DialogBox Create Prestore,0,0,220,12
Style WS_CAPTION|WS_SYSMENU|WS_MINIMIZEBOX
Font 8,"helv"
Caption "Prestore"
LoseFocus True
j = 1
bw = 20
CheckBox "R",Rec,0,0,bw,12
Loop
While j < 10
id = "P":j
PushButton id,$id,j*bw,0,bw,12
ControlCommand $id,DLib:"OnPClick ":j

j+=1
Repeat
```

Related Script Commands

[DialogBox Control](#)

DialogBox CheckListBox

Syntax

DialogBox CheckListBox *name, var_name, x, y, width, depth, [style]*

Synonyms

DB CX

Description

The DialogBox CheckListBox command creates a list of items that have a check box beside them.

To turn on or off a check you either click on it with the mouse or press space when the item has the focus.

See the "Reference - Script Controls" chapter for details of the styles, properties, methods and events for this control.

Parameters

The following table describes the parameters of the DialogBox CheckListBox command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New. (If you used DialogBox Create, you do not need this parameter.)
<i>var_name</i>	Specifies the name of the list box control.
<i>x</i>	Specifies the box column position in dialog box units.
<i>y</i>	Specifies the box position in dialog box units.
<i>width</i>	Defines the box width in dialog box units.
<i>depth</i>	Defines the box depth in dialog box units.
<i>style</i>	Style for the control. See "Reference - Script Controls" for details.

Example

The following example shows how to create a check list box.

```
DialogBox Create MeetingDays 18, 18, 111, 96
Caption "Select meeting days"
Style WS_CAPTION | WS_POPUP | WS_VISIBLE | WS_SYSMENU
Font 8,"MS Sans Serif"
* Default to Monday checked
InitCommand "MeetingDays.days.SetCheck 1, true"
LText "Meeting days",None, 5, 10, 100, 8
CheckListBox days, 5, 21, 100, 52, LBS_NOSEL | WS_BORDER
DefPushButton "OK", OK, 30, 76, 50, 14
OnEvent OK, "Click", "DialogBox End MeetingDays"
EndCreate
*
* Setup days of week
MeetingDays.days.List =
"Monday":cr:"Tuesday":cr:"Wednesday":cr:"Thursday":cr:"Friday"
* Set update mode to return names of days selected
MeetingDays.days.UpdateMode = 16
*
DialogBox Show MeetingDays
If ReturnValue = "OK" Then
    MessageBox "Days Chosen":cr:MeetingDays.days.CheckText, "Meeting
days selected"
EndIf
DialogBox Delete MeetingDays
```

Related Script Commands

[DialogBox AddToList](#), [DialogBox Control](#), [DialogBox Select](#), [DialogBox SetTabs](#),
[DialogList](#), [DialogIndex](#), [DialogBox ListBox](#)

Version

5.1 Original version

DialogBox ColorButton

Syntax

DialogBox ColorButton *name, text, var_name, x, y, width, depth, [style]*

Synonyms

DB CO

Description

The DialogBox ColorButton command adds a raised rectangle push button to a dialog box. Use push buttons to initiate some kind of action in the dialog box. This button is used instead of the standard push button if you wish to change the text or background color of the button.

The unit of measure for the parameters x, y, width, and depth 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.

See the "Reference - Script Controls" chapter for details of the styles, properties, methods and events for this control.

Parameters

The following table describes the parameters of the DialogBox ColorButton command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New. (If you used DialogBox Create, you do not need this parameter.)
<i>text</i>	Specifies the text of the button caption.
<i>var_name</i>	Specifies the name for this control.
<i>x</i>	Specifies the column position for displaying the button in dialog box units.
<i>y</i>	Specifies the row position for displaying the button in dialog box units.

Parameter	Description
<i>width</i>	Specifies the width to display the button in dialog box units.
<i>depth</i>	Specifies the depth or height to display the button in dialog box units.
<i>style</i>	Style for the control. See "Reference - Script Controls" for details.

Example

Set up a button with red text

```
DialogBox Create ColButTest 0,0,196,84
...
ColorButton "This button should have red text", RedButton, 31, 6, 132,
16
ColButTest.RedButton.ForeColor = RGB_Red
...
EndCreate
```

Related Script Commands

[DialogBox Control](#), [DialogBox DefPushButton](#), [DialogBox GraphicButton](#),
[DialogBox PushButton](#)

Version

5.1 Original

DialogBox ComboBox

Syntax

DialogBox ComboBox [*name*], *text*, *var_name*, *x*, *y*, *width*, *depth*, [*style*]

Synonyms

DB CB

Description

The DialogBox ComboBox command creates a list-type box with various options.

You can add an action as the focus goes or leaves the control, using the DialogBox Default and DialogBox Validate commands. If you want to limit the amount of characters allowed in the combo box, use the DialogBox LimitText command. Extra options within the list can be added or subtracted using the DialogBox AddToList or DialogBox DeleteFromList commands.

See the "Reference - Script Controls" chapter for details of the styles, properties, methods and events for this control.

Parameters

The following table describes the parameters of the DialogBox ComboBox command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New. (If you used DialogBox Create, you do not need this parameter.)
<i>text</i>	Specifies the combo box name.
<i>var_name</i>	Specifies the text for the combo box caption.
<i>x</i>	Specifies the column position for displaying the control in dialog box units.
<i>y</i>	Specifies the row position for displaying the control in dialog box units.

Parameter	Description
<i>width</i>	Specifies the width to display the control in dialog box units.
<i>depth</i>	Specifies the depth or height for displaying the control in dialog box units.
<i>style</i>	Style for the control. See "Reference - Script Controls" for details.

Example

The following example is part of the WC.WIS example script.

Notice that the label for the combobox is made with the DialogBox LText command.

```
DialogBox Create wc 44,0,229,18
Caption "Execute a Command"
Style WS_CAPTION | WS_POPUP | WS_VISIBLE | WS_SYSMENU | WS_MINIMIZEBOX
| WS_GROUP
Font 8,"Helv"
LText "&Command:",None,4,5,35,8
ComboBox Cmnd, 41,2,150,94,CBS_DROPDOWN | CBS_AUTOHSCROLL | WS_VSCROLL
| WS_TABSTOP
DefPushButton "&Run",OK,195,2,31,14
ControlCommand OK,DLib:"RunCmnd"
Control "Close",Cancel,"Button",BS_PUSHBUTTON,197,20,27,12
ControlCommand Cancel,DLib:"CloseWC"
EndCreate
```

Related Script Commands

[DialogBox AddToList](#), [DialogBox Control](#), [DialogBox Default](#), [DialogBox LimitText](#), [DialogBox Select](#), [DialogBox Validate](#), [DialogIndex](#), [DialogList](#)

DialogBox Control

Syntax

DialogBox Control [*name*], *text*, *var_name*, *class*, *style*, *x*, *y*, *width*, *depth*

Synonyms

DB CN

Description

The DialogBox Control command is an alternative way of adding controls to a dialog box.

Adding controls this way allows more flexibility in control styles. There are also some controls that you can only add this way, such as AutoCheckBox and BlackFrame.

See the Class/Style table below for a quick description of the controls available with DialogBox Control.

Parameters

The following table describes the parameters of the DialogBox Control command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New. (If you used DialogBox Create, you do not need this parameter.)
<i>text</i>	The text to display on the control. (For example, the name of a button.)
<i>var_name</i>	The name you designate for this control.
<i>class</i>	The class of the control. See the Class/Style table below for a quick description of the control classes.
<i>style</i>	The style for the control. See below.
<i>x</i>	Specifies the column position for displaying the control in dialog units.

Parameter	Description
<i>y</i>	Specifies the column position for displaying the control in dialog units.
<i>width</i>	Specifies the width for displaying the control in dialog units.
<i>depth</i>	Specifies the depth or height for displaying the control in dialog units.

Values for class

Value	Description
<i>Button</i>	The various button controls depending on the style. PushButton, Checkbox, Radio button, Group box
<i>static</i>	The text and rectangle controls.
<i>edit</i>	The edit control
<i>listbox</i>	The listbox control
<i>Combobox</i>	The combobox control
<i>Scrollbar</i>	The scrollbar control
<i>winteg.graphic</i>	The wIntegrate graphic control
<i>winteg.button</i>	The wIntegrate graphic button control
<i>winteg.draw</i>	The wIntegrate drawing control

Values for style

The following table describes additional styles that all controls can also have:

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).

Example

The following example comes from the WC.WIS demonstration script.

```
The default push button was added using DefPushButton and the second
button was added using DialogBox Control.
DefPushButton "&Run",OK,195,2,31,14
ControlCommand OK,DLib:"RunCmnd"
Control "Close",Cancel,"Button",BS_PUSHBUTTON,197,20,27,12
ControlCommand Cancel,DLib:"CloseWC"
```

DialogBox ControlCommand

Syntax

DialogBox ControlCommand [*name*], *var_name*, *script_command*

Synonyms

DB CC

Description

The DialogBox ControlCommand command specifies a script command to run when a control takes an action. The action depends on the kind of control. For example, a button control requires a click, and a scroll bar requires movement. Refer to each control for the action required.

Parameters

The following table describes the parameters of the DialogBox ControlCommand command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New. (If you used DialogBox Create, you do not need this parameter.)
<i>var_name</i>	The name of the control (button, check box, etc.). script_command The script command to run when the action takes place.
<i>script_command</i>	The script command to run when the action takes place.

Example

The following example is taken from the BARChart.WIS script example.

```
DialogBox Create $dlgname 0,0,207,150
Caption title
Style WS_CAPTION|WS_VISIBLE|WS_SYSMENU|WS_MINIMIZEBOX
Font 8,"helv"
Control "",Graph,"winteg.draw",WS_VISIBLE,0,10,200,120
DefPushButton "&Close",Cancel,84,136,37,12
```

```
ControlCommand Cancel,"DialogBox End ":dlgname:";DialogBox  
Delete ":dlgname  
MinIcon "icon\wint_bar.ico"  
EndCreate
```

Related Script Commands

[DialogBox Default](#), [DialogBox Validate](#)

DialogBox Create

Syntax

DialogBox Create *name, x, y, width, depth*

Synonyms

DB CR

Description

The DialogBox Create command creates a dialog box and puts the script into dialog box creation mode. This means that you do not need to include the DialogBox keyword or the dialog box name in the lines following. Use the EndCreate command to stop the creation of the dialog box.

Parameters

The following table describes the parameters of the DialogBox Create command:

Parameter	Description
<i>name</i>	The name that you designate for the dialog box. This name is used in many other dialog box commands to refer to this box.
<i>x</i>	Specifies the column position for displaying the dialog box in dialog box units.
<i>y</i>	Specifies the row position for displaying the dialog box in dialog box units.
<i>width</i>	Specifies the width for displaying the dialog box in dialog box units.
<i>depth</i>	Specifies the depth or height for displaying the dialog box in dialog box units.

Example

The following example is taken from the GBUTTON.WIS demonstration program.

```
* Demonstrate graphics push-buttons
* Version 2.0
DialogBox Create Toboggan 18, 18, 182, 147
CAPTION "wIntegrate Graphic Buttons"
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
LTEXT "Flat Buttons", None, 9, 8, 43, 8
LTEXT "Raised Buttons", None, 117, 8, 51, 8
LTEXT "Single Bitmap", None, 65, 40, 48, 8
LTEXT "Two Bitmaps", None, 68, 78, 43, 8
LTEXT "Three Bitmaps", None, 64, 118, 51, 8
GRAPHICBUTTON "image\normal.bmp", but1, 9, 27, 32, 32, WS_CHILD |
WS_VISIBLE | WS_TABSTOP
GRAPHICBUTTON "image\normal.bmp;pressed.bmp", but2, 9, 66, 32, 32,
WS_CHILD | WS_VISIBLE | WS_TABSTOP
GRAPHICBUTTON "image\normal.bmp;pressed.bmp;focus.bmp", but3, 9, 105,
32,
32, WS_CHILD | WS_VISIBLE | WS_TABSTOP
GRAPHICBUTTON "image\normal.bmp", but4, 135, 27, 32, 32, WS_CHILD |
WS_VISIBLE | WS_TABSTOP | GBS_RAISED
GRAPHICBUTTON "image\normal.bmp;pressed.bmp", but5, 135, 66, 32, 32,
WS_CHILD | WS_VISIBLE | WS_TABSTOP | GBS_RAISED
GRAPHICBUTTON "image\normal.bmp;pressed.bmp;focus.bmp", but6, 135,
105,
32, 32, WS_CHILD | WS_VISIBLE | WS_TABSTOP | GBS_RAISED
EndCreate
```

Related Script Commands

[DialogBox EndCreate](#), [DialogBox New](#), [DialogBox Show](#)

DialogBox CText

Syntax

DialogBox CText *name, text, var_name, x, y, width, depth, [style]*

Synonyms

DB CT

Description

The DialogBox CText command creates a center-justified text box on a dialog box. This command is generally used to create a label for some other kind of control, such as an edit box or a list box. If the text for this control will not change, use the control name "None".

See the "Reference - Script Controls" chapter for details of the styles, properties, methods and events for this control.

Note: Dialog box units are 1/4 of the average width and 1/8 of the average depth of the font.

Parameters

The following table describes the parameters of the DialogBox CText command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox Create or DialogBox New.
<i>text</i>	The text you want for a label.
<i>var_name</i>	The name you give to this text box. If the text will not change, use the name "None".
<i>x</i>	Specifies the column position of the text in dialog box units.
<i>y</i>	Specifies the row position of the text in dialog box units.
<i>width</i>	Specifies the width of the text box in dialog box units.

Parameter	Description
<i>depth</i>	Specifies the depth or height of the text box in dialog box units.
<i>style</i>	Style for the control. See "Reference - Script Controls" for details.

Example

See the example for the DialogBox LText command.

Related Script Commands

[DialogBox Control](#), [DialogBox LText](#), [DialogBox RText](#)

DialogBox DateTime

Syntax

DialogBox DateTime *dlg_name, ctrl_name, x, y, width, depth, style*

Synonyms

DB DM

Description

The DialogBox DateTime Control create a control which allows you to enter a date and/or a time. It can optionally provide a drop down calendar.

See the "Reference - Script Controls" section of this manual for details on the styles, properties, methods and events for this control.

Parameters

The following table describes the parameters of the DialogBox DateTime command:

Parameter	Description
<i>dlg_name</i>	The name of the dialog box.
<i>ctrl_name</i>	The control name.
<i>x</i>	Specifies the column position for displaying the control in dialog box units.
<i>y</i>	Specifies the row position for displaying the control in dialog box units.
<i>width</i>	Specifies the width for displaying the control in dialog box units.
<i>depth</i>	Specifies the depth or height for displaying the control in dialog box units.
<i>style</i>	See the "Reference - Script Controls" section of this manual.

Example

Adding and setting up a date time control

```
DialogBox Create DTT 0,0,265,89
...
DateTime DateTime1,81,7,124,12
OnEvent DateTime1, "SetFocus", "Print 'entering datetime1'"
OnEvent DateTime1, "KillFocus", "Print 'leaving datetime1'"
OnEvent DateTime1, "Changed", "Print 'datetime has changed'"
DTT.DateTime1.FormatType = 0
...
EndCreate
```

Version

4.0.1 Original

4.1 ForeColor/BackColor property added

DialogBox Default

Syntax

DialogBox Default [*name*], *var_name*, *command*

Synonyms

DB DF

Description

The DialogBox Default command runs a wIntegrate script command when the specified control gains input focus.

Parameters

The following table describes the parameters of the DialogBox Default command:

Parameter	Description
<i>name</i>	The name of the dialog box designated in DialogBox New or DialogBox Create
<i>var_name</i>	The name you specified for the edit, list, or combo box control.
<i>command</i>	The wIntegrate script command to run when the control gains the focus.

Related Script Commands

[DialogBox Validate](#), [DialogBox OnEvent](#)

DialogBox DefPushButton

Syntax

DialogBox DefPushButton *name, text, var_name, x, y, width, depth, [style]*

Synonyms

DB DP

Description

The DialogBox DefPushButton command creates a "default push button", a bordered button control to indicate the default action for the dialog box.

There can be only one default push button per dialog box.

Attach the action associated with the push button with the DialogBox ControlCommand command or DialogBox OnEvent command.

You can change the push button text with the DialogBox SetText command.

See the "Reference - Script Controls" chapter for details of the styles, properties, methods and events for this control.

Parameters

The following table describes the parameters of the DialogBox DefPushButton command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New. (If you used DialogBox Create, you do not need this parameter.)
<i>text</i>	The caption you specify for this push button control.
<i>var_name</i>	The name you specify for this push button control.
<i>x</i>	Specifies the column position for displaying the button in dialog box units.

Parameter	Description
<i>y</i>	Specifies the row position for displaying the button in dialog box units.
<i>width</i>	Specifies the width to display the button in dialog box units.
<i>depth</i>	Specifies the depth or height to display the button in dialog box units. Note Dialog box units are 1/4 of the average width and 1/8 of the average depth of the font.
<i>style</i>	Style for the control. See "Reference - Script Controls" for details.

Example

The following example is a part of the WC.WIS demonstration program.

```
DefPushButton "&Run",OK,195,2,31,14
ControlCommand OK,DLib:"RunCmnd"
```

Related Script Commands

[DialogBox Control](#), [DialogBox GraphicButton](#), [DialogBox PushButton](#)

DialogBox Delete

Syntax

DialogBox Delete *name*

Synonyms

DB DL

Description

The DialogBox Delete command deletes a dialog box and associated variables from memory. If the dialog box is shown, you must use first use the DialogBox End command before you can delete it.

Parameters

The following table describes the parameters of the DialogBox Delete command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox Create or DialogBox New.

Example

The following example is taken from the PRODM2.WIS demonstration program.

```
* Cancel Button is used for two purposes, Void the current entry
* and exit the program.
Sub OnCancel()
If ProdM_Mode Then
ClearEntry
Else
Library 'wintsys\lib\server'
CloseServer
DialogBox End ProdM
DialogBox Delete ProdM
```

Related Script Commands

[DialogBox Create](#), [DialogBox New](#)

DialogBox DeleteFromList

Syntax

DialogBox DeleteFromList [*name*], *control_name*, *position*

Synonyms

DB DT

Description

The DialogBox DeleteFromList command deletes the entry from a list box or combo box at the specified position. The first entry is position 1.

Parameters

The following table describes the parameters of the DialogBox DeleteFromList command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New. (If you used DialogBox Create, you do not need this parameter.)
<i>control_name</i>	The name of the list box or combo box.
<i>position</i>	The position of the entry to delete.

Example

Clear the first entry from BPList

```
DialogBox DeleteFromList MyBar, BPList, 1;
```

Related Script Commands

[DialogBox AddToList](#), [DialogBox ComboBox](#), [DialogBox ListBox](#), [DialogBox Select](#), [DialogIndex](#), [DialogList](#), [DialogListFind](#)

DialogBox EditText

Syntax

DialogBox EditText *name, var_name, x, y, width, depth, [style]*

Synonyms

DB ET

Description

The DialogBox EditText command creates a text box for user input. You can set DialogBox Default or DialogBox Validate to add action to the control, or limit the amount of text allowed by using DialogBox LimitText.

See the "Reference - Script Controls" chapter for details of the styles, properties, methods and events for this control.

Parameters

The following table describes the parameters of the DialogBox EditText command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New. (If you used DialogBox Create, you do not need this parameter.)
<i>var_name</i>	The name you give to this edit box.
<i>x</i>	Specifies the column position for displaying the edit box in dialog box units.
<i>y</i>	Specifies the row position for displaying the edit box in dialog box units.
<i>width</i>	Specifies the width to display the edit box in dialog box units.
<i>depth</i>	Specifies the depth or height to display the edit box in dialog box units.
<i>style</i>	Style for the control. See "Reference - Script Controls" for details.

Example

The following example shows the DialogBox EditText command:

```
LTEXT "Prod. Ref.", None, 7, 9, 36, 8, WS_CHILD | WS_VISIBLE |  
WS_GROUP  
EDITTEXT ProdRef, 53, 6, 62, 12, ES_LEFT | WS_CHILD | WS_VISIBLE |  
WS_BORDER | WS_TABSTOP
```

Related Script Commands

[DialogBox Control](#), [DialogBox Default](#), [DialogBox LimitText](#), [DialogBox Validate](#)

DialogBox Enable

Syntax

DialogBox Enable *name, var_name, flag*

Synonyms

DB EN

Description

The Dialogbox Enable command enables or disables a specified control. When a control is disabled, it displays as dimmed and cannot be clicked on to activate.

Parameters

The following table describes the parameters of the DialogBox Enable command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New.
<i>var_name</i>	The name of the control you want to enable.
<i>flag</i>	The flag enables or disables the control. TRUE = enables the control. FALSE = disables the control.

Example

The following example is part of the PRODM2.WIS demonstration program.

```
* Enable fields/disable id entry
Sub EnableFields(flag)
DialogBox EnableAll ProdM, 1, flag
DialogBox Enable ProdM, ProdRef, Not(flag)
DialogBox Enable ProdM, Cancel, True
If Flag Then
SetFocus ProdM.Name
Else
SetFocus ProdM.ProdRef
EndIf
EndSub
```


DialogBox EnableAll

Syntax

DialogBox EnableAll *name, option, flag*

Synonyms

DB EA

Description

The DialogBox EnableAll command allows the dialog box or specified controls in the dialog box to be enabled or disabled.

Parameters

The following table describes the parameters of the DialogBox EnableAll command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New.
<i>option</i>	Enables or disables all or some of the controls. See below
<i>flag</i>	The flag enables or disables the controls. TRUE = enables the control. FALSE = disables the control.

Values for option

Value	Description
<i>1</i>	enables or disables all controls that have a name other than "None".

Example

The following example shows the DialogBox EnableAll command:

```
*Enable all of the controls
DialogBox EnableAll ProdM, 1, True
*Then disable only one
DialogBox Enable ProdM, ProdRef, False
```

Related Script Commands

[DialogBox Enable](#)

DialogBox End

Syntax

DialogBox End *name*, *update_flag*, [*return_value*]

Synonyms

DB ED

Description

The DialogBox End command closes a dialog box.

Parameters

The following table describes the parameters of the DialogBox End command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New.
<i>update_flag</i>	This parameter updates any changed variables. True = update variables. False = retain original variables.
<i>return_value</i>	Sets the global variable ReturnValue. If return_value is not specified, the ReturnValue is set to OK if update_flag is True, and to Cancel if update_flag is False.

Example

The following example is part of the WC.WIS demonstration program.

```
* Save list for next time WC is run
Sub CloseWC()
L = dialoglist(wc,cmdnd)
DialogBox End WC
WC.Cmdnd = L
EndSub
```

Related Script Commands

[DialogBox Delete](#), [DialogBox IconBar](#), [DialogBox Show](#), [DialogBox Window](#)

DialogBox EndCreate

Syntax

DialogBox EndCreate *name*

Synonyms

DB EC

Description

The DialogBox EndCreate command stops the dialog box create mode that was started with DialogBox Create. Since you do not need to use the DialogBox designation when in create mode, write this command as EndCreate.

Parameters

The following table describes the parameters of the DialogBox EndCreate command:

Parameter	Description
<i>name</i>	The name of the dialog box designated in DialogBox Create.

Example

The following example is taken from the COPYMENU.WIS demonstration script.

```
* Popup menu for copying data
* Version 2.0
CopyLib = "Library '" : ScriptFile : "';"
* Executed line is, for example:
* Library 'c:\winteg\example\script\copymenu.wis';CTExcel
Menu Create CopyMenu
AddItem TextToNotepad, "Text to &Notepad", CopyLib : "CopyTextNotepad"
AddItem TextToWrite, "&Text to Write", CopyLib : "CopyTextWrite"
AddItem BitmapToPbrush, "Bitmap to &Paintbrush", CopyLib :
"CopyBitmapPbrush"
AddItem TableToExcel, "Table to &Excel", CopyLib : "CopyTableExcel"
AddItem TextToWord, "Text To &Word", CopyLib : "CopyTextWord"
AddItem BitToWord, "&Bitmap To Word", CopyLib : "CopyBitmapWord"
AddItem ScreenToWord, "&Screen To Word", CopyLib : "CopyScreenWord"
AddItem ScreenToPrinter, "Screen To P&rinter", CopyLib :
"CopyScreenPrinter"
```

```
AddItem PieChart, "Create Pie &Chart", "Script 'example\script\parse';  
Script  
'example\script\piechart'"  
AddItem BarChart, "Create B&ar Chart", "Script  
'example\script\parse';Script  
'example\script\barchart'"  
EndCreate
```

Related Script Commands

[DialogBox Create](#)

DialogBox EventLibrary

Syntax

DialogBox EventLibrary [*name*], [*library_file*]

Synonyms

DB EL

Description

This command attaches a library of script commands to a dialog box. This library is then automatically included for any events generated by the dialog box allowing the events to call subroutines or functions in the script without having to use the Library command in each event.

Parameters

The following table describes the parameters of the DialogBox EventLibrary command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New. (If you used DialogBox Create, you do not need this parameter.)
<i>library_file</i>	The name of the script file to use the subroutines from. If this argument is omitted or null ("") the script or library that runs this command will be used.

Related Script Commands

[DialogBox OnEvent](#), [Library](#)

Version

5.0 Original version

DialogBox ExStyle

Syntax

DialogBox ExStyle *name, extended_style*

Synonyms

DB ES

Description

The DialogBox ExStyle command sets the extended styles of a dialog box.

Parameters

The following table describes the parameters of the DialogBox ExStyle command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New. (If you used DialogBox Create, you do not need this parameter.)
<i>extended_style</i>	See the following extended styles table.

Values for extended_style

The extended style can be a combination of one or more of the following.

Value	Description
<i>WS_EX_APPWINDOW</i>	The dialog box will show its caption in the task bar and can be selected from there. If the main session is visible it will also be brought to the front when the dialog box is selected on the task bar.
<i>WS_EX_TOOLWINDOW</i>	Display the dialog box with a reduced height caption.

Example

Put the Execute commands caption in the task bar

```
* After the following lines in example\script\wc.ws:
DialogBox Create wc 44,0,229,18
Caption "Execute a Command"
Style
WS_CAPTION|WS_POPUP|WS_VISIBLE|WS_SYSMENU|WS_MINIMIZEBOX|WS_GROUP

* Add this line for the extended style:
ExStyle WS_EX_APPWINDOW
```

Related Script Commands

[DialogBox Create](#), [DialogBox Style](#)

Version

5.1.3 Original version

DialogBox Font

Syntax

DialogBox Font [*name*], *point_size*, *typeface*

Synonyms

DB FN

Description

The DialogBox Font command specifies the font for a dialog box. You can use this command only after the box is created, and before you add any of the controls.

Parameters

The following table describes the parameters of the DialogBox Font command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New. (If you used DialogBox Create, you do not need this parameter.)
<i>point_size</i>	The point size of the font.
<i>typeface</i>	The name of the font to use.

Related Script Commands

[DialogBox Create](#)

DialogBox Graphic

Syntax

DialogBox Graphic [*name*], *path*, *var_name*, *x*, *y*, *width*, *depth*, [*style*]

Synonyms

DB GR

Description

The DialogBox Graphic command creates an image control in a dialog box. You can add a script with DialogBox ControlCommand to attach action when the graphic is clicked on. The graphic can be changed when the path name of the picture changes.

Note Dialog box units are 1/4 of the average width and 1/8 of the average depth of the font.

See the "Reference - Script Controls" chapter for details of the styles, properties, methods and events for this control.

Parameters

The following table describes the parameters of the DialogBox Graphic command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New. (If you used DialogBox Create, you do not need this parameter.)
<i>path</i>	The path name of the image.
<i>var_name</i>	The name you give to this graphic control.
<i>x</i>	Specifies the column position for displaying this control in dialog box units.
<i>y</i>	Specifies the row position for displaying this control in dialog box units.
<i>width</i>	Specifies the width for displaying this control in dialog box units.

Parameter	Description
<i>depth</i>	Specifies the depth or height for displaying this control in dialog box units.
<i>style</i>	Style for the control. See "Reference - Script Controls" for details.

Example

The following example is part of the PRODM2.WIS demonstration program.

```
Graphic "", Pic, 97, 40, 96, 68, WS_CHILD | WS_VISIBLE  
CONTROL "Show Image", ShowImage, "BUTTON", BS_AUTOCHECKBOX | WS_CHILD  
| WS_VISIBLE | WS_TABSTOP, 97, 112, 96, 12
```

Display a clipped image

```
Graphic AppDir:"image.bmp",SizedToFit,34,15,108,53,  
GraphicTest.SizedToFit.Display = 1
```

Resizing a control to exactly fit an image

```
* In the InitCommand subroutine  
...  
With GraphicTest.SizedToFit  
.Width = .ImageWidth  
.Height = .ImageHeight  
EndWith  
...
```

Related Script Commands

[DialogBox Control](#), [DialogBox ControlCommand](#), [DialogBox GraphicButton](#)

DialogBox GraphicButton

Syntax

DialogBox GraphicButton *name, image, control_name, x, y, width, depth, style*

Synonyms

DB GB

Description

The DialogBox GraphicButton command creates a push button control with a graphic image. You can attach a script command to the image with the DialogBox ControlCommand command.

See the "Reference - Script Controls" chapter for details of the styles, properties, methods and events for this control.

Parameters

The following table describes the parameters of the DialogBox GraphicButton command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New. (If you used DialogBox Create, you do not need this parameter.)
<i>image</i>	The file name(s) for the image(s) of this control. The images in order are the normal image, the pressed image and the focus image. The image names are separated by semi-colons.
<i>control_name</i>	The name for this control
<i>x</i>	Specifies the column position for displaying the control in dialog box units.
<i>y</i>	Specifies the row position for displaying the control in dialog box units.
<i>width</i>	Specifies the width for displaying the control in dialog box units.

Parameter	Description
<i>depth</i>	Specifies the depth or height for displaying the control in dialog box units.
<i>style</i>	Style for the control. See "Reference - Script Controls" for details.

Related Script Commands

[DialogBox Control](#), [DialogBox ControlCommand](#), [DialogBox Graphic](#)

DialogBox Grid

Syntax

DialogBox Grid *dlg_name, ctrl_name, x, y, width, depth, [style]*

Synonyms

DB GD

Description

The DialogBox 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.

See the "Reference - Script Controls" section of this manual for details on the properties, events and methods available for this control.

Parameters

The following table describes the parameters of the DialogBox Grid command:

Parameter	Description
<i>dlg_name</i>	The name of the dialog.
<i>ctrl_name</i>	The name of the control
<i>x</i>	Specifies the column position for displaying the control in dialog box units.
<i>y</i>	Specifies the row position for displaying the control in dialog box units.
<i>width</i>	Specifies the width to display the control in dialog box units.
<i>depth</i>	Specifies the depth or height to display the control in dialog box units.
<i>style</i>	Style for the control. See "Reference - Script Controls" for details.

DialogBox GroupBox

Syntax

DialogBox GroupBox *name, text, var_name, x, y, width, depth, style*

Synonyms

DB GP

Description

The DialogBox GroupBox command 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. DialogBox GroupBox must be used before DialogBox RadioButton to create a box with radio buttons.

See the "Reference - Script Controls" chapter for details of the styles, properties, methods and events for this control.

Parameters

The following table describes the parameters of the DialogBox GroupBox command:

Parameter	Description
<i>name</i>	The name you give to this group box.
<i>text</i>	The text for the groupbox caption.
<i>var_name</i>	The dialog box variable name.
<i>x</i>	Specifies the column position for displaying the box in dialog box units. Specifies the column position for displaying the header in dialog box units.
<i>y</i>	Specifies the row position for displaying the box in dialog box units. Specifies the row position for displaying the header in dialog box units.
<i>width</i>	Specifies the width for displaying the box in dialog box units. Specifies the width for displaying the header in dialog box units.

Parameter	Description
<i>depth</i>	Specifies the depth or height for displaying the box in dialog box units.
<i>style</i>	Style for the control. See "Reference - Script Controls" for details.

Example

The following example shows just the portion of a dialog box script that creates a group box and the radio buttons within it

```
* Note that the last radio button is dimmed, which indicates that it
is disabled.
* Add a group box with radio buttons
GroupBox "Output to:",Output,115,25,60,50, BS_GROUPBOX | WS_TABSTOP
RadioButton "Screen", Screen, 120, 35, 55, 10
RadioButton "Local Printer", Local, 120, 45, 55, 10
RadioButton "Host Printer", Host, 120, 55, 55, 10, WS_DISABLED
```

DialogBox Header

Syntax

DialogBox Header *dlg_name, ctrl_name, x, y, width, depth, style*

Synonyms

DB HD

Description

The DialogBox Header command creates a header in a dialog box.

See the "Reference - Script Controls" chapter for details of the styles, properties, methods and events for this control.

Parameters

The following table describes the parameters of the DialogBox Header command:

Parameter	Description
<i>dlg_name</i>	The name of the dialog box.
<i>ctrl_name</i>	The control name.
<i>x</i>	Specifies the column position for displaying the header in dialog box units.
<i>y</i>	Specifies the row position for displaying the header in dialog box units.
<i>width</i>	Specifies the width for displaying the header in dialog box units.
<i>depth</i>	Specifies the depth or height for displaying the header in dialog box units.
<i>style</i>	Style for the control. See "Reference - Script Controls" for details.

DialogBox Icon

Syntax

DialogBox Icon *name, winteg, var_name, x, y, width, depth*

Synonyms

DB IN

Description

The DialogBox Icon command creates a control which holds an icon from the wIntegrate executable (winteg.exe). The only icon available is the "winteg" icon.

Tip: You can be more flexible with icons by using the DialogBox Graphic command.

Parameters

The following table describes the parameters of the DialogBox Icon command:

Parameter	Description
<i>name</i>	The name of the dialog box.
<i>winteg</i>	The icon name - must be "winteg".
<i>var_name</i>	The control name.
<i>x</i>	Specifies the column position for displaying the icon in dialog box units.
<i>y</i>	Specifies the row position for displaying the icon in dialog box units.
<i>width</i>	Specifies the width of the icon dialog box units.
<i>depth</i>	Specifies the depth or height of the icon in dialog box

Example

Show the winteg icon

```
DialogBox Icon "winteg", None, 10, 10, 32,32
```

Related Script Commands

[DialogBox Graphic](#)

DialogBox IconBar

Syntax

DialogBox IconBar *name*

Synonyms

DB IB

Description

The DialogBox IconBar command registers a dialog box as an icon bar. If an icon bar is already shown, it will change to the one you name here.

This command is kept for backward compatability with older versions of wIntegrate. Newer versions should use the CommandBar script commands for defining iconbars.

If you choose Preferences from the Setup Menu, you can select multiple toolbars rather than turning on/off one iconbar.

Parameters

The following table describes the parameters of the DialogBox IconBar command:

Parameter	Description
<i>name</i>	The name of the dialog box to use as an icon bar.

Example

The following example sets the general icon bar.

```
DialogBox IconBar general_bar
```

Related Script Commands

[DialogBox Show](#), [DialogBox Window](#), [CommandBar Add](#)

DialogBox InitCommand

Syntax

DialogBox InitCommand [*name*], *command*

Synonyms

DB IC

Description

The DialogBox InitCommand attaches a command to be run when a dialog box displays.

Parameters

The following table describes the parameters of the DialogBox InitCommand command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New. (If you used DialogBox Create, you do not need this parameter.)
<i>command</i>	The wIntegrate script command to run.

Example

The following example is part of the STRING.WIS demonstration program.

```
InitCommand "Script 'example\script\dostring' "  
ControlCommand OK, "Script 'example\script\dostring' "  
ControlCommand CANCEL, "DialogBox End StringTest,FALSE"  
EndCreate
```

Related Script Commands

[DialogBox IconBar](#), [DialogBox Show](#), [DialogBox Window](#)

DialogBox InputOK

Syntax

DialogBox InputOK *name, ok_flag*

Synonyms

DB IO

Description

The DialogBox InputOK command is used with the DialogBox Option command using "V1" and "V2" options inside the processing of a DialogBox Validate command whenever these options are set or the dialog box does not accept any more input. This command must be called in every validation script when these options are set or the dialog box does not accept any more input.

Parameters

The following table describes the parameters of the DialogBox InputOK command:

Parameter	Description
<i>name</i>	The dialog box name, designated in Dialog New or Dialog Create. The name of the dialog box, designated in DialogBox New. (If you used DialogBox Create, you do not need this parameter.)
<i>ok_flag</i>	Can be one of the following: 0 -False. Processing returns to the field being validated and the next event is discarded. 1 - True. Processing moves to the next field selected (via the Tab button or a mouse click).

Example

The following example is taken from the PRODM2.WIS demonstration program.

```
* Validate Price
Sub OnValidatePrice()
If Not(Match(ProdM.Price,"1N0N")) Or ProdM.Price < 1 Then
MessageBox 'Price must be an integer greater than zero',
"Product Maintenance", MB_ICONHAND|MB_OK, ProdM
```

```
DialogBox InputOK, ProdM, False  
Else  
DialogBox InputOK ProdM, True  
If ProdM_Mode < 8 Then CheckWrite 4  
EndIf
```

DialogBox LimitText

Syntax

DialogBox LimitText *name, var_name, max*

Synonyms

DB LM

Description

The DialogBox LimitText command specifies the maximum number of characters allowed in an edit control or combo box.

Parameters

The following table describes the parameters of the DialogBox LimitText command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox
<i>var_name</i>	The name of the control where the text limit is set.
<i>max</i>	The maximum number of characters to allow in the control. The number can be 0 - 32767.

Example

The following example sets up a dialog box with an edit control. LimitText sets the maximum text at 25 characters.

```
DialogBox Create Test 10, 30, 264, 117
.
.
.
LTEXT "Original Text:", None, 5, 10, 46, 8, WS_CHILD | WS_VISIBLE |
WS_GROUP
EDITTEXT text, 57, 7, 152, 12
LIMITTEXT text, 25
```

Related Script Commands

[DialogBox ComboBox](#), [DialogBox EditText](#)

DialogBox ListBox

Syntax

DialogBox ListBox *name, var_name, x, y, width, depth, [style]*

Synonyms

DB LB

Description

The DialogBox ListBox command creates a list box with various options.

You can highlight single or multiple options with the mouse or keyboard. You can add a double-click action using DialogBox ControlCommand. To highlight or select a particular option in the box, use the DialogBox Select command. Extra options within the list can be added using the DialogBox AddToList command.

See the "Reference - Script Controls" chapter for details of the styles, properties, methods and events for this control.

Parameters

The following table describes the parameters of the DialogBox ListBox command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New. (If you used DialogBox Create, you do not need this parameter.)
<i>var_name</i>	Specifies the name of the list box control.
<i>x</i>	Specifies the box column position in dialog box units.
<i>y</i>	Specifies the box position in dialog box units.
<i>width</i>	Defines the box width in dialog box units.
<i>depth</i>	Defines the box depth in dialog box units.
<i>style</i>	Style for the control. See "Reference - Script Controls" for details.

Example

The following example shows how to create a list box.

```
DialogBox Create test 18, 18, 240, 150
CAPTION "List Files"
FONT 8, "helv"
STYLE WS_POPUP | WS_CAPTION | WS_SYSMENU | WS_MINIMISE
LTEXT "Files", None, 5,10,32,8, WS_CHILD | WS_VISIBLE | WS_GROUP
Listbox filelist, 34,3,65,75, WS_BORDER, | WS_VSCROLL | WS_TABSTOP
Files test.filelist
```

Related Script Commands

[DialogBox AddToList](#), [DialogBox Control](#), [DialogBox Select](#), [DialogBox SetTabs](#),
[DialogList](#), [DialogIndex](#), [DialogBox CheckListBox](#)

DialogBox ListView

Syntax

DialogBox ListView *dlg_name, ctrl_name, x, y, width, depth, [style]*

Synonyms

DB LV

Description

The Dialog ListView control displays a list of items with optional images and additional information.

See the "Reference - Script Controls" chapter for details of the styles, properties, methods and events for this control.

Parameters

The following table describes the parameters of the DialogBox ListView command:

Parameter	Description
<i>dlg_name</i>	The name of the dialog box.
<i>ctrl_name</i>	The control name.
<i>x</i>	Specifies the column position for displaying the control in dialog box units.
<i>y</i>	Specifies the row position for displaying the control in dialog box units.
<i>width</i>	Specifies the width for displaying the control in dialog box units.
<i>depth</i>	Specifies the depth or height for displaying the control in dialog box units.
<i>style</i>	Style for the control. See "Reference - Script Controls" for details.

DialogBox LoseFocus

Syntax

DialogBox LoseFocus [*name*], *flag*

Synonyms

DB LF

Description

The DialogBox LoseFocus function makes a dialog box lose focus when a button or list box is activated. LoseFocus works only when a dialog box is displayed as a window or an icon bar. The default for LoseFocus is False with DialogBox Window and True with DialogBox IconBar.

Parameters

The following table describes the parameters of the DialogBox LoseFocus command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New. (If you used DialogBox Create, you do not need this parameter.)
<i>flag</i>	Sets the flag for this command. True - The window or iconbar loses focus. False - The window or icon bar retains the focus.

Example

The following example is part of the PRESTORE.WIS example program.

```
DialogBox Create Prestore,0,0,220,12
Style WS_CAPTION|WS_SYSMENU|WS_MINIMIZEBOX
Font 8,"helv"
Caption "Prestore"
LoseFocus True
This results in the title bar of the window dimming when a button is
clicked.
```

Related Script Commands

[DialogBox Create](#), [DialogBox IconBar](#), [DialogBox Window](#)

DialogBox LText

Syntax

DialogBox LText *name, text, var_name, x, y, width, depth, [style]*

Synonyms

DB LT

Description

The DialogBox LText command Creates a left-justified text box on a dialog box. This command is generally used to create a label for some other kind of control, like an edit box or a list box. If the text for this control will not change, use the control name "None".

See the "Reference - Script Controls" chapter for details of the styles, properties, methods and events for this control.

Note: Dialog box units are 1/4 of the average width and 1/8 of the average depth of the font.

Parameters

The following table describes the parameters of the DialogBox LText command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox Create or DialogBox New.
<i>text</i>	The text you want for this text box.
<i>var_name</i>	The name you give to the this text box. If the text will not change, use the name "None".
<i>x</i>	The left position of the text in dialog box units.
<i>y</i>	The top position of the text in dialog box units.
<i>width</i>	The width of the text box in dialog box units.
<i>depth</i>	The depth of the text box in dialog box units.

Parameter	Description
<i>style</i>	Style for the control. See "Reference - Script Controls" for details.

Example

The following example is a part of the PRODM2.WIS demonstration script.

```
DialogBox LText Styles
LTEXT "Prod. Ref.", None, 7, 9, 36, 8, WS_CHILD | WS_VISIBLE |
WS_GROUP
EDITTEXT ProdRef, 53, 6, 62, 12, ES_LEFT | WS_CHILD | WS_VISIBLE |
WS_BORDER | WS_TABSTOP
LTEXT "Name:", None, 7, 25, 38, 8, WS_CHILD | WS_VISIBLE | WS_GROUP
```

Related Script Commands

[DialogBox Control](#), [DialogBox CText](#), [DialogBox RText](#)

DialogBox MaxSize

Syntax

DialogBox MaxSize [*name*], *width*, *depth*

Synonyms

DB MS

Description

The DialogBox MaxSize command specifies the maximum size to which a dialog box can be resized.

If you created the dialog box with the WS_HSCROLL and WS_VSCROLL style, this command automatically adds vertical and horizontal scroll bars that will scroll to the maximum size you set in DialogBox MaxSize.

To set the scroll step increments, use the DialogBox ScrollStep command after DialogBox MaxSize.

If either the width or depth is specified as -1, the width or depth is assumed to be the same the size specified in DialogBox New or DialogBox Create.

Parameters

The following table describes the parameters of the DialogBox MaxSize command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New. (If you used DialogBox Create, you do not need this parameter.)
<i>width</i>	The maximum width of the dialog box in dialog box units.
<i>depth</i>	The maximum depth of the dialog box in dialog box units.

Example

The following example shows how to set the MaxSize and ScrollStep commands:

```
DialogBox Create Cust 16,2,236,120
Caption "Cust"
Style WS_CAPTION | WS_VISIBLE | WS_SYSMENU | WS_THICKFRAME |
WS_MINIMIZEBOX |
WS_MAXIMIZEBOX | WS_TABSTOP | WS_GROUP | WS_HSCROLL | WS_VSCROLL
.
.
.
Pushbutton "Cancel",Cancel,184,3,40,12
ControlCommand Cancel,"DIALOGBOX END Cust,FALSE"
Pushbutton "OK",OK,184,20,40,12
ControlCommand OK,"DIALOGBOX END Cust,True"
MaxSize 250, 150
ScrollStep 10, 10
EndCreate
DialogBox Show Cust
DialogBox Delete Cust
```

Related Script Commands

[DialogBox ScrollStep](#)

DialogBox MinIcon

Syntax

DialogBox MinIcon [*name*], *filename*

Synonyms

DB MI

Description

The DialogBox MinIcon command specifies the icon to use for a dialog box when it is minimized. You can use this only when the dialog box style WS_MINIMIZEBOX is set with the DialogBox Style command.

Parameters

The following table describes the parameters of the DialogBox MinIcon command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New
<i>filename</i>	The file name of the icon.

Example

The following example is taken from the BARCHART.WIS example program.

```
DialogBox Create $dlgname 0,0,207,150
Caption title
Style WS_CAPTION|WS_VISIBLE|WS_SYSMENU|WS_MINIMIZEBOX
Font 8,"helv"
Control "",Graph,"winteg.draw",WS_VISIBLE,0,10,200,120
DefPushButton "&Close",Cancel,84,136,37,12
ControlCommand Cancel,"DialogBox End ":dlgname:";DialogBox
Delete ":dlgname
MinIcon "icon\wint_bar.ico"
EndCreate
*
```

When you create a wIntegrate bar chart and you minimize the dialog box, the minimized icon appears:

Related Script Commands

[DialogBox Style](#)

DialogBox MinSize

Syntax

DialogBox MinSize *dlg_name, width, depth*

Synonyms

DB MZ

Description

The DialogBox MinSize command specifies the minimum size a user can resize a sizeable dialog box by dragging one of its edges.

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 DialogBox MinSize command:

Parameter	Description
<i>dlg_name</i>	The name of the dialog box.
<i>width</i>	The minimum width. -1 means use default for the dialog box
<i>depth</i>	The minimum depth. -1 means use the default for the dialog box

Example

Ensure the dialog box can not be reduce to below 50x50 dialog units

```
* (this is between a DialogBox Create/EndCreate)
MinSize 50,50
```

Change the minimum size to 100x60

```
DialogBox MinSize MyDialog, 100,60
```

Version

4.2.1 Original

DialogBox Move

Syntax

DialogBox Move [*name*], *x*, *y*

Synonyms

DB MV

Description

The DialogBox Move command changes the position of a dialog box.

Parameters

The following table describes the parameters of the DialogBox Move command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New
<i>x</i>	The column destination of the dialog box.
<i>y</i>	The row destination of the dialog box.

Example

The following example moves the dialog box to the top left corner.

```
DialogBox Move MyDialog, 0,0
```

Related Script Commands

[DialogBox Size](#)

DialogBox MoveControl

Syntax

DialogBox MoveControl *dlg_name, ctrl_name, left, top, width, depth, dialog_units*

Synonyms

DB MC

Description

The DialogBox MoveControl command allows a control to be moved or resized. The dialog box must be displayed at the time this command is issued.

Parameters

The following table describes the parameters of the DialogBox MoveControl command:

Parameter	Description
<i>dlg_name</i>	The name of the dialog box.
<i>ctrl_name</i>	The name of the control
<i>left</i>	The new left hand side of the control. Leave blank to leave unchanged
<i>top</i>	The new top of the control. Leave blank to leave unchanged
<i>width</i>	The new width of the control. Leave blank to leave unchanged
<i>depth</i>	The new depth of the control. Leave blank to leave unchanged
<i>dialog_units</i>	0 - size is in pixels (default). 1 - size is in dialog units

Related Script Commands

[DialogBox ShowControl](#)

Version

4.0.2 Original

DialogBox New

Syntax

DialogBox New *name, x, y, width, depth*

Synonyms

DB NW

Description

The DialogBox New command creates a new dialog box. Unlike DialogBox Create, this command does not put the script into dialog box creation mode. This means that you must use the DialogBox designation before each dialog box command, and you must refer to the dialog box name. For this reason, it is easier to use DialogBox Create rather than DialogBox New.

Parameters

The following table describes the parameters of the DialogBox New command:

Parameter	Description
<i>name</i>	The name that you designate for the dialog box. You must refer to this name in other dialog box commands to refer to this box.
<i>x</i>	Specifies the column position for displaying the dialog box in dialog box units.
<i>y</i>	Specifies the row position for displaying the dialog box in dialog box units.
<i>width</i>	Specifies the width of the dialog box in dialog box units.
<i>depth</i>	Specifies the depth or height of the dialog box in dialog box units.

Example

The following example shows the first line of a script creating a new dialog box.

```
DialogBox New Transfer, 20, 20, 100, 50
```

Related Script Commands

[DialogBox Create](#)

DialogBox NextControl

Syntax

DialogBox NextControl *dlg_name*, [*reverse_flag*]

Synonyms

DB NC

Description

The DialogBox NextControl command moves the focus in the named dialog box to the next control in the tab order. If the reverse_flag is set to true, the focus is set to the previous control.

Parameters

The following table describes the parameters of the DialogBox NextControl command:

Parameter	Description
<i>dlg_name</i>	Name of the dialog.
<i>reverse_flag</i>	Reverse direction flag.

Example

The following example sets up a next and previous button to move the focus

```
* As the "Next" button has the name OK hitting Return will run it
* otherwise Alt N will go to the next field and Alt P to the previous
Pushbutton "&Next",OK,154,26,40,12
ControlCommand OK,'DB NextControl NextTest'
Pushbutton "&Previous",Previous,155,46,40,12
ControlCommand Previous,'Db NextControl NextTest, True'
```

DialogBox OnDialogEvent

Syntax

DialogBox OnDialogEvent *dlg_name, event, command, arguments*

Synonyms

DB OD

Description

The DialogBox OnDialogEvent command sets a script command to be run when the specified event occurs on the dialog box.

Parameters

The following table describes the parameters of the DialogBox OnDialogEvent command:

Parameter	Description
<i>dlg_name</i>	The name of the dialog box.
<i>event</i>	Name of the event in quotes. See below
<i>command</i>	Script command to be run when the event is received
<i>arguments</i>	List of variable names to be set with event specific parameters

Values for event

The following values can be used for the event.

Value	Description
<i>Init</i>	This event occurs whenever the dialog box is displayed. This is identical to using the DialogBox InitCommand command.

Value	Description
<i>Size</i>	The dialog box has had its size changed. e.g. by the user dragging a corner. See below for the arguments that can be returned.
<i>Activate</i>	The dialog box has been activated or deactivated

Size Event Arguments

Argument	Description
state	The state of the dialog when the size event occurred. 0 - normal, 1 - minimized, 2 - maximized
cx	The width of the client area of the dialog in pixels
cy	The depth of the client area of the dialog in pixels

Activate Event Arguments

Argument	Description
state	The activation state 0 - the window is being deactivated, 1 the window is being activated, 2 the window is being activated by a mouse click.
minimized	The minimized state of the window is being activated/deactivated
window	The Windows handle of the window that was deactivated when this window was activated or that will be activated when this window is deactivated.

Example

Display the size of the dialog box on the status bar when it is resized

```
DialogBox Create MyDialog
...
OnDialogEvent "Size", "Print 'state = ':state:', width = ':cx:', depth
=:cy", state,cx,cy
...
EndCreate
```

Close the dialog box when the user clicks elsewhere

```
OnDialogEvent "Activate","If state = 0 Then DialogBox
EndMyDialog,False",state
```

Related Script Commands

[DialogBox InitCommand](#)

Version

4.0.2 Original

4.2.1 Activate event added

DialogBox OnEvent

Syntax

DialogBox OnEvent *dlg_name, ctrl_name, event_id, command, [arg1, arg2, ... argN]*

Synonyms

DB OE

Description

The DialogBox OnEvent command processes events from an activeX control.

Parameters

The following table describes the parameters of the DialogBox OnEvent command:

Parameter	Description
<i>dlg_name</i>	The name of the dialog.
<i>ctrl_name</i>	The name of the control.
<i>event_id</i>	The name or ID for the event.
<i>command</i>	The script command to run.
<i>arg1, arg2, ... argN</i>	The names of variables to set with the parameters from the event.

Related Script Commands

[DialogBox ControlCommand](#), [DialogBox Default](#), [DialogBox Validate](#)

DialogBox Option

Syntax

DialogBox Option [*name*], *option*

Synonyms

DB OP

Description

The DialogBox Option command sets options that affect dialog box processing or appearance. The options cover validation processing, mapping of characters and the starting appearance of the dialog box.

Parameters

The following table describes the parameters of the DialogBox Option command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New. (If you used DialogBox Create, you do not need this parameter.)
<i>option</i>	Option to set. See Below

Values for option

Combine one or more of the following options in a string

Value	Description
<i>V0</i>	Run Validate script and do not block processing of subsequent dialog box messages.
<i>V1</i>	Run Validate script and block processing of subsequent dialog box processing.
<i>V2</i>	Same as V0 but input to dialog is disabled.
<i>L</i>	Light font. This has no effect and is retained for backwards compatibility only

Value	Description
<i>M</i>	Dialog boxes defined on the host will use the emulation character mapping. This is required only if the dialog box is defined on the host and/or the dialog box values are being sent to or from the host. The emulation can also be mapped to the host using the MapChars function.
<i>B</i>	Change all the controls in the dialog box to a bold version of the dialog box font and resize them. This option is to allow a dialog box to have the same size and appearance as a dialog box which is being displayed in wIntegrate 3.x.

Example

The following example is taken from the PRODM2.WIS demonstration program.

```
* New option command, placed here as so main dialog can
* still be edited by the dialog designer
DialogBox Option ProdM, "V2"
ProdM.ShowImage = True
Global ProdM_Mode = 0
EndSub
```

The following line can be added to a dialog box after the Font definition in version 3.x or 4.1.1+ so the dialog boxes have identical appearances in both programs.

```
If Version >= "4.1.1" Then Option "B"
```

Related Script Commands

[DialogBox InputOK](#), [DialogBox Validate](#)

Version

4.1.1

DialogBox Panel

Syntax

DialogBox Panel *dlg_name, panel, position*

Synonyms

DB PN

Description

The DialogBox Panel command 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 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 DialogBox MinSize command, 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.

Note: When using a list box control in a resizable panel you must specify the non-integral height style.

Parameters

The following table describes the parameters of the DialogBox Panel command:

Parameter	Description
<i>dlg_name</i>	The name of the dialog box.
<i>panel</i>	The panel to add or move. Must be one of Left, Right, Top or Bottom.
<i>position</i>	The position (measured from the left/top) for the panel. If set to 0 the panel will be removed.

Example

Setup all four panels in a dialog box

```
* Set the panel 50 dialog units in from the width and 20 units
* in from the depth of a 300 by 200 sized dialog box
Panel Left 50
Panel Right 250
Panel Top 20
Panel Bottom 180
```

Version

4.2.1 Original

DialogBox PopupMenu

Syntax

DialogBox PopupMenu { *dialog* / *control* }, *x*, *y*, *flags*, *position_flag*

Synonyms

DB PM

Description

The DialogBox PopupMenu command pops up the menu on a dialog box. The menu must have been previously created using the Menu Create command and the dialog box must be displayed.

This command differs from the Menu Popup command in that it allows the position for the menu to be specified relative to the client area of the dialog box.

Parameters

The following table describes the parameters of the DialogBox PopupMenu command:

Parameter	Description
<i>{ dialog / control }</i>	The name of the dialog box. If the dialog box name contains a period (.) it is taken as a control name
<i>x</i>	The column position to show the menu. The meaning of this parameter is modified by <i>position_flag</i> below.
<i>y</i>	The row position to show the menu. The meaning of this parameter is modified by <i>position_flag</i> below.
<i>flags</i>	Flags, see below.
<i>position_flag</i>	Specified the meaning of the <i>x</i> , <i>y</i> co-ordinates, see table below.

Values for flags

Value	Description
2	Allows the right mouse button to select menu items.
4	Centers the menu around the x position.
8	Right-aligns the menu to the x position.

Values for position_flag

Value	Description
0	dialog units from the top left of the client area of the dialog box
1	pixels from the top left of the client area of the dialog box or control if a control name is specified.
2	screen position of the dialog box

Example

Popup a right click menu on the TaskList grid

```
* In the right-click event for the grid control SchedDlg.TaskList
* the x,y pixel position of the click are given relative to the grid,
* so we popup the menu in the correct place with:
DialogBox PopupMenu SchedDlg.TaskList, SchedTasksMenu, x, y,,1
```

Related Script Commands

[Menu Create](#)

Version

4.2.1 Original

DialogBox PopupWindow

Syntax

DialogBox PopupWindow *name, parent name, control name, alignment*

Synonyms

DB PW

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 DialogBox PopupWindow command:

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

Example

Popup the CalcDlg dialog box

```
DialogBox PopupWindow CalcDlg, MainDlg, BrowseButton
```

Version

4.2.1 Original

DialogBox ProgressBar

Syntax

DialogBox ProgressBar *dlg_name, ctrl_name, x, y, width, depth, [style]*

Synonyms

DB PR

Description

The DialogBox ProgressBar control creates a progress bar.

See the "Reference - Script Controls" chapter for details of the styles, properties, methods and events for this control.

Parameters

The following table describes the parameters of the DialogBox ProgressBar command:

Parameter	Description
<i>dlg_name</i>	The name of the dialog box.
<i>ctrl_name</i>	The control name.
<i>x</i>	Specifies the column position for displaying the control in dialog box units.
<i>y</i>	Specifies the row position for displaying the control in dialog box units.
<i>width</i>	Specifies the width for displaying the header in control box units.
<i>depth</i>	Specifies the depth or height for displaying the control in dialog box units.
<i>style</i>	Style for the control. See "Reference - Script Controls" for details.

DialogBox PushButton

Syntax

DialogBox PushButton *name, text, var_name, x, y, width, depth, [style]*

Synonyms

DB PB

Description

The DialogBox PushButton command 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*, *width*, and *depth* 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.

See the "Reference - Script Controls" chapter for details of the styles, properties, methods and events for this control.

Parameters

The following table describes the parameters of the DialogBox PushButton command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New. (If you used DialogBox Create, you do not need this parameter.)
<i>text</i>	Specifies the text of the button caption.
<i>var_name</i>	Specifies the name for this push button control.
<i>x</i>	Specifies the column position for displaying the button in dialog box units.
<i>y</i>	Specifies the row position for displaying the button in dialog box units.
<i>width</i>	Specifies the width to display the button in dialog box units.

Parameter	Description
<i>depth</i>	Specifies the depth or height to display the button in dialog box units.
<i>style</i>	Style for the control. See "Reference - Script Controls" for details.

Example

The following example is taken from the STRING.WIS demonstration program.

```
DialogBox Create StringTest 10, 30, 264, 117
CAPTION "String function test"
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
LTEXT "Original Text:", None, 5, 10, 46, 8, WS_CHILD | WS_VISIBLE |
WS_GROUP
EDITTEXT text, 57, 7, 152, 12
PUSHBUTTON "&Execute", OK, 222, 5, 34, 14, WS_CHILD | WS_VISIBLE |
WS_TABSTOP
.
.
.
ControlCommand OK, "Script 'example\script\dostring'"
```

Related Script Commands

[DialogBox Control](#), [DialogBox DefPushButton](#), [DialogBox GraphicButton](#)

DialogBox RadioButton

Syntax

DialogBox RadioButton [*name*], *text*, *var_name*, *x*, *y*, *width*, *depth*, [*style*]

Synonyms

DB RB

Description

The DialogBox RadioButton command adds a selectable control with text to a dialog box.

You can group several radio buttons together in a box by using DialogBox GroupBox. In this case DialogBox GroupBox must be used before DialogBox RadioButton to create a box with radio buttons.

Note: Dialog box units are 1/4 of the average width and 1/8 of the average depth of the font.

See the "Reference - Script Controls" chapter for details of the styles, properties, methods and events for this control.

Parameters

The following table describes the parameters of the DialogBox RadioButton command:

Parameter	Description
<i>name</i>	The name of the dialog box designated in DialogBox New. (If you used DialogBox Create, you do not need this parameter.)
<i>text</i>	The text for the radio button caption.
<i>var_name</i>	The name you give to this radio button.
<i>x</i>	Specifies the column position for the radio button in dialog box units.

Parameter	Description
<i>y</i>	Specifies the row position for the radio button in dialog box units.
<i>width</i>	Specifies the width for the radio button in dialog box units.
<i>depth</i>	Specifies the depth or height position for the radio button in dialog box units.
<i>style</i>	Style for the control. See "Reference - Script Controls" for details.

Example

The following example shows just the portion of a dialog box script that creates a group box and the radio buttons within it.

Note that the last radio button is dimmed, which indicates that it is disabled.

DialogBox RadioButton Styles

* Add a group box with radio buttons

GroupBox "Output to:",Output,115,25,60,50, BS_GROUPBOX | WS_TABSTOP

RadioButton "Screen", Screen, 120, 35, 55, 10

RadioButton "Local Printer", Local, 120, 45, 55, 10

RadioButton "Host Printer", Host, 120, 55, 55, 10, WS_DISABLED

Related Script Commands

[DialogBox Control](#)

DialogBox Reply

Syntax

DialogBox Reply *dlg_name, reply*

Synonyms

DB RP

Description

The DialogBox Reply command is used in event handling routines to specify the response for events which require a response.

Note This command must be used before any command that causes multitasking or it will have no effect.

Parameters

The following table describes the parameters of the DialogBox Reply command:

Parameter	Description
<i>dlg_name</i>	Name of dialog box for reply.
<i>reply</i>	Reply to message.

DialogBox ResetControls

Syntax

DialogBox ResetControls *dialog_name*, [*option*]

Synonyms

DB RC

Description

The DialogBox ResetControls command allows all the controls in a dialog box to be reset. After running this command, you must remake any selection in a list box or combo box. This command does not affect Draw controls.

If a text control has a name but does not have its value set to the text before the dialog box is shown, option 0 clears it to empty string ("").

Option 1 resets all controls to "" or off, so only one button in each radio group can be selected.

Parameters

The following table describes the parameters of the DialogBox ResetControls command:

Parameter	Description
<i>dialog_name</i>	Name of dialog box to reset the controls for.
<i>option</i>	Reset options. 0 - The values are reset to the values of the dialog box when it was first shown. This is the default. 1 - All the values are set to "".

Example

The following example is taken from the query builder Clear button processing (subroutine OnClear).

```
DialogBox ResetControls Query,1
Query_Verb = L("SORT")
Query.Verb = L("LIST"):cr:L("SORT"):cr:L("SELECT"):cr:L("SSELECT")
DialogBox Select Query_Verb, Query_Verb
```

```
Query.OutputTo = "Screen"
```

DialogBox RText

Syntax

DialogBox RText *name, text, var_name, x, y, width, depth, style*

Synonyms

DB RT

Description

The DialogBox RText command creates a right-justified text box on a dialog box. This command is generally used to create a label for some other kind of control, like an edit box or a list box. If the text for this control will not change, use the control name "None".

See the "Reference - Script Controls" chapter for details of the styles, properties, methods and events for this control.

Parameters

The following table describes the parameters of the DialogBox RText command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox Create or DialogBox New.
<i>text</i>	The text you want for this text box.
<i>var_name</i>	The name you give to the this text box. If the text will not change, use the name "None".
<i>x</i>	Specifies the column position for displaying the text in dialog box units.
<i>y</i>	Specifies the row position for displaying the text in dialog box units.
<i>width</i>	Specifies the width for displaying the text in dialog box units.
<i>depth</i>	Specifies the depth or height position for displaying the text in dialog box units.

Parameter	Description
<i>style</i>	Style for the control. See "Reference - Script Controls" for details.

Example

The following example shows the creation of two right text boxes.

```
DialogBox Create MyDialog, 10, 10, 400, 100
RText "Total:", None, 10, 10, 128, 14
RText "", Info, 10, 40, 200, 14
```

Related Script Commands

[DialogBox Control](#), [DialogBox CText](#), [DialogBox LText](#)

DialogBox ScrollBar

Syntax

DialogBox ScrollBar [*name*], *text*, *var_name*, *x*, *y*, *width*, *depth*, [*style*]

Synonyms

DB SB

Description

The DialogBox ScrollBar command creates a scroll bar in a dialog box. You can add an action with DialogBox ControlCommand and set the range for scrolling with the DialogBox SetRange command.

See the "Reference - Script Controls" chapter for details of the styles, properties, methods and events for this control.

Parameters

The following table describes the parameters of the DialogBox ScrollBar command:

Parameter	Description
<i>name</i>	The name of the dialog box; designated in DialogBox New. (If you used DialogBox Create, you do not need this parameter.)
<i>text</i>	The text for the scroll bar caption.
<i>var_name</i>	The name you give to this scroll bar.
<i>x</i>	Specifies the column position for displaying the scroll bar in dialog box units.
<i>y</i>	Specifies the row position for displaying the scroll bar in dialog box units.
<i>width</i>	Specifies the width for displaying the scroll bar in dialog box units.
<i>depth</i>	Specifies the depth or height for displaying the scroll bar in dialog box units.

Parameter	Description
<i>style</i>	Style for the control. See "Reference - Script Controls" for details.

Example

Define a scrollbar and set its initial position

```
* The following will be between the DialogBox Create/EndCreate
ScrollBar Value, 2, 4, 82, 44
MyDialog.Value = 50
```

Related Script Commands

[DialogBox Control](#), [DialogBox SetRange](#)

DialogBox ScrollStep

Syntax

DialogBox ScrollStep [*name*], *x_step*, *y_step*

Synonyms

DB SP

Description

The DialogBox ScrollStep command specifies the amount the window moves with the scroll bars. Use with DialogBox MaxSize command.

Parameters

The following table describes the parameters of the DialogBox ScrollStep command:

Parameter	Description
<i>name</i>	The name of the dialog box designated in DialogBox New. (If you used DialogBox Create, you do not need this parameter.)
<i>x_step</i>	The scroll increment for the horizontal scroll bar.
<i>y_step</i>	The scroll increment for the vertical scroll bar.

Example

The following example shows how to set the MaxSize and ScrollStep commands:

```
DialogBox Create Cust 16,2,236,120
Caption "Cust"
Style WS_CAPTION | WS_VISIBLE | WS_SYSMENU | WS_THICKFRAME |
WS_MINIMIZEBOX |
WS_MAXIMIZEBOX | WS_TABSTOP | WS_GROUP | WS_HSCROLL | WS_VSCROLL
.
.
.
Pushbutton "Cancel",Cancel,184,3,40,12
ControlCommand Cancel,"DIALOGBOX END Cust,FALSE"
Pushbutton "OK",OK,184,20,40,12
ControlCommand OK,"DIALOGBOX END Cust,True"
MaxSize 250, 150
```

```
ScrollStep 10, 10  
EndCreate  
DialogBox Show Cust  
DialogBox Delete Cust
```

DialogBox Select

Syntax

DialogBox Select *name*, *var_name*, *string*

Synonyms

DB SL

Description

The DialogBox Select command selects a value from a list box or combo box in a currently displayed dialog box.

Parameters

The following table describes the parameters of the DialogBox Select command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New or DialogBox Create.
<i>var_name</i>	The name of the list or combo box control.
<i>string</i>	The name of the item in the list to select.

Example

The following example highlights CATEGORIES in the list box.

```
DialogBox Select Cust, Files, "CATEGORIES"
```

Related Script Commands

[DialogBox AddToList](#), [DialogBox ComboBox](#), [DialogBox ListBox](#), [DialogIndex](#), [DialogList](#)

DialogBox SetColor

Syntax

DialogBox SetColor *dlg_name, ctrl_name, foreground, [background]*

Synonyms

DB SC

Description

The DialogBox SetColor command enables you to set the text color and background color for an individual control within a dialog box.

DialogBox SetColor affects the entire control for static text, edit, and list boxes. When you use it with check boxes, radio buttons, group boxes and push buttons, the colors apply to only the text part of the controls. Only the background color changes on scroll bars. Do not use this command with other control types. If "ctrl_name" is None, the command changes colors for all controls named None.

You can change colors at any time, even after a dialog box is shown. See the example script: wintegrate\example\script\dbc_color.wis.

You can also spell this command DialogBox SetColour (Synonym DB SCU).

Parameters

The following table describes the parameters of the DialogBox SetColor command:

Parameter	Description
<i>dlg_name</i>	The name of the dialog box containing the control.
<i>ctrl_name</i>	The name of control to modify.
<i>foreground</i>	The foreground color.
<i>background</i>	The background color.

Example

The following example shows how to set the color during creation of a dialog box:

```
LText "Text Colored",text,5,20,88,12
SetColor text, RGB_White, RGB_Black
CheckBox "Checkbox",CheckBox,5,31,59,12
SetColor CheckBox, RGB_Blue, RGB_Yellow
```

DialogBox SetFont

Syntax

DialogBox SetFont *dlg_name, ctrl_name, font_name, size, [font style]*

Synonyms

DB SF

Description

The DialogBox SetFont command specifies the font used by an individual control. You can set the fonts used for a control to be different from the font used by default for the dialog box.

As with other dialogbox commands, you can omit the "DialogBox" and the *dlg_name* parameter when a script is in dialog box creation mode (see DialogBox Create). You can change the font at any time, even after a dialog box has been shown. Not all fonts support all the sizes and styles.

Note: The size of the control defined is not affected by this command, because the dialog units used are still based on the font chosen for the dialog box.

See the example script: wintegrate\example\script\dbfont.wis.

Parameters

The following table describes the parameters of the DialogBox SetFont command:

Parameter	Description
<i>dlg_name</i>	The name of dialog box containing the control.
<i>ctrl_name</i>	The name of control to modify.
<i>font_name</i>	The name of font to select.
<i>size</i>	The font point size.
<i>font style</i>	Optional styles for the font. See below

Values for font style

Style can contain a font weight and one or more effects.

Value	Description
<i>FONT_Light</i>	Light weight font
<i>FONT_Normal</i>	Normal weight font
<i>FONT_Bold</i>	Bold weight font
<i>FONT_Heavy</i>	Extra bold weight font
<i>FONT_Italic</i>	Italic effect
<i>FONT_Underline</i>	Underline effect
<i>FONT_StrikeOut</i>	Strike out effect

Example

The following example shows how to set the font for a text control during creation of a dialog box.

```
RText "Times Roman Italic",TRI,7,24,158,13
SetFont TRI, "Times New Roman",8,FONT_Italic | FONT_Normal
```

DialogBox SetKey

Syntax

DialogBox SetKey *dlg_name*, *key_name*, *key_value*

Synonyms

DB SE

Description

The DialogBox SetKey command defines an action for a key in a dialog box.

If the key_value starts with "\m", the rest of the definition is run as a script command. In any other case its value is sent to the host. Key names are specified when defining keys for the session. For example, the definition can be prefixed with Shift, Control, Alt or any combination of the three.

Parameters

The following table describes the parameters of the DialogBox SetKey command:

Parameter	Description
<i>dlg_name</i>	Name of dialog box for key definition.
<i>key_name</i>	Name of the key to be reprogrammed. A full list of key names can be found under "SetupKeyboard" in the "Reference - Script Menu Options" section of this manual.
<i>key_value</i>	Value for key.

Example

Set up Function keys for the Customer maintenance dialog box

```
* F1 to start a windows help file
DialogBox SetKey CustMaint, F1, "\mRun 'custhelp.hlp'"
The next example causes the F2 key to send text to the host when in
the Customer Maintenance
screen:
* F2 send text "Save" to host
DialogBox SetKey CustMaint, F2, "Save"
```

```
* F6 sets focus to Search box
DialogBox SetKey CustMaint, F6, "SetFocus CustMaint.Search"
* Ctrl Home sets the house checkbox
DialogBox SetKey CustMaint, Control_Home, "CustMaint.House = 1"
```

DialogBox SetMenu

Syntax

DialogBox SetMenu *dlg_name, menu_name*

Synonyms

DB SM

Description

The DialogBox SetMenu command enables you to use a previously defined menu in a dialog box. The menu must not be attached to another menu or dialog box. When creating the dialog box, add extra height to allow for the menu.

Note: A dialog box which has a menu cannot have the DS_MODALFRAME style (See the DialogBox Style command).

All menu commands can be applied to a dialog box menu, except the commands OnOpen, OnClose, and Help.

Parameters

The following table describes the parameters of the DialogBox SetMenu command:

Parameter	Description
<i>dlg_name</i>	Name of dialog box to contain the menu.
<i>menu_name</i>	Name of menu to use on the dialog box.

Example

The following example demonstrates the use of the DialogBox SetMenu command:

```
* TestMenuBar already created with Menu Create
DialogBox Create MenuTest 37,33,213,130
Caption "Test Menus"
Style WS_CAPTION|WS_VISIBLE|WS_SYSMENU
Font 8,"helv"
SetMenu TestMenuBar
```

DialogBox SetRange

Syntax

DialogBox SetRange [*name*], *var_name*, *min*, *max*

Synonyms

DB SR

Description

The DialogBox SetRange command specifies the range of values represented by a scroll bar.

Parameters

The following table describes the parameters of the DialogBox SetRange command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New. (If you used DialogBox Create, you do not need this parameter.)
<i>var_name</i>	The name of the scroll bar control designated in DialogBox ScrollBar.
<i>min</i>	The minimum value of the scroll bar. Values are 0 to 32767.
<i>max</i>	The maximum value of the scroll bar. Values are 0 to 32767.

Example

The following example sets the scroll bars across the bottom and side of the dialog box.

```
DialogBox Create Cust, 16, 2, 236, 120
.
.
.
ScrollBar Vert, 2, 4, 82, 44
SetRange Vert, 50, 125
```

Related Script Commands

[DialogBox ScrollBar](#)

DialogBox SetTabs

Syntax

DialogBox SetTabs *name, var_name, tabs*

Synonyms

DB SS

Description

The DialogBox SetTabs command sets the tab stops in a list box created with LBS_USETABSTOPS style.

Parameters

The following table describes the parameters of the DialogBox SetTabs command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New. (If you used DialogBox Create, you do not need this parameter.)
<i>var_name</i>	The name of list box.
<i>tabs</i>	A string containing comma-separated tab positions with an optional justification code after each position. See below.

Values for tabs

The justification codes following each tab position number in the comma-separated list are:

Value	Description
<i>L</i>	Left (default if no justification code is specified)
<i>C</i>	Center
<i>R</i>	Right

Example

The following example is taken from the GVDISP.WIS demonstration program.

```
ControlCommand Exit, "DialogBox End gvdlg,TRUE"  
ControlCommand OK, 'DialogBox SetTabs gvdlg,VarDisplay,  
gvdlg.tabs;Execute  
"gvdlg.VarDisplay = ":gvdlg.VarName"  
EndCreate
```

Related Script Commands

[DialogBox ListBox](#)

DialogBox SetText

Syntax

DialogBox SetText *name, var_name, text*

Synonyms

DB ST

Description

The DialogBox SetText command changes the text of a control. You may want to use this command to change the text of a push button control after an action has occurred.

Parameters

The following table describes the parameters of the DialogBox SetText command:

Parameter	Description
<i>name</i>	The name of the dialog box; designated in DialogBox New.
<i>var_name</i>	The name of the control.
<i>text</i>	The new text for the specified control.

Example

The following example is a part of the PRESTORE.WIS demonstration program.

```
Sub OnPClick(pno)
id = "P":pno
cname = "Prestore_":id
If Extract(Prestore_State,pno) Then
* Recording in progress
Capture Off $cname
Rep Prestore_text, pno, CaptureText($cname, DF_BS)
Capture Delete $cname
DialogBox SetText Prestore,$id,id
Rep Prestore_State,pno,0
Capture On $cname, "", "Keystrokes"
DialogBox SetText Prestore,$id,"R":pno
Rep Prestore_State,pno,1
Else
```

```
Type Extract(Prestore_text, pno)
EndIf
EndSub
```

Related Script Commands

[DialogBox PushButton](#)

DialogBox Show

Syntax

DialogBox Show *name, attach_to*

Synonyms

DB SH

Description

The DialogBox Show command displays a dialog box, allowing input only to the current dialog box. You must use this command to display the dialog box after you create it with DialogBox Create or DialogBox New, include all the necessary controls, and finish the dialog box with DialogBox End or DialogBox EndCreate.

Parameters

The following table describes the parameters of the DialogBox Show command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New.
<i>attach_to</i>	Name of the dialog box to attach to. An attached modal dialog box appears as normal but prevents any button clicks or key strokes from going to the dialog box it is attached to until it has been closed.

Example

The following example is taken from the GBUTTON.WIS demonstration program.

```
DialogBox Show GButton
```

In the next example, the dialog box CustLookup is attached to the CustMaint dialog box. You must dismiss CustLookUp before you can access CustMaint.

```
DialogBox Show CustLookup, CustMaint
```

Related Script Commands

[DialogBox Create](#), [DialogBox End](#), [DialogBox EndCreate](#), [DialogBox New](#),
[DialogBox Window](#), [DialogBox PopupWindow](#)

DialogBox ShowControl

Syntax

DialogBox ShowControl *dlg_name, ctrl_name, show*

Synonyms

DB SW

Description

The DialogBox ShowControl command shows or hides the specified control. The dialog box must be displayed at the time this command is issued.

Parameters

The following table describes the parameters of the DialogBox ShowControl command:

Parameter	Description
<i>dlg_name</i>	The name of the dialog box.
<i>ctrl_name</i>	The name of the control
<i>show</i>	True - show control, False - hide control

Related Script Commands

[DialogBox MoveControl](#)

Version

4.0.1 Original

DialogBox Size

Syntax

DialogBox Size [*name*], *width*, *depth*

Synonyms

DB SZ

Description

The DialogBox Size command changes the size of a dialog box. The width and depth parameters are in screen pixels rather than in dialog box units.

Parameters

The following table describes the parameters of the DialogBox Size command:

Parameter	Description
<i>name</i>	The name of the dialog box; designated in DialogBox New. (If you used DialogBox Create, you do not need this parameter.)
<i>width</i>	The new width of the dialog box, in screen pixel units.
<i>depth</i>	The new depth of the dialog box, in screen pixel units.

Example

In the following example, a control is set to reduce the size of the dialog using DialogBox Size.

```
DLib = 'Library "':ScriptFile:'';'  
Library 'wintsys\lib\server'  
DialogBox Create Cust 16,2,236,120  
Caption "Customers"  
Style  
WS_CAPTION | WS_VISIBLE | WS_SYSMENU | WS_THICKFRAME | WS_MAXIMIZEBOX  
| DS_MODALFRAME | WS_TABSTOP | WS_GROUP  
.  
.
```

```
.  
PushButton "Resize",reduce, 115,3,40,12  
ControlCommand reduce, "Execute 'DialogBox Size Cust,150,50'"
```

Related Script Commands

[DialogBox Move](#)

DialogBox Style

Syntax

DialogBox Style [*name*], *style*

Synonyms

DB SY

Description

The DialogBox Style command sets the style of a dialog box.

Parameters

The following table describes the parameters of the DialogBox Style command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New. (If you used DialogBox Create, you do not need this parameter.) style See the following Styles table.
<i>style</i>	See the following Styles table.

Values for style

The style parameter is always set at WS_VISIBLE and WS_CHILD. If you create a dialog box and do not specify DialogBox Style, the default styles are: DS_MODALFRAME | WS_SYSMENU | WS_CAPTION | WS_POPUP.

Value	Description
<i>WS_BORDER</i>	Adds a border to the dialog box.
<i>WS_DLGFRAME</i>	Makes the dialog box a modal frame with no title.
<i>WS_MODALFRAME</i>	Makes the dialog box a modal frame that can include a title.
<i>WS_THICKFRAME</i>	Creates a sizeable window frame.
<i>WS_CAPTION</i>	Includes the caption of the dialog box.

Value	Description
<i>WS_MINIMIZEBOX</i>	Includes the button to minimize the window.
<i>WS_MAXIMIZEBOX</i>	Includes the button to maximize the window.
<i>WS_SYSMENU</i>	Includes the system menu.
<i>WS_VSCROLL</i>	Adds a vertical scroll bar to the window.
<i>WS_HSCROLL</i>	Adds a horizontal scroll bar to the window.
<i>WS_POPUP</i>	Makes the dialog box a pop-up window that is independent of the current session.
<i>DS_ABSALIGN</i>	Aligns the dialog box relative to the screen origin instead of to the upper-left corner of the parent window.
<i>DS_SYSMODAL</i>	Allows only this dialog box to run.
<i>WS_VISIBLE</i>	Makes this dialog box visible.
<i>WS_DISABLED</i>	Does not allow input to the dialog box.

Example

The following example code is taken from the GVDISP.WIS example script.

```
* Display a global variable
* Version 2.0
if Not(IsDialog(gvdlg)) Then
DialogBox Create gvdlg 18, 18, 240, 150
CAPTION "Global Variable Display"
FONT 8, "helv"
STYLE WS_POPUP | WS_CAPTION | WS_SYSMENU | WS_MINIMISE
```

Related Script Commands

[DialogBox Caption](#), [DialogBox Create](#), [DialogBox ExStyle](#)

DialogBox TabControl

Syntax

DialogBox TabControl *dlg_name, ctrl_name, x, y, width, depth, [style]*

Synonyms

DB TC

Description

The DialogBox TabControl command creates a series of tabs in a dialog box.

See the "Reference - Script Controls" chapter for details of the styles, properties, methods and events for this control.

Parameters

The following table describes the parameters of the DialogBox TabControl command:

Parameter	Description
<i>dlg_name</i>	The name of the dialog box.
<i>ctrl_name</i>	The control name.
<i>x</i>	Specifies the column position for displaying the control in dialog box units.
<i>y</i>	Specifies the row position for displaying the control in dialog box units.
<i>width</i>	Specifies the width for displaying the control in dialog box units.
<i>depth</i>	Specifies the depth or height for displaying the control in dialog box units.
<i>style</i>	Style for the control. See "Reference - Script Controls" for details.

DialogBox ToolTip

Syntax

DialogBox ToolTip *dlg_name, ctrl_name, text*

Synonyms

DB TT

Description

The DialogBox ToolTip command creates a ToolTip (text to be displayed when the mouse is left over the specified control for a few seconds).

Note: ToolTips can be changed after the dialog box has been shown, but at least one ToolTip must have been defined in the dialog box prior to it being shown.

Parameters

The following table describes the parameters of the DialogBox ToolTip command:

Parameter	Description
<i>dlg_name</i>	Name of dialog box.
<i>ctrl_name</i>	Name of control to add the tip.
<i>text</i>	Text to display.

DialogBox Trackbar

Syntax

DialogBox Trackbar *dlg_name, ctrl_name, x, y, width, depth, [style]*

Synonyms

DB TB

Description

The Dialogbox Trackbar command defines a track bar control in the dialog box

Parameters

The following table describes the parameters of the DialogBox Trackbar command:

Parameter	Description
<i>dlg_name</i>	The name of the dialog box.
<i>ctrl_name</i>	The control name.
<i>x</i>	Specifies the column position for displaying the control in dialog box units.
<i>y</i>	Specifies the row position for displaying the control in dialog box units.
<i>width</i>	Specifies the width for displaying the control in dialog box units.
<i>depth</i>	Specifies the depth or height for displaying the control in dialog box units.
<i>style</i>	See "Reference - Script Controls" for details

Example

Code extract

```
DialogBox Create Track 0,0,179,75
...
Trackbar Trackbar2,21,27,136,18,TBS_TOP|TBS_LEFT|TBS_AUTOTICKS
OnEvent Trackbar2,"Scroll",'Print "track bar 2 is at":Track.Trackbar2'
Track.trackbar2.RangeMin = 0
```



```
Track.trackbar2.TickFrequency = 10
Track.trackbar2.PageSize = 10
...
EndCreate
```

Version

4.0.1 Original

4.0.4 BackColor property added

DialogBox TreeView

Syntax

DialogBox TreeView *dlg_name, ctrl_name, x, y, width, depth, [style]*

Synonyms

DB TV

Description

The DialogBox TreeView control creates a tree structure 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.

See the "Reference - Script Controls" chapter for details of the styles, properties, methods and events for this control.

Parameters

The following table describes the parameters of the DialogBox TreeView command:

Parameter	Description
<i>dlg_name</i>	The name of the dialog box.
<i>ctrl_name</i>	The control name.
<i>x</i>	Specifies the column position for displaying the control in dialog box units.
<i>y</i>	Specifies the row position for displaying the control in dialog box units.
<i>width</i>	Specifies the width for displaying the control in dialog box units.
<i>depth</i>	Specifies the depth or height for displaying the control in dialog box units.

Parameter	Description
<i>style</i>	Style for the control. See "Reference - Script Controls" for details.

DialogBox UpDownButton

Syntax

DialogBox UpDownButton *dlg_name, ctrl_name, x, y, width, depth, [style]*

Synonyms

DB UD

Description

The DialogBox UpDownButton control adds a spin box (a text box with up and down arrows) to a dialog box.

See the "Reference - Script Controls" chapter for details of the styles, properties, methods and events for this control.

Parameters

The following table describes the parameters of the DialogBox UpDownButton command:

Parameter	Description
<i>dlg_name</i>	The name of the dialog box.
<i>ctrl_name</i>	The control name.
<i>x</i>	Specifies the column position for displaying the control in dialog box units.
<i>y</i>	Specifies the row position for displaying the control in dialog box units.
<i>width</i>	Specifies the width for displaying the control in dialog box units.
<i>depth</i>	Specifies the depth or height for displaying the control in dialog box units.
<i>style</i>	Style for the control. See "Reference - Script Controls" for details.

DialogBox Validate

Syntax

DialogBox Validate [*name*], *var_name*, *command*

Synonyms

DB VL

Description

The DialogBox Validate command specifies which command to run when a dialog box control loses the input focus.

Note: Use this command only with edit, list, and combo box controls.

Parameters

The following table describes the parameters of the DialogBox Validate command:

Parameter	Description
<i>name</i>	The name of the dialog box designated in DialogBox New. (If you used DialogBox Create, you do not need this parameter.)
<i>var_name</i>	The name of the control.
<i>command</i>	The command to run when the control's action occurs, and it loses the input focus.

Example

The following example is taken from the PRODM2.WIS program. It shows the controls being created and the validate commands for the controls.

```
LTEXT "Prod. Ref.",None,7,9,36,8,WS_CHILD | WS_VISIBLE | WS_GROUP
EDITTEXT ProdRef,53,6,62,12,ES_LEFT | WS_CHILD | WS_VISIBLE |
WS_BORDER | WS_TABSTOP
LTEXT "Name:", None,7,25,38,8,WS_CHILD | WS_VISIBLE | WS_GROUP
EDITTEXT Name,53,23,135,12,ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER
| WS_TABSTOP
.
```

```
.  
Validate ProdRef, DLib:'If NotCancel() Then OnValidateId'  
Validate Name, DLib:"If NotCancel() Then OnValidateName"  
Validate Price, DLib:'If NotCancel() Then OnValidatePrice'  
InitCommand DLib:"ClearEntry"  
EndCreate
```

Related Script Commands

[DialogBox ControlCommand](#), [DialogBox Default](#), [DialogBox OnEvent](#)

DialogBox Variable

Syntax

DialogBox Variable *dlg_name, var_list*

Synonyms

DB VR

Description

The DialogBox Variable command creates a variable for the dialog box. The variable is then accessed in the same way as the default value for a control.

Parameters

The following table describes the parameters of the DialogBox Variable command:

Parameter	Description
<i>dlg_name</i>	The name of the dialog.
<i>var_list</i>	List of one or more comma separated variable optionally with there initial value set by putting and equals sign (=) followed by the value after the variable name

DialogBox WebStyleName

Syntax

DialogBox WebStyleName *name*

Synonyms

DB WY

Description

This command sets the web style for a dialog box. It must be set before any controls are defined.

The web styles are defined in a cascading style sheet and are used by the Java Client only. All other versions ignore this command. The web style allows the default fonts and colours for the dialog box to be changed.

See the Java Client documentation on how to specify the style sheet used.

Parameters

The following table describes the parameters of the DialogBox WebStyleName command:

Parameter	Description
<i>name</i>	Name of the web style to use

Example

The following is the first few lines of a simple dialog

```
DialogBox Create WST 18,1,338,291
Caption "Web style test"
Style WS_CAPTION|WS_POPUP|WS_VISIBLE|WS_SYSMENU|DS_MODALFRAME
Font 14,"Script"
WebStyleName ".bluetest"
LText "Text",T1,15,11,87,12
```

Related Script Commands

[WebStyle Global](#), [WebStyle ControlType](#)

Version

6.0.0 Original version

DialogBox Window

Syntax

DialogBox Window *name*, [*attach_to*]

Synonyms

DB WN

Description

The DialogBox Window command displays a dialog box as a window or modeless dialog box, allowing the user to access several dialog boxes at the same time.

Parameters

The following table describes the parameters of the DialogBox Window command:

Parameter	Description
<i>name</i>	The name of the dialog box designated in DialogBox New. (If you used DialogBox Create, you do not need this parameter.)
<i>attach_to</i>	Specifies which dialog box to attach this window. When a window is attached, it always displays on top of the dialog box, even when the dialog box has the focus.

Example

The following example is part of the WC.WIS demonstration program.

```
DialogBox Window wc
```

Related Script Commands

[DialogBox Show](#), [DialogBox PopupWindow](#)

DialogBox WindowState

Syntax

DialogBox WindowState *dlg_name, option*

Synonyms

DB WS

Description

The Dialogbox WindowState command minimizes, maximizes, hides or restores a dialog box. The dialog box must be displayed at the time this command is issued.

Parameters

The following table describes the parameters of the DialogBox WindowState command:

Parameter	Description
<i>dlg_name</i>	The name of the dialog box.
<i>option</i>	Show state for the window. See below

Values for option

Value	Description
<i>0</i>	Restore the dialog box to normal size after it has been minimized or maximized
<i>1</i>	Minimize the dialog box
<i>2</i>	Maximize the dialog box
<i>3</i>	Hide the dialog box

Example

Minimize the wc dialog box

```
DialogBox WindowState wc, 1
```

Switch pages (implemented as child dialog boxes) in a dialog box

```
* Show the new page
DialogBox WindowState Options,0
* Hide the page we are switching from
DialogBox WindowState General,3
```

Related Script Commands

[DialogWindowState](#)

Version

4.0.1 Original

5.1.1 Option 3 hidden added

DialogIndex

Syntax

DialogIndex *name*, *var_name*

Description

The DialogIndex function returns the line number of current selection in a list box or combo box. For a multiple selection list box, it returns a list of all selections separated by a carriage return.

Parameters

The following table describes the parameters of the DialogIndex command:

Parameter	Description
<i>name</i>	The name of the dialog box.
<i>var_name</i>	The name of the list or combo box control.

Return Value

Numbers of the lines currently selected separated by a cr or null (" ") if there is no selection or the control is not shown

Related Script Commands

[DialogBox ComboBox](#), [DialogBox ListBox](#), [DialogList](#)

DialogList

Syntax

DialogList *name*, *var_name*

Description

The DialogList command returns all the lines from the list control list with each line separated by a carriage return.

Parameters

The following table describes the parameters of the DialogList command:

Parameter	Description
<i>name</i>	The name of the dialog box.
<i>var_name</i>	The name of the list or combo box control.

Return Value

List of all the items in the listbox/combobox.

Example

The following example is taken from the WC.WIS demonstration program.

```
* Run the command and copy it to list part of combo box
Sub RunCmnd()
pos = Locate(DialogList(wc,Cmnd), wc.Cmnd, "EU")
If pos = 0 Then DialogBox AddToList wc,cmnd,wc.cmnd,1
Execute wc.Cmnd
If IsShown(wc) Then DialogBox SetText wc,Cmnd, ''
EndSub
```

Related Script Commands

[DialogBox ComboBox](#), [DialogBox ListBox](#), [DialogIndex](#)

DialogListFind

Syntax

DialogListFind *name, control_name, text, [start_pos], [prefix]*

Description

The DialogListFind command returns the position (numbered from 1) of a string in a list box or combo box. The search is not case sensitive. If start_pos is specified, the search starts at the following entry and loops to check from the beginning of the list.

Parameters

The following table describes the parameters of the DialogListFind command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New or dialogBox Create.
<i>control_name</i>	The name of the control.
<i>text</i>	The text or string to search for.
<i>start_pos</i>	The position from which to start the search.
<i>prefix</i>	Searches only the first characters of a string.

Return Value

The line number where the text was found or 0 if it wasn't found

Example

Find the first entry starting WIN.PROGS in BPList

```
pos = DialogListFind(MyBar, BPList, "WIN.PROGS", ,TRUE)
```

Find the location of Widgets in products list and delete it

```
pos = DialogListFind(ProdMaint, ProductList, "Widgets")
If pos > 0 Then
    DialogBox DeleteFromList ProdMaint, ProductList, pos
```

EndIf

Related Script Commands

[DialogBox AddToList](#), [DialogBox ComboBox](#), [DialogBox DeleteFromList](#),
[DialogBox ListBox](#), [DialogBox Select](#), [DialogIndex](#), [DialogList](#)

DialogNextEvent

Syntax

DialogNextEvent *name, option*

Description

The DialogNextEvent function returns a notification of the next event (if any) when it is called from a validation routine in a dialog box if DialogBox Option "V1" or "V2" is set.

Parameters

The following table describes the parameters of the DialogNextEvent command:

Parameter	Description
<i>name</i>	The name of the dialog box, designated in DialogBox New.
<i>option</i>	Options. See below

Values for option

Value	Description
<i>1</i>	Returns the event code.
<i>2</i>	Returns the control name.
<i>3</i>	Returns the event code and control name. This is the default. See the following Event Codes table.

Return Value

Event code and/or control name depending on the setting of option. See below for event codes

The following table describes the event codes for the DialogNextEvent function:

Value	Description
<i>C</i>	Button click
<i>V</i>	Validation
<i>D</i>	Default
<i>L</i>	List box double click or combo box selection.

Example

The following example is part of the PRODM2.WIS demonstration program.

```
* NotCancel, checks if Cancel button has been clicked
* If it has it overrides the validation and returns False
Define NotCancel()
If DialogNextEvent(ProdM) = "C;Cancel" Then
DB InputOK ProdM, True
return False
EndIf
return True
```

Related Script Commands

[DialogBox Option](#), [DialogBox Validate](#)

DialogRect

Syntax

DialogRect *dlg_name*, [*ctrl_name*], [*option*]

Description

The DialogRect function returns rectangle which gives the position and size of the specified dialog box or control in pixels. The rectangle is return as cr separated values in the order left,top,right, bottom.

Parameters

The following table describes the parameters of the DialogRect command:

Parameter	Description
<i>dlg_name</i>	The name of the dialog box.
<i>ctrl_name</i>	The name of the control. Leave blank if the information is required for the dialog box
<i>option</i>	Which rectangle to return. See below

Values for option

Value	Description
0	Position of dialog box. For a control this is the position relative to the dialog box which contains it
1	Client rectangle. This is 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
3	Position of dialog or control relative to its parent. Returns the absolute position for a dialog box that does not have a parent

Value	Description
4	Client rectangle relative to the edge of the control.

Return Value

The return value is the position and size of the specified dialog box or control in pixels. The rectangle is return as cr separated values in the order left,top,right,bottom.

Example

Position a button horizontally centered in a dialog box

```
client_rect = DialogRect(MyDialog,,1)
button_rect = DialogRect(MyDialog,MyButton)
* new_pos is the width of the dialog box client area minus the
* width of the button divided by 2
new_pos = Extract(client_rect,3) - (Extract(button_rect,3) -
Extract(button_rect,1))
new_pos = new_pos / 2
DialogBox MoveControl MyDialog,MyButton, new_pos
```

Version

4.0.2 Original

4.2.1 Added options 3 and 4

DialogUnit

Syntax

DialogUnit *name, option, value*

Description

The DialogUnit function converts between dialog box units and screen pixels for a given dialog box.

A dialog box unit is calculated as 1/4 of the average width and 1/8 of the average depth of the font used for the dialog box.

Dialog units are used to allow dialog box and control positions and sizes to be defined indepently of the current users desktop resolution.

Parameters

The following table describes the parameters of the DialogUnit command:

Parameter	Description
<i>name</i>	The name of the dialog box to convert.
<i>option</i>	Conversion to perform. See Table below.
<i>value</i>	The value to convert.

Values for option

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.

Value	Description
4	Convert column dialog units to column pixels.
5	Convert row dialog units to row pixels.
6	Convert multiple column dialog units to column pixels
7	Convert multiple row dialog units to row pixels.

Return Value

Converted value depending on the option. For options that convert multiple items, any numbers in the value are converted. Nonnumeric values are not converted.

Example

Calculate position to center a dialog

```
* Return x and y to center dialog in Terminal Window, width
* and depth are the width and depth in dialog units as used
* when the dialog was created with the DialogBox Create
* command.
Sub CenterPos($dlgname, $xvar,$yvar, width, depth)
* Get current terminal width
twidth = Field(WindowPos(0), ",", 3)
tdepth = Field(WindowPos(0), ",", 4)
* Convert width and depth from dialog units to pixels
width = DialogUnit($dlgname, 4, width)
depth = DialogUnit($dlgname, 5, depth)
* Add size of caption to height
depth+= SystemMetrics(4)
* Calculate position and offset by Terminals top left
$xvar = (twidth - width) / 2 +
$yvar = (tdepth - depth) / 2 +
Field(WindowPos(1), ",", 2)
EndSub
```

Related Script Commands

[SystemMetrics](#), [WindowPos](#)

DialogWindowState

Syntax

DialogWindowState *dlg_name*

Description

The DialogWindowState function returns a number that identifies the current show state of a dialog box.

Parameters

The following table describes the parameters of the DialogWindowState command:

Parameter	Description
<i>dlg_name</i>	The name of the dialog box.

Return Value

The minimized/maximized/hidden state of the window. See table below.

Value	Description
<i>0</i>	Dialog box is shown normally
<i>1</i>	Dialog box is Minimized
<i>2</i>	Dialog box is Maximized
<i>3</i>	Dialog box is hidden

Example

See if user has minimized the command dialog box wc

```
If DialogWindowState(wc) = 1 Then
  MsgBox "You've minimized wc"
EndIf
```

Related Script Commands

[DialogBox](#) [WindowState](#)

Version

4.0.1 Original

5.1.1 Returns 3 for a hidden window

DirExist

Syntax

DirExist *directory*

Description

The DirExist function determines whether a directory exists.

Parameters

The following table describes the parameters of the DirExist command:

Parameter	Description
<i>directory</i>	The full path name of the directory to check.

Return Value

True if the directory exists, otherwise False

Example

The following example checks for a directory, then creates it if it does not exist.

```
* Create the pctest directory if it does not exist
If DirExist("C:\pctest") Then
  MsgBox "The directory pctest exists"
Else File CreateDir "C:\pctest"
EndIf
```

Related Script Commands

[FileExist](#)

Display At

Syntax

Display At *x*, *y*

Synonyms

DP AT

Description

The Display command moves the cursor to a specified position.

Parameters

The following table describes the parameters of the Display At command:

Parameter	Description
<i>x</i>	The new column position for the cursor.
<i>y</i>	The new row position for the cursor.

Example

The following example uses Display At to move the cursor to a position for the next command,

```
Display Text.  
* Demonstrate Display commands  
* Clear the screen  
Display FF  
* Display a box with a block border  
Display Box 0,10,5,5,3  
* Display a box with a double line border, filled with w's  
Display Box 10,10,15,5,2  
Display CharFill 11,9,14,6,"w"  
* Display the CharFill command with no box  
Display CharFill 20,10,25,5,"@"  
* Move the cursor and display text  
Display At 0,15  
Display Text"This is the Display commands demo"
```

Related Script Commands

[Display Command](#), [Display Box](#), [Display CharFill](#), [Display Direct](#), [Display Effect](#),
[Display EffectFill](#), [Display Text](#), [Display UpdateOff](#), [Display UpdateOn](#)

Display Attribute

Syntax

Display Attribute *type, new_value*

Synonyms

DP AR

Description

The Display Attribute directly sets the attributes to be used for displaying characters.

Parameters

The following table describes the parameters of the Display Attribute command:

Parameter	Description
<i>type</i>	The type of attribute to set. See table below
<i>new_value</i>	The new value of the attribute. See the description in the type table below for values.

Values for type

The type specifies the part of the current attribute that is changed. Note: Setting the effect turns off the current colors and setting a color turns off the current effect.

Value	Description
<i>1</i>	Effect (see table below)
<i>2</i>	Font bank (0-3)
<i>4</i>	Foreground color (0-15)
<i>8</i>	Background color (0-15)
<i>12</i>	Foreground color + 16 * background colour
<i>31</i>	All attributes

Values for new_value

The effect value is either 0 or any combination of the following values added together.

Value	Description
<i>1</i>	Dim
<i>2</i>	Reverse
<i>4</i>	Flash
<i>8</i>	Underline
<i>16</i>	Bold
<i>32</i>	Secret

Example

Saving and restoring the display attribute

```
* Save the current attribute using the attribute function
Display Effect "Reverse"; * Change the attribute to reverse
Display Text "Reverse"
Display Attribute 4,COL_Red;* Change the foreground color to red
Display Text "This is red"
Color COL_LightBlue ;* An easier way to change the colour
Display Text "Light Blue"
Display Attribute 31, save_attr
Display Text "We should be back where we started"
```

Related Script Commands

[Attribute](#)

Version

4.1.1 Original version

Display Box

Syntax

Display Box *x1, y1, x2, y2, style*

Synonyms

DP BX

Description

The Display Box command displays a box in the wIntegrate terminal window. If *x1* is the same as *x2*, it displays a vertical line. If *y1* is the same as *y2*, it displays a horizontal line.

Parameters

The following table describes the parameters of the Display Box command:

Parameter	Description
<i>x1</i>	The x position of the top left of the box.
<i>y1</i>	The y position of the top left of the box.
<i>x2</i>	The x position of the bottom right of the box.
<i>y2</i>	The y position of the bottom right of the box.
<i>style</i>	See the following Style table.

Values for style

Value	Description
<i>0</i>	Displays no border
<i>1</i>	Displays a single line border.
<i>2</i>	Displays a double line border.
<i>3</i>	Displays a block border made of dots.
<i>4</i>	Displays a solid block border.

Value	Description
(+)16 [* 0-4] (see above)	Displays the vertical lines as a single line and the sides as the style added. Example: 2+16 creates a box with double lines on the top and bottom and single lines on the right and left sides. You can multiply this result by a number 0-4 to change the style for the sides.
(+)256	Leaves the box empty.
(+)512	Merges box lines with lines already on the screen. (See the DISPBOX.WIS script for more information on merging boxes.)

Example

The following example demonstrates several Display commands.

```
* Clear the screen
Display FF
* Display a box with a block border
Display Box 0,10,5,5,3
* Display a box with a double line border, filled with w's
Display Box 10,10,15,5,2
Display CharFill 11,9,14,6,"w"
* Display the CharFill command with no box
Display CharFill 20,10,25,5,"@"
* Move the cursor and display text
Display At 0,15
Display Text"This is the Display commands demo"
```

Related Script Commands

[Display Command](#), [Display At](#), [Display CharFill](#), [Display Direct](#), [Display Effect](#), [Display EffectFill](#), [Display Text](#), [Display UpdateOff](#), [Display UpdateOn](#)

Display CharFill

Syntax

Display CharFill *x1, y1, x2, y2, char, [font]*

Synonyms

DP CF

Description

The Display CharFill command fills specified areas with a character.

Parameters

The following table describes the parameters of the Display CharFill command:

Parameter	Description
<i>x1</i>	The x position of the top left of the area.
<i>y1</i>	The y position of the top left of the area.
<i>x2</i>	The x position of the bottom right of the area.
<i>y2</i>	The y position of the bottom right of the area.
<i>char</i>	The character with which to fill the area.
<i>font</i>	font bank to use (0 to 3) or "M" to use the active character mapping from the emulation. Default 0.

Example

The following example demonstrates several Display commands.

```
* Demonstrate Display commands
* Clear the screen
Display FF
* Display a box with a block border
Display Box 0,10,5,5,3
* Display a box with a double line border, filled with w's
Display Box 10,10,15,5,2
Display CharFill 11,9,14,6,"w"
* Display the CharFill command with no box
Display CharFill 20,10,25,5,"@"
```

```
* Move the cursor and display text
Display At 0,15
Display Text"This is the Display commands demo"
```

Related Script Commands

[Display Command](#), [Display At](#), [Display Box](#), [Display Direct](#), [Display Effect](#), [Display EffectFill](#), [Display Text](#), [Display UpdateOff](#), [Display UpdateOn](#)

Version

4.1.1 Added font parameter

Display Command

Syntax

Display Command *emulation_command*, [*param1*], [*param2*]

Synonyms

DP CM

Description

The Display command executes an emulation command.

The word "Command" is optional. e.g. "Display Command FF" can also be written as "Display FF"

Parameters

The following table describes the parameters of the Display Command command:

Parameter	Description
<i>emulation_command</i>	The emulation command to run. See table below
<i>param1</i>	A parameter that the emulation command requires.
<i>param2</i>	Any additional parameter that the emulation command requires.

Values for emulation_command

Value	Description
<i>CursorDown n</i>	The number of positions to move down. Defaults to 1.
<i>CursorLeft n</i>	The number of positions to move left. Defaults to 1.
<i>CursorRight n</i>	The number of positions to move right. Defaults to 1.
<i>CursorUp n</i>	The number of positions to move up. Defaults to 1.

Value	Description
<i>DeleteChar n</i>	The number of characters to delete. Defaults to 1.
<i>DeleteLine n</i>	The number of lines to delete. Defaults to 1.
<i>Effect effect_name</i>	Any of the effect names singly or concatenated.
<i>FF</i>	Formfeed - clears the screen
<i>InsertChar n</i>	The number of characters to insert. Defaults to 1.
<i>InsertLine n</i>	The number of lines to insert. Defaults to 1.
<i>Move x,y</i>	The column and row positions to move to.
<i>MoveX x</i>	The column position to move to.
<i>MoveY y</i>	The row position to move to.

Example

In the following example, the Display command is used to clear the screen:

```
Display FF
```

* We could have also written this as:
Display Command FF

In the next example, the Display command is used moves to line 5:

```
Display MoveY 5
```

In the next example, the Display command is used to move 10 characters to the right:

```
Display CursorRight 10
```

In the next example, the Display command is used to change the current effect:

```
Display Effect "DimReverse"
```

Related Script Commands

[Display At](#), [Display Box](#), [Display CharFill](#), [Display Direct](#), [Display Effect](#), [Display EffectFill](#), [Display Text](#), [Display UpdateOff](#), [Display UpdateOn](#)

Display Direct

Syntax

Display Direct *text*

Synonyms

DP DR

Description

The Display Direct command puts data directly onto the wIntegrate terminal screen without putting it through the emulation. This means that, unlike with the Display Text command, there is no character mapping and character codes less than 32 and greater than 126 are displayed directly with the current font without triggering escape sequences.

Parameters

The following table describes the parameters of the Display Direct command:

Parameter	Description
<i>text</i>	Text to display. Use "\cNNNN" to display characters from the IBSTFont character set.

Related Script Commands

[Display Command](#), [Display At](#), [Display Box](#), [Display CharFill](#), [Display Effect](#), [Display EffectFill](#), [Display Text](#), [Display UpdateOff](#), [Display UpdateOn](#)

Display Effect

Syntax

Display Effect *name*

Synonyms

DP

Description

The Display Effect command selects the effect to display in the wIntegrate window.

Parameters

The following table describes the parameters of the Display Effect command:

Parameter	Description
<i>name</i>	The name of the effect. You can use any combination (e.g., BoldUnderline) of the following: Normal, Bold, Flash, Dim, Reverse, Underline, Secret

Example

The following example is part of the EFFECT.WIS demonstration program. It demonstrates how to use effects.

```
* Demonstrate wIntegrate's Effects
* Version 2.0
Display FF
Display At 29,0
Display Text "Effects Colour Chart"
x = 0
y = 0
j = 1
Loop
EffectName = ""
if j & 1 Then EffectName = "Dim"
if j & 2 Then EffectName := "Reverse"
if j & 4 Then EffectName := "Underline"
if j & 8 Then EffectName := "Flash"
if j & 16 Then EffectName := "Bold"
```

```
if j & 32 Then EffectName := "Secret"
j += 1
y+=1
if y = 22 Then
y = 1
x += 26
EndIf
Display At x+25,y
Display Effect "Normal"
Display At x,y
Display Text Left(EffectName,24)
Until j = 64
Repeat
Display At 0,22
Display Effect "Normal"
```

Related Script Commands

[Display Command](#), [Display At](#), [Display Box](#), [Display CharFill](#), [Display Direct](#),
[Display EffectFill](#), [Display Text](#), [Display UpdateOff](#), [Display UpdateOn](#)

Display EffectFill

Syntax

Display EffectFill *x1, y1, x2, y2, option, [effect_name], [fore_color], back_color*

Synonyms

DP EF

Description

The Display EffectFill command fills specified areas with an effect or color.

Parameters

The following table describes the parameters of the Display EffectFill command:

Parameter	Description
<i>x1</i>	The x position of the top left of the area.
<i>y1</i>	The y position of the top left of the area.
<i>x2</i>	The x position of the bottom right of the area.
<i>y2</i>	The y position of the bottom right of the area.
<i>option</i>	The option specifies whether to fill the area with an effect or a color. 0 - Use an effect, 1 - Use color.
<i>effect_name</i>	The effect with which to fill the area. Omit this parameter if option is 1. Any of the effects can be used individually or concatenated as a string, (e.g. DimReverse, BoldUnderline): Normal, Bold, Flash, Dim, Reverse, Underline, Secret
<i>fore_color</i>	Specifies the foreground color. See the following Colors table. Omit this parameter if option = 0
<i>back_color</i>	Specifies the background color. See the following Colors table (for the fore_color parameter). Omit this value if option = 0

Values for fore_color

The colors for the foreground can be a number of one of the following constants:

Value	Description
<i>COL_Black</i>	0
<i>COL_Blue</i>	1
<i>COL_Green</i>	2
<i>COL_Cyan</i>	3
<i>COL_Red</i>	4
<i>COL_Magenta</i>	5
<i>COL_Brown</i>	6
<i>COL_LightGray</i> <i>or</i> <i>COL_LightGrey</i>	7
<i>COL_Gray or COL_Grey</i>	8
<i>COL_LightBlue</i>	9
<i>COL_LightGreen</i>	10
<i>COL_LightCyan</i>	11
<i>COL_LightRed</i>	12
<i>COL_LightMagenta</i>	13
<i>COL_Yellow</i>	14
<i>COL_White</i>	15

Example

The following example shows how to use color rather than effects with this command.

```
* Demonstrate Display commands
* Clear the screen
Display FF
* Display a box with a block border
Display Box 0,10,5,5,3
* Display a box with a double line border, with magenta fill
```

```
Display Box 10,10,15,5,2
Display EffectFill 10,10,15,5,1,2,5
* Display the CharFill command with no box
Display CharFill 20,10,25,5,"@"
* Move the cursor and display text
Display At 0,15
Display Text"This is the Display commands demo"
```

Related Script Commands

[Display Command](#), [Display At](#), [Display Box](#), [Display CharFill](#), [Display Direct](#),
[Display Effect](#), [Display Text](#), [Display UpdateOff](#), [Display UpdateOn](#)

Display Text

Syntax

Display Text *text*

Synonyms

DP TX

Description

The Display Text command sends text to the screen via the terminal emulation.

The text to be sent to the screen is first converted in the same way as the BStoAscii function and then converted to the current host encoding before going to the emulation for display.

Parameters

The following table describes the parameters of the Display Text command:

Parameter	Description
<i>text</i>	The text you want to display on the screen.

Example

The following example displays the text "Now starting the user application":

```
Display Text "Now starting the user application"
```

Related Script Commands

[Display Command](#), [Display At](#), [Display Box](#), [Display CharFill](#), [Display Direct](#),
[Display Effect](#), [Display EffectFill](#), [Display UpdateOff](#), [Display UpdateOn](#)

Display UpdateOff

Syntax

Display UpdateOff [*formfeed_on*]

Synonyms

DP UF

Description

The Display UpdateOff command stops updating the wIntegrate screen. When screen updating is off, characters received from the host or PC do not display on the screen until updating mode is turned back on. Any section of the screen that is repainted (for example, when another application or dialog box is placed in front of the screen and then moved), displays with the new contents.

Note: If *formfeed_on* is set to true, the next form feed turns screen updating back on.

Parameters

The following table describes the parameters of the Display UpdateOff command:

Parameter	Description
<i>formfeed_on</i>	Set to true to automatically turn display updating back on at the next formfeed

Example

The following example turns off the display while listing WIN.ORDER until the "Enter to continue" prompt displays and then turns the display back on:

```
Display UpdateOff
Enter "LIST WIN.ORDER CUSTNAME"
Wait ForText "Enter"
Display UpdateOn
```

Related Script Commands

[Display Command](#), [Display At](#), [Display Box](#), [Display CharFill](#), [Display Direct](#),
[Display Effect](#), [Display EffectFill](#), [Display Text](#), [Display UpdateOn](#), [Screen Off](#),
[Screen On](#)

Display UpdateOn

Syntax

Display UpdateOn

Synonyms

DP UN

Description

The Display UpdateOn command restarts updating of the wIntegrate screen after a Display UpdateOff command has been issued.

Parameters

None

Example

The following example turns off the display while listing WIN.ORDER until the "Enter to continue" prompt displays and then turns the display back on:

```
Display UpdateOff
Enter "LIST WIN.ORDER CUSTNAME"
Wait ForText "Enter"
Display UpdateOn
```

Related Script Commands

[Display Command](#), [Display At](#), [Display Box](#), [Display CharFill](#), [Display Direct](#), [Display Effect](#), [Display EffectFill](#), [Display Text](#), [Display UpdateOff](#), [Screen Off](#), [Screen On](#)

DisplayOn

Syntax

DisplayOn

Description

The DisplayOn function checks if the screen display is on. When the screen display is off the screen is shown as blank until the Display is turned back on again.

Parameters

None

Return Value

True (1) if ths display is currently on otherwise False (0).

Related Script Commands

[Display Command](#), [ScreenOn](#)

Version

5.0 Original

Draw Arc

Syntax

Draw Arc [*x1*], [*y1*], *x2*, *y2*, *xs*, *ys*, *xe*, *ye*

Synonyms

DR AR

Description

The Draw Arc command draws the arc of an ellipse within a rectangle defined by the parameters X1,Y1 to X2,Y2. If X1 and Y1 are omitted, then the top left of the rectangle is the current graphic cursor position.

Tip: This command uses the current pen and color set by Draw Pen.

Parameters

The following table describes the parameters of the Draw Arc command:

Parameter	Description
<i>x1</i>	The column position of the top left of the rectangle.
<i>y1</i>	The row position of the top left of the rectangle.
<i>x2</i>	The column position of the bottom right of the rectangle.
<i>y2</i>	The row position of the bottom right of the rectangle.
<i>xs</i>	The column position of the start of the arc angle.
<i>ys</i>	The row position of the start of the arc angle.
<i>xe</i>	The column position of the end of the arc angle
<i>ye</i>	The row position of the end of the arc angle.

Example

The following example line draws a small arc in the middle of the wIntegrate window:

```
Draw Arc 300, 40, 600, 80, 200, 45, 100, 90
```

Related Script Commands

[Draw Brush](#), [Draw Chord](#), [Draw Ellipse](#), [Draw Erase](#), [Draw Font](#), [Draw Image](#), [Draw Line](#), [Draw Move](#), [Draw Pen](#), [Draw Pie](#), [Draw Polygon](#), [Draw Rect](#), [Draw Size](#), [Draw Text](#), [Draw To](#)

Draw Brush

Syntax

Draw Brush *color, style*

Synonyms

DR BR

Description

The Draw Brush command sets the brush color and style. The brush is used by several other Draw commands. You can change the Pen and Brush colors and styles for different components of a drawing. See the "Draw Module" example in "Appendix A - Scripting Libraries and Modules" for an example of different ways to use Draw Brush.

Parameters

The following table describes the parameters of the Draw Brush command:

Parameter	Description
<i>color</i>	Specifies the color of the brush. Use the RGB command or one of the following constants: RGB_Black, RGB_Blue, RGB_Green, RGB_Cyan, RGB_Red, RGB_Magenta, RGB_Brown, RGB_Gray, RGB_Grey, RGB_LightGray, RGB_LightGrey, RGB_LightBlue, RGB_LightGreen, RGB_LightCyan, RGB_LightRed, RGB_LightMagenta, RGB_Yellow, RGB_White
<i>style</i>	Specifies the style of the brush. See the following Styles table.

Values for style

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

Value	Description
<i>BRUSH_Null</i>	Specifies no color and ignores the COLOR argument.
<i>BRUSH_Horizontal</i>	Specifies horizontal lines.
<i>BRUSH_Vertical</i>	Specifies vertical lines.
<i>BRUSH_FDiagonal</i>	Specifies diagonal slashes from left to right.
<i>BRUSH_BDiagonal</i>	Specifies diagonal slashes from right to left.
<i>BRUSH_Cross</i>	Specifies cross-hatching.
<i>BRUSH_DiagCross</i>	Specifies diagonal cross-hatching

Example

The following example is taken from the PIECHART.WIS demonstration program.

```
* Labels
Draw Rect tx,ty,tx+char_size,ty+char_size * 2
Draw Pen RGB_Black
Draw Brush 0,BRUSH_Null
Draw Text tx + char_size+10, ty, label
ty += char_size * 3
```

Related Script Commands

[Draw Arc](#), [Draw Chord](#), [Draw Ellipse](#), [Draw Erase](#), [Draw Font](#), [Draw Image](#), [Draw Line](#), [Draw Move](#), [Draw Pen](#), [Draw Pie](#), [Draw Polygon](#), [Draw Rect](#), [Draw Size](#), [Draw Text](#), [Draw To](#)

Draw Chord

Syntax

Draw Chord [*x1*], [*y1*], *x2*, *y2*, *xs*, *ys*, *xe*, *ye*

Synonyms

DR CH

Description

The Draw Chord command draws a chord of an ellipse bounded by the rectangle *x1*, *y1*,*tox2*, *y2*. The chord is drawn with the current pen color and style you set with the Draw Pen command, and the brush color and style you set with the Draw Brush command. If you omit the *X1* and *Y1* coordinates, the starting position of the arc is the current cursor position.

Parameters

The following table describes the parameters of the Draw Chord 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>xs</i>	The column position of the start of the arc angle.
<i>ys</i>	The row position of the start of the arc angle.
<i>xe</i>	The column position of the end of the arc angle
<i>ye</i>	The row position of the end of the arc angle.

Example

The following example shows the Draw Chord command:

```
Draw Chord 300, 40, 600, 80, 320,40, 500, 80
```

Related Script Commands

[Draw Arc](#), [Draw Brush](#), [Draw Ellipse](#), [Draw Erase](#), [Draw Font](#), [Draw Image](#), [Draw Line](#), [Draw Move](#), [Draw Pen](#), [Draw Pie](#), [Draw Polygon](#), [Draw Rect](#), [Draw Size](#), [Draw Text](#), [Draw To](#)

Draw CopyTo

Syntax

Draw CopyTo [*destination*], [*file*]

Synonyms

DR CT

Description

The Draw CopyTo command copies the current drawing (set with the Draw To command) to the clipboard (as a Windows MetaFile), Printer or File.

Parameters

The following table describes the parameters of the Draw CopyTo command:

Parameter	Description
<i>destination</i>	Destination for the copy:- Clipboard, Printer or File. Default Clipboard
<i>file</i>	Filename for the copy if the destination is File

Example

Copy a piechart to the printer

```
* The following is used in example\script\PieChart.wis to
* copy the pie chart in the dialog box to the printer
Draw To PieChart.Graph
Draw CopyTo Printer
Draw To Screen
```

Related Script Commands

[Draw To](#)

Version

4.0.3 Original

Draw Ellipse

Syntax

Draw Ellipse [*x1*], [*y1*], *x2*, *y2*

Synonyms

DR EL

Description

The Draw Ellipse command draws an ellipse within a rectangle with corners specified at *x1*, *y1*, and *x2*, *y2*. The ellipse is drawn with the current pen color and style you set with the Draw Pen command, and the brush color and style you set with the Draw Brush command. If you omit the *x1* and *y1* coordinates, the starting position of the ellipse is the current cursor position.

Parameters

The following table describes the parameters of the Draw Ellipse 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

The following example demonstrates the Draw Ellipse command:

```
Draw Move 700,60
Draw To Screen
.
.
.
* Set the border (Pen) and fill (Brush) for the ellipse
Draw Pen RGB_Black, PEN_Dot
Draw Brush RGB_Blue, BRUSH_FDiagonal
Draw Ellipse 330,110,410,190
```

Related Script Commands

[Draw Arc](#), [Draw Brush](#), [Draw Chord](#), [Draw Erase](#), [Draw Font](#), [Draw Image](#), [Draw Line](#), [Draw Move](#), [Draw Pen](#), [Draw Pie](#), [Draw Polygon](#), [Draw Rect](#), [Draw Size](#), [Draw Text](#), [Draw To](#)

Draw Erase

Syntax

Draw Erase *x1, y1, x2, [y2], flags*

Synonyms

DR ER

Description

The Draw Erase command erases objects drawn by any of the draw commands.

Parameters

The following table describes the parameters of the Draw Erase 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>flags</i>	See the following Flags table.

Values for flags

The flags define the objects or shapes to be erased. Use one of the first three flags and any of the other flags together (separated by a vertical bar "|"), to get the effect you want.

Value	Description
<i>ERASE_All</i>	Delete all objects.
<i>ERASE_First</i>	Delete only the first object drawn.
<i>ERASE_Last</i>	Delete only the last object drawn.
<i>ERASE_Point</i>	Erase the object that contains the point x1, y1.

Value	Description
<i>ERASE_PointInBound</i>	The point x1, y1 is within the object's bounding rectangle.
<i>ERASE_Area</i>	The rectangular area x1, y1 to x2, y2 overlaps the object's bounding rectangle.
<i>ERASE_AreaInside</i>	The rectangular area x1, y1 to x2, y2 completely covers the object's bounding rectangle.
<i>ERASE_AreaPoint</i>	The rectangular area x1, y1 to x2, y2 encloses one of the points of the object.

Example

The following example shows how to erase all drawings from the screen:

```
* Erase all drawing from the screen
Draw Erase 0, 0, 799, 239, ERASE_All | ERASE_Area
```

In the next example, the first object draw is erased that is in the bottom part of the screen:

```
* Erase first object drawn which is in bottom part of screen
Draw ERase 0, 120, 799, 239, ERASE_First | ERASE_AreaInside
```

In the next example, a rectangle is drawn and then erased:

```
* Draw a rectangle
Draw Rect 10, 20, 199, 153
.
.
.
* Erase the rectangle we just created
Draw Erase 199, 153, ERASE_Last | ERASE_Point
```

Related Script Commands

[Draw Arc](#), [Draw Brush](#), [Draw Chord](#), [Draw Ellipse](#), [Draw Font](#), [Draw Image](#), [Draw Line](#), [Draw Move](#), [Draw Pen](#), [Draw Pie](#), [Draw Polygon](#), [Draw Rect](#), [Draw Size](#), [Draw Text](#), [Draw To](#)

Draw Font

Syntax

Draw Font *fontname, width, height, [style]*

Synonyms

DR FN

Description

The Draw Font command specifies the font to use for the Draw Text command.

Parameters

The following table describes the parameters of the Draw Font command:

Parameter	Description
<i>fontname</i>	The name of the Windows font to use.
<i>width</i>	The width of the font, specified in graphic units. The default width and height are 10, 10. To show a font twice as large, specify width and height as 20, 20.
<i>height</i>	The height of the font, specified in graphic units. The default width and height are 10, 10. To show a font twice as large, specify width and height as 20, 20.
<i>style</i>	Specifies the style of font to use. See the following Styles table.

Values for style

You can use one of the first four, then or more of the remaining styles, separated by a vertical bar "|".

Value	Description
<i>FONT_Light</i>	Sets a light-weight font.
<i>FONT_Normal</i>	Sets a normal-weight font.

Value	Description
<i>FONT_Bold</i>	Sets a bold font.
<i>FONT_Heavy</i>	Sets a heavy-weight (bold) font.
<i>FONT_Italic</i>	Sets an italic font.
<i>FONT_Underline</i>	Underlines the text.
<i>FONT_StrikeOut</i>	Places a line through the text.

Example

The following example is taken from the PIECHART.WIS demonstration program.

```
PercentTable
```

```
char_size = 20
Draw Font "IBSFont",char_size,char_size * 2
tx = 800 - (Parse_LabelLen + 1) * char_size
ty = 20
xl = char_size * 4
xr = tx - char_size * 4
yt = char_size * 3
yb = 800 - char_size * 2
```

Related Script Commands

[Draw Arc](#), [Draw Brush](#), [Draw Chord](#), [Draw Ellipse](#), [Draw Erase](#), [Draw Image](#), [Draw Line](#), [Draw Move](#), [Draw Pen](#), [Draw Pie](#), [Draw Polygon](#), [Draw Rect](#), [Draw Size](#), [Draw Text](#), [Draw To](#)

Draw Image

Syntax

Draw Image [*x1*], [*y1*], *x2*, *y2*, *image*, [*display_option*]

Synonyms

DR IM

Description

The Draw Image command inserts an image in a rectangle specified by the coordinates. If *x1* and *y1* are omitted, then the top left of the rectangle is the current graphic cursor position.

Parameters

The following table describes the parameters of the Draw Image command:

Parameter	Description
<i>x1</i>	The x position of the start point.
<i>y1</i>	The y position of the top left of the start point.
<i>x2</i>	The x position of the bottom right of the image.
<i>y2</i>	The y position of the bottom right of the image.
<i>image</i>	The full path of the image file (unless file is in the wintegrate directory).
<i>display_option</i>	Specifies how to display the image in its bounding rectangle. Defaults to 0 to scale the image.

Values for display_option

Value	Description
0	Scale the image. This is the default.
1	Clip the image
2	Tile the image

Example

The following example draws the "image.bmp" image on the screen. Since the image file is located in the wIntegrate directory, the full path name is not necessary.

```
Draw Image 400, 20, 700, 200, "image.bmp"
```

Related Script Commands

[Draw Arc](#), [Draw Brush](#), [Draw Chord](#), [Draw Ellipse](#), [Draw Erase](#), [Draw Font](#), [Draw Line](#), [Draw Move](#), [Draw Pen](#), [Draw Pie](#), [Draw Polygon](#), [Draw Rect](#), [Draw Size](#), [Draw Text](#), [Draw To](#)

Version

5.0 Added tile parameter

Draw Line

Syntax

Draw Line [*x1*], [*y1*], *x2*, *y2*

Synonyms

DR LN

Description

The Draw Line command draws a line from *x1*, *y1* to *x2*, *y2*. If *x1* and *y1* are omitted, then the top left of the rectangle is the current graphic cursor position. The line is drawn with the current pen color and style you set with the Draw Pen command.

Parameters

The following table describes the parameters of the Draw Line command:

Parameter	Description
<i>x1</i>	The x position of the start point.
<i>y1</i>	The y position of the top left of the start point.
<i>x2</i>	The x position of the bottom right of the start point.
<i>y2</i>	The y position of the bottom right of the start point.

Example

The following example is part of the BARCHART.WIS demonstration program.

```
Draw Pen RGB_Black
Draw Brush 0,BRUSH_Null
Draw Line x1,yt,xl,yb
Draw Line xr,yb
```

Related Script Commands

[Draw Arc](#), [Draw Brush](#), [Draw Chord](#), [Draw Ellipse](#), [Draw Erase](#), [Draw Font](#), [Draw Image](#), [Draw Move](#), [Draw Pen](#), [Draw Pie](#), [Draw Polygon](#), [Draw Rect](#), [Draw Size](#), [Draw Text](#), [Draw To](#)

Draw Move

Syntax

Draw Move *x*, *y*

Synonyms

DR MV

Description

The Draw Move command moves the current cursor position without drawing.

Parameters

The following table describes the parameters of the Draw Move command:

Parameter	Description
<i>x</i>	The column position for the cursor.
<i>y</i>	The row position for the cursor.

Example

The following example shows the Draw Move command:

```
* Start the drawing at the bottom half of the screen
Draw To Screen
Draw Move 700,60
```

Related Script Commands

[Draw Arc](#), [Draw Brush](#), [Draw Chord](#), [Draw Ellipse](#), [Draw Erase](#), [Draw Font](#), [Draw Image](#), [Draw Line](#), [Draw Pen](#), [Draw Pie](#), [Draw Polygon](#), [Draw Rect](#), [Draw Size](#), [Draw Text](#), [Draw To](#)

Draw Pen

Syntax

Draw Pen *color*, [*style*], [*width*]

Synonyms

DR PN

Description

The Draw Pen command sets the color, style, and thickness of the drawing pen. The pen is used by several other Draw commands. You can change the Pen and Brush colors and styles for different components of a drawing. See the "Draw Module" example in "Appendix A - Scripting Libraries and Modules" for an example of different ways to use Draw Pen.

Parameters

The following table describes the parameters of the Draw Pen command:

Parameter	Description
<i>color</i>	Specifies the color of the drawing pen. Use the RGB command or one of the following constants: RGB_Black, RGB_Blue, RGB_Green, RGB_Cyan, RGB_Red, RGB_Magenta, RGB_Brown, RGB_Gray, RGB_Grey, RGB_LightGray, RGB_LightGrey, RGB_LightBlue, RGB_LightGreen, RGB_LightCyan, RGB_LightRed, RGB_LightMagenta, RGB_Yellow, RGB_White
<i>style</i>	Specifies the style of the drawing pen. See the following Style table.
<i>width</i>	Sets the thickness of the pen. This parameter is available only when you use the "Solid" argument for style. This can be a number from 1 through 100. Default is 1.

Values for style

Value	Description
<i>PEN_Solid</i>	Specifies a solid color (the default).
<i>PEN_Dash</i>	Specifies a dashed line. Specify the color of the area out-side the dashes with the Draw Brush command.
<i>PEN_Dot</i>	Specifies a dotted line. Specify the color of the area out-side the dots with Draw Brush.
<i>PEN_DashDot</i>	Specifies a line of alternating dashes and dots. Specify the color of the area outside the dashes/dots with Draw Brush.
<i>PEN_DashDotDot</i>	Specifies a line of dashes, each separated by two dots. Specify the color of the area outside the dashes/dots with Draw Brush.
<i>PEN_Null</i>	Specifies no color and ignores the COL argument.
<i>PEN_InsideFrame</i>	Specifies a line drawn inside the edges of a polygon.

Example

The following example is taken from the PIECHART.WIS demonstration program.

```
* Labels
Draw Rect tx,ty,tx+char_size,ty+char_size * 2
Draw Pen RGB_Black
Draw Brush 0,BRUSH_Null
Draw Text tx + char_size+10, ty, label
ty += char_size * 3
```

Related Script Commands

[Draw Arc](#), [Draw Brush](#), [Draw Chord](#), [Draw Ellipse](#), [Draw Erase](#), [Draw Font](#), [Draw Image](#), [Draw Line](#), [Draw Move](#), [Draw Pie](#), [Draw Polygon](#), [Draw Rect](#), [Draw Size](#), [Draw Text](#), [Draw To](#)

Draw Pie

Syntax

Draw Pie [*x1*], [*y1*], *x2*, *y2*, *xs*, *ys*, *xe*, *ye*

Synonyms

DR PI

Description

The Draw Pie command draws a pie segment of an ellipse bounded by the rectangle *x1*, *y1* to *x2*, *y2*.

Parameters

The following table describes the parameters of the Draw Pie 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>xs</i>	The column position of the start of the arc angle.
<i>ys</i>	The row position of the start of the arc angle.
<i>xe</i>	The column position of the end of the arc angle.
<i>ye</i>	The row position of the end of the arc angle.

Example

The following example is part of the PIECHART.WIS demonstration program.

```
* Pie segment
If pc Then
x1 = cospc(cpc)
y1 = -sinpc(cpc)
Draw Pie x1,yt,xr,yb, xc + x0, yc + y0, xc + x1, yc + y1
x0 = x1
y0 = y1
```


EndIf

The next example shows a more simplified way to use this command.

```
* Set the border (Pen) and fill (Brush) for the pie segment
Draw Pen RGB_Gray
Draw Brush RGB_Cyan
Draw Pie 480,110,580,190,240,115,120,230
```

Related Script Commands

[Draw Arc](#), [Draw Brush](#), [Draw Chord](#), [Draw Ellipse](#), [Draw Erase](#), [Draw Font](#), [Draw Image](#), [Draw Line](#), [Draw Move](#), [Draw Pen](#), [Draw Polygon](#), [Draw Rect](#), [Draw Size](#), [Draw Text](#), [Draw To](#)

Draw Polygon

Syntax

Draw Polygon *x1, y1, ..., xn, yn*

Synonyms

DR PL

Description

The Draw Polygon command draws a polygon with up to 5 vertices.

Parameters

The following table describes the parameters of the Draw Polygon command:

Parameter	Description
<i>x1</i>	The x position of the top left of the polygon.
<i>y1</i>	The y position of the top left of the polygon.
...	Position of vertices 2 up to n
<i>xn</i>	The x position of one of a number of vertices.
<i>yn</i>	The y position of one of a number of vertices.

Example

The following example shows the Draw Polygon command:

```
* Start the drawing at the bottom half of the screen
Draw Move 700,60
Draw To Screen
* Set the border (Pen) and fill (Brush) for the polygon
Draw Pen RGB_White, PEN_Solid, 4
Draw Brush RGB_LightMagenta
Draw Polygon 450,110,415,110,450,180
```

Related Script Commands

[Draw Arc](#), [Draw Brush](#), [Draw Chord](#), [Draw Ellipse](#), [Draw Erase](#), [Draw Font](#), [Draw Image](#), [Draw Line](#), [Draw Move](#), [Draw Pen](#), [Draw Pie](#), [Draw Rect](#), [Draw Size](#), [Draw Text](#), [Draw To](#)

Draw Rect

Syntax

Draw Rect [*x1*], [*y1*], *x2*, *y2*

Synonyms

DR RC

Description

The Draw Rect command draws a rectangle within specified coordinates. If the first two coordinates are omitted, wIntegrate starts the drawing at the current cursor position.

Parameters

The following table describes the parameters of the Draw Rect 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

The following example is part of the PIECHART.WIS demonstration program.

```
* Labels
Draw Rect tx,ty,tx+char_size,ty+char_size * 2
Draw Pen RGB_Black
Draw Brush 0,BRUSH_Null
Draw Text tx + char_size+10, ty, label
ty += char_size * 3
```

Related Script Commands

[Draw Arc](#), [Draw Brush](#), [Draw Chord](#), [Draw Ellipse](#), [Draw Erase](#), [Draw Font](#), [Draw Image](#), [Draw Line](#), [Draw Move](#), [Draw Pen](#), [Draw Pie](#), [Draw Polygon](#), [Draw Size](#), [Draw Text](#), [Draw To](#)

Draw Size

Syntax

Draw Size *width, depth*

Synonyms

DR SZ

Description

The Draw Size command sets the logical size of a drawing. Draw Size is reset if the lines, columns, or back pages are changed in Setup Terminal.

The default logical size for the screen is ten times the number of columns and lines for the terminal screen. The default logical size for a draw control is 1000 X 1000.

Parameters

The following table describes the parameters of the Draw Size command:

Parameter	Description
<i>width</i>	The new logical width of the screen.
<i>depth</i>	The new logical depth or height of the screen.

Example

The following example sizes the drawing to the size of the 80 X 24 terminal screen.

```
Draw Size 800, 240
```

Related Script Commands

[Draw Arc](#), [Draw Brush](#), [Draw Chord](#), [Draw Ellipse](#), [Draw Erase](#), [Draw Font](#), [Draw Image](#), [Draw Line](#), [Draw Move](#), [Draw Pen](#), [Draw Pie](#), [Draw Polygon](#), [Draw Rect](#), [Draw Text](#), [Draw To](#)

Draw Text

Syntax

Draw Text [*x*], [*y*], *text*

Synonyms

DR TX

Description

The Draw Text command draws text on the screen.

Tip: The Draw Font command specifies the font to use for the Draw Text command.

Parameters

The following table describes the parameters of the Draw Text command:

Parameter	Description
<i>x</i>	The column position for the start of the text.
<i>y</i>	The row position for the start of the text.
<i>text</i>	The text to draw on the wIntegrate screen.

Example

In the first example, a simple program shows how to use this command.

```
* Program to demonstrate wIntegrate Draw commands
* Start the drawing at the bottom half of the screen
Draw Move 700,60
Draw To Screen
* Draw a rectangle
Draw Rect 30, 100, 600, 200
* Set the font and pen color for the text
Draw Font "Times_Roman", 20,20,Font_Heavy|Font_Italic
Draw Pen RGB_None
Draw Text 75,140, "wIntegrate"
```

The next example is taken from the PIECHART.WIS demonstration program.

```
* Labels
Draw Rect tx,ty,tx+char_size,ty+char_size * 2
Draw Pen RGB_Black
Draw Brush 0,BRUSH_Null
Draw Text tx + char_size+10, ty, label
ty += char_size * 3
```

Related Script Commands

[Draw Arc](#), [Draw Brush](#), [Draw Chord](#), [Draw Ellipse](#), [Draw Erase](#), [Draw Font](#), [Draw Image](#), [Draw Line](#), [Draw Move](#), [Draw Pen](#), [Draw Pie](#), [Draw Polygon](#), [Draw Rect](#), [Draw Size](#), [Draw To](#)

Draw To

Syntax

Draw To {screen / control}

Synonyms

DR TO

Description

This function directs drawing to the wIntegrate terminal screen or to a wIntegrate draw control.

Parameters

The following table describes the parameters of the Draw To command:

Parameter	Description
<i>{screen / control}</i>	Must be the literal "screen", which directs drawing to the terminal screen, or the name of a draw control.

Example

The following example is part of the BARCHART.WIS demonstration program.

```
drawctrl = dlname:".Graph"  
Draw To $drawctrl  
char_size = 20  
Draw Font "IBSFont",char_size,char_size * 2
```

Reset the drawing back to the screen

```
Draw To Screen
```

Related Script Commands

[Draw Arc](#), [Draw Brush](#), [Draw Chord](#), [Draw Ellipse](#), [Draw Erase](#), [Draw Font](#), [Draw Image](#), [Draw Line](#), [Draw Move](#), [Draw Pen](#), [Draw Pie](#), [Draw Polygon](#), [Draw Rect](#), [Draw Size](#), [Draw Text](#)

EditInput

Syntax

EditInput *x, y, var, length, [exit_var], [no_lines], [display_lines], [opts], [valid], [exit_keys]*

Synonyms

EI

Description

The EditInput command enables you to edit a variable on the host screen.

If the session global variable EditInputKeys is True, then the Soft Edit keys return special values when EditInput is running.

The host subroutine WIN.EI uses this command to allow screen editing of a host input field.

You may finish the input in one of these ways:

Entering a carriage return - the default method.

Entering the Esc key.

Entering this combination: Ctrl-F Shift-End.

Defining an exit key in the exit_keys parameter.

Parameters

The following table describes the parameters of the EditInput command:

Parameter	Description
<i>x</i>	The column position for the input.
<i>y</i>	The row position for the input.
<i>var</i>	The variable to be edited
<i>length</i>	The length of the input.

Parameter	Description
<i>exit_var</i>	Specifies the characters to exit the routine. Set to "" for a carriage return exit.
<i>no_lines</i>	The number of lines of the input. The default is 1.
<i>display_lines</i>	The number of lines to display at one time.
<i>opts</i>	See the following Options below.
<i>valid</i>	See the following Validations table.
<i>exit_keys</i>	A string (in backslash format) that ends input.

Values for opts

Options are used in conjunction with the validation parameter. The options for the EditInput command 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 uppercase.
<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>	Pads the background with the specified character.
<i>I</i>	Starts the editing in insert mode (rather than default mode)
<i>R</i>	Read only. The text displayed can not be altered

Value	Description
<i>K</i>	Force use of EditInputKeys regardless of the global variable setting

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.
<i>"x"</i>	Any character inside quotes. To designate a choice of characters or text, separated the choices by a vertical bar (). Example: "'EXIT' 'QUIT'".
<i>"text"</i>	Any text to validate. To designate a choice of characters or text, separated the choices by a vertical bar ().

Example

The following line enables input of a number from -999.99 to 999.99, and exits when the character "*" is entered.

```
EditInput 5,10, Value, 8, exit_char, , , "MBS", "0-1'-'1-3N'.'2N", "*"

```

Version

4.1.1 I, R and K options added

EditInput2

Syntax

EditInput2 *var*, *params*, [*exit_var*], [*state_var*]

Synonyms

E2

Description

The EditInput2 command enables you to edit a wIntegrate variable on the host screen.

If the session global variable EditInputKeys is True, then the Soft Edit keys return special values when EditInput 2 is running.

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 "Escape" 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 parameters.

Pressing a key defined in the exit_keys field of the parameters

The EditInput2 command is an enhanced version of the EditInput command.

Parameters

The following table describes the parameters of the EditInput2 command:

Parameter	Description
<i>var</i>	The variable to edit
<i>params</i>	The parameters that control the edit session. See below

Parameter	Description
<i>exit_var</i>	A variable that receives the character that exited the routine.
<i>state_var</i>	A variable which sets/gets the position and state of the edit session. See below

Values for params

This parameter is made up of the following cr separated fields.

Value	Description
<i>1</i>	The column position for the input
<i>2</i>	The row position for the input
<i>3</i>	The length of the input line
<i>4</i>	The number of lines. Default 1
<i>5</i>	The number of columns to display. Default is the length of the input line (field 3)
<i>6</i>	The number of lines to display. Default is the number of line (field 4)
<i>7</i>	Options. See options table below
<i>8</i>	Validation. See validation table below
<i>9</i>	Exit chars. The characters that will cause the input to exit. E.g. "※?"
<i>10</i>	Exit keys. Tab 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 dialog box. E.g. "Escape":tab:"F1"

Options

Options are used in conjunction with the validation parameter. The options for the

EditInput command can be one or more of the option letters, concatenated and inside of quotes, e.g., "AM":

Option	Description
M	Makes the validation parameter mandatory
U	Converts input into uppercase.
B	Beeps when an error occurs.
D	Shows the error on a dialog box
S	Displays the error on the status bar.
A	Causes an automatic carriage return. The field is automatically finished when the last character is typed.
V	Validates on exit by exit_chars.
P(char)	Pads the background with the specified character.
I	Starts the editing in insert mode (rather than default mode)
R	Read only. The text displayed can not be altered
K	Force use of EditInputKeys regardless of the global variable setting

Validations

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)).

Pattern	Description
N	A numeric character.
A	An alpha character.
X	Any character

Pattern	Description
"text"	Any text to validate. To designate a choice of characters or text, separated the choices by a vertical bar (). Example: "'EXIT' "QUIT'".

Values for state_var

The state variable is used to set the start up position and start of the edit session. If it is omitted or its contents are empty the edit session will start up at the beginning of the edit field in overwrite mode. When an edit session finishes the state variable will be updated to reflect the state of the edit session when it was exited.

Value	Description
1	State of the input. Zero or more of the following characters: I - Insert mode M - modified. This value is reported at the end of an edit session and has no effect when starting an edit session
2	Edit col - the column position of the cursor in the editor
3	Edit row - the row position of the cursor in the editor
4	Display col - the column that is displayed at the left hand side of the edit area
5	Display row - the row that is displayed at the left hand side of the edit area

Version

4.1.1 Original

EditInputInfo

Syntax

EditInputInfo *Options*

Description

This function returns information on the currently running EditInput or EditInput2 script command. For all options other than option 1, "" is returned if there is no edit input session.

Parameters

The following table describes the parameters of the EditInputInfo command:

Parameter	Description
<i>Options</i>	See below

Values for Options

Value	Description
<i>1</i>	Returns 1 if an edit session is running, otherwise 0.
<i>2</i>	Status. This is in the same format as the status variable from the EditInput2 command
<i>3</i>	The current text of the edit control
<i>4</i>	The text from the current line of the edit control

Return Value

The information on the EditInput state specified by the options parameter.

Version

4.1.1 Original

Effect

Syntax

Effect *option*, *x*, *y*, [*page*]

Description

The Effect function returns a text string specifying the screen effect.

Parameters

The following table describes the parameters of the Effect command:

Parameter	Description
<i>option</i>	Specifies the information to return. Must be one of the following: 0 - Current effect, 1 - Effect at a specified position.
<i>x</i>	x coordinate for returning information.
<i>y</i>	y coordinate for returning information.
<i>page</i>	Back page number.

Return Value

returned as Effect_effectname or Color_foreground_color, background_color depending on the option you specify.

Example

In the following example, the program segment uses the Effect function to check the current effect and the current color:

```
* See if we are currently printing to the terminal in
* reverse
current_effect = Effect (0)
If Not(current_effect = "Effect_Reverse") Then
    Display Effect "Reverse"
EndIf

* Check for an error by the color it is drawn on the
* bottom of the screen
```

```
* Error colors are white (15) on red (1)
If Effect(1, 0, 23) = "Color_15,1" Then
    MessageBox ScreenText (0, 23, 79, 23)
EndIf
```

Else

Syntax

Else

Description

See the If command

Parameters

None

Related Script Commands

[If](#)

EndCreateIconBar

Syntax

EndCreateIconBar

Description

The EndCreateIconBar command stops the creation mode of icon bars.

The EndCreateIconBar command is a Icon Library command. For more information, see "Appendix A - Scripting Libraries and Modules".

Parameters

None

Related Script Commands

[CreateIconBar](#)

EndIf

Syntax

EndIf

Description

For information, see the If command.

Parameters

None

Related Script Commands

[If](#), [Else](#)

EndOfFile

Syntax

EndOfFile *Name*

Description

This function returns true if the file pointer is at the end of the opened file.

Parameters

The following table describes the parameters of the EndOfFile command:

Parameter	Description
<i>Name</i>	The identifier for this file, specified in File Open.

Return Value

The return value is 1 if the file pointer is at the end of the file or the file handle is invalid or 0 if the file pointer is before the end of the file.

Example

Read a file a byte at a time

```
File Open f_test, mytest, ok
If ok Then
  Loop
  Until EndOfFile(f_test)
  File Read f_test, next_char, 1
  * Process the character here
  ...
Repeat
File Close f_test
EndIf
```

Related Script Commands

[File Pointer](#)

Version

4.0.3 Original

EndScript

Syntax

EndScript

Synonyms

ES

Description

The EndScript command exits the current script. Note You must use this command before you include any user-defined functions or create subroutines.

Parameters

None

Example

The following example is taken from the DDE_SCR.WIS demonstration program.

```
* DDE_SCR
* Script to create DDE links to spreadsheet
Library "wintsys\lib\server"
DDE Initiate SpreadSheet, "excel", "sheet1", OK
If Not(OK) Then
  MessageBox "Excel not started, or SHEET1.XLS not open"
EndScript
EndIf
```

EndSub

Syntax

EndSub

Synonyms

EB

Description

The EndSub command exits the current user-defined subroutine.

Parameters

None

Example

The following example is taken from the DISPBOX.WIS demonstration program.

```
* Display the box at correct position in required style
Sub DispBox(x, y, Style)
Display Box x, y, x+BoxWidth, y+BoxDepth, Style ;* Outer border
Display Box x+BoxWidth/2, y, x+BoxWidth/2, y+BoxDepth, Style
;* Vertical line
Display Box x, y+BoxDepth/2, x+BoxWidth, y+BoxDepth/2, Style
;* Horizontal line
EndSub
```

Related Script Commands

[Sub](#)

EndWith

Syntax

EndWith

Synonyms

EW

Description

Ends a With block. See the With command for details.

Parameters

None

Related Script Commands

[With](#)

Version

4.0.2 Original

Enter

Syntax

Enter *string*, [*translate*]

Synonyms

ET

Description

The Enter command sends text followed by a carriage return to the host. Type is a similar command, but does not add a carriage return to the end of the text string.

Parameters

The following table describes the parameters of the Enter command:

Parameter	Description
<i>string</i>	The text to send to the host.
<i>translate</i>	Translates the string from backslash format. The default is True.

Example

The following example sends the string to the host. Since the Enter command sends a carriage return, the SORT is executed immediately.

```
Enter "SORT CUSTOMER NAME BY NAME"
```

Related Script Commands

Type

Event Delete

Syntax

Event Delete *type*, [*text*]

Synonyms

EV DL

Description

The Event Delete command deletes a previously defined event.

Parameters

The following table describes the parameters of the Event Delete command:

Parameter	Description
<i>type</i>	The event type. This can be any of the events created with the Event command and is specified as their name or abbreviaiton: • OnHostText or HT • OnExit or OE • OnRestore or OR • OnPortClose or OP • OnPortOpen or OO (wIntegrate 98 only)
<i>text</i>	Text to match. (For type = OnHostText only.)

Example

In the following example, the Event Delete command is used to delete events from an Event OnHostText command:

```
Event Delete OnHostText "alan"
Event Delete OnHostText ":",
```

Event OnExit

Syntax

Event OnExit *command*

Synonyms

EV OE

Description

The Event OnExit command sets up another command to run whenever the current session ends.

When an exit command is defined, the script can be exited only with Invoke FileExit or by deleting this event with the Event Delete command. This command prevents the normal Exit dialog from being shown.

Tip: You can determine whether an OnExit event has been defined by using the IsEvent function.

Parameters

The following table describes the parameters of the Event OnExit command:

Parameter	Description
<i>command</i>	Script command to run

Example

The following example shows the Event OnExit command:

```
* Run the script stopexit.wis when an attempt is made to exit the
program
* The stopexit.wis example script is provided with wIntegrate.
* Running it once from Run Script will set up the event.
Event OnExit 'Script "example\script\stopexit"'
```

In the next example, the Event OnExit command is used to check if a user is logged off:

```
Event OnExit 'reply = MessageBox("Are you logged  
off?","Exit",MB_YESNO);If reply = "Yes" Then Invoke FileExit'
```

Event OnHostText

Syntax

Event OnHostText *text*, *command*

Synonyms

EV HT

Description

The Event OnHostText command specifies a script expression to be run whenever the specified text is received from the host.

Parameters

The following table describes the parameters of the Event OnHostText command:

Parameter	Description
<i>text</i>	Text to look for. text will match only text that is displayed on the screen, it will not match text which is part of an emulation escape sequence. The low byte of each character in the text is used to match the bytes received from the host.
<i>command</i>	Expression to run.

Example

In the following example, the program segment counts the number of colons received from the host.

```
* Count the number of colons which come to the screen
Global cc = 1
Event OnHostText ":", "cc+=1; Print cc:' colons received;"
```

In the next example, the program segment highlights the text "alan" whenever it is received from the host:

```
* Highlight the text alan whenever it appears
Event OnHostText "alan", "dp effectfill
Cursor(Get_X)-4, Cursor(Get_Y), Cursor(get_x)-1, Cursor(Get_y), 0,
```

'Reverse' "

Event OnPortClose

Syntax

Event OnPortClose *command*

Synonyms

EV OP

Description

The Event OnPortClose command sets up another command to run whenever the connected port closes.

Tip: You can determine if an OnPortClose event has been defined by using the IsEvent function.

Parameters

The following table describes the parameters of the Event OnPortClose command:

Parameter	Description
<i>command</i>	The script command to run

Example

The following example shows the Event OnPortClose command:

```
* Turn off check box on MyDialog when connection closes
Event OnPortClose "MyDialog.Connected = False"
```

Event OnPortOpen

Syntax

Event OnPortOpen *command*

Synonyms

EV OO

Description

The Event OnPortOpen command sets up another command to run whenever the a connection port opens.

Tip: You can determine if an OnPortOpen event has been defined by using the IsEvent function.

Parameters

The following table describes the parameters of the Event OnPortOpen command:

Parameter	Description
<i>command</i>	The script command to run

Example

The following example shows the Event OnPortOpen command:

```
* Turn off check box on MyDialog when connection opens
Event OnPortOpen "MyDialog.Connected = True"
```

Event OnRestore

Syntax

Event OnRestore *command*

Synonyms

EV OR

Description

The Event OnRestore command sets up another command to run whenever the user attempts to restore or maximize a minimized session.

When a restore command has been defined it is not possible to maximize wIntegrate until the restore command is deleted with the Event Delete command.

Tip: You can determine if an OnRestore event has been defined by using the IsEvent function.

Parameters

The following table describes the parameters of the Event OnRestore command:

Parameter	Description
<i>command</i>	The script command to run

Example

The following example shows the Event OnRestore command:

```
* Run the script stoprest.wis when an attempt is made to restore the
program.
* The stoprest.wis example script is provided with wIntegrate.
* Running it once from Run Script will set up the event.
Event OnRestore 'Script "example\script\stoprest.wis"'
```

ExecAccess

Syntax

ExecAccess

Description

The ExecAccess command runs the ECL commands sent to the server after a StoreAccess command.

This is kept for backward compatability, use the MultiExec library command for new scripts.

The ExecAccess command is a Server Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

None

Related Script Commands

[MultiExec](#), [StoreAccess](#)

Version

4.0 Original

Execute

Syntax

Execute *text*, *C*

Synonyms

EX

Description

The Execute command runs the expression (*text*) as if it were a script.

If *C* is specified the *script_text* is run as if it was a script file that had been run by the Chain command.

The Execute command can also run compiled scripts if the *script_text* passed to it contains the byte code of a compiled script.

Parameters

The following table describes the parameters of the Execute command:

Parameter	Description
<i>text</i>	The script command to execute.
<i>C</i>	Literal 'C'. Optional parameter to stop the original script when the execute is run

Example

The following example is part of the WC.WIS demonstration program.

```
* Run the command and copy it to list part of combo box
Sub RunCmnd()
pos = Locate(DialogList(wc,Cmnd), wc.Cmnd, "EU")
If pos = 0 Then DialogBox AddToList wc,cmnd,wc.cmnd,1
Execute wc.Cmnd
If IsShown(wc) Then DialogBox SetText wc,Cmnd, ''
EndSub
```

Related Script Commands

[Chain](#), [Script](#)

Version

4.0.1 C option added

4.2.1 Ability to execute compiled scripts

Extract

Syntax

Extract *string*, *line*, [*tab*]

Description

The Extract function returns a substring of a string at a specified position.

Parameters

The following table describes the parameters of the Extract command:

Parameter	Description
<i>string</i>	The string to extract data.
<i>line</i>	The line number in the string to extract the sub string from.
<i>tab</i>	The tab number in the line.

Return Value

The sub_string string at the specified position

Example

The following example is part of the DIRLIST.WIS demonstration program, which extracts the file names in the winteg directory and writes them to a file.

```
StartDir = AppDir ;* Default to main WINTEG directory
StartLen = Length(StartDir)

OutputList = "List of directories and files in " : StartDir : CR : LF
: CR : LF

Directories = StartDir

Loop
CurrDir = Extract(Directories, 1)
Del Directories,1
While CurrDir
OutDir = Mid(CurrDir, StartLen+1);* Strip off start directory
```

```
OutDir = Left(OutDir, Length(OutDir)-1 ;* Strip off trailing backslash
OutputList := "Directory " : OutDir : CR : LF
FList = FileList(CurrDir:"*.*", 3)
Max = Count(FList, CR) + 1
J = 0
```

Related Script Commands

[Del](#), [Ins](#), [Rep](#)

Field

Syntax

Field *string, delimiter, occurrence, [count]*

Description

The Field function returns a substring delimited by a specified character.

Parameters

The following table describes the parameters of the Field command:

Parameter	Description
<i>string</i>	The text or string to check.
<i>delimiter</i>	The single character delimiter between fields.
<i>occurrence</i>	The field number to extract.
<i>count</i>	The number of fields to extract. Defaults to 1.

Return Value

The occurrence of the text between the delimiters.

Example

The following example is a part of the PARSE.WIS demonstration script.

```
* Parse columns from selected area of screen
MessageBox "No section of screen selected"
EndScript
EndIf
dum = SelectInfo()
x1 = Field(dum, ",", 1)
y1 = Field(dum, ",", 2)
x2 = Field(dum, ",", 3)
y2 = Field(dum, ",", 4)
type = Field(dum, ",", 5) + 2; * Makes type into table type
b1 = Field(dum, ",", 6)
b2 = Field(dum, ",", 7)
text = ScreenText(x1,y1,x2,y2,type,b1,b2)
Convert text, lf, "" ; * Remove linefeeds from returned text
```

Get the folder a file is in

```
delim_count = Count(file_name, "\")  
folder = Field(file_name, "\", 1, delim_count)
```

Related Script Commands

[Index](#), [InString](#), [Locate](#)

Version

5.0 count parameter added

Fields

Syntax

Fields *list_to, filename*

Description

The Fields command sends a list of the fields in a selected host file to a list box or combo box.

The Fields command is a Server Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

The following table describes the parameters of the Fields command:

Parameter	Description
<i>list_to</i>	This parameter is made up of the dialog box name and the list box name, separated by a period.
<i>filename</i>	The name of the host file to list.

Example

In the following example, the partial script shows a list box within a dialog box, and how the Fields command uses the dialog and the list box name.

```
LText "Fields",None,87,2,40,12
ListBox FIELDLIST,73,15,55,73,WS_BORDER|WS_VSCROLL
LText "Items",None,141,2,40,12
ListBox ITEMLIST,139,15,49,71
Fields Cust.FIELDLIST, "CUSTOMER"
If Server_Error Then MsgBox "Fields" : ServerErrMsg(Server_Error)
```

File Close

Syntax

File Close *name*

Synonyms

FL CL

Description

The File Close command closes a file opened by File Open.

Parameters

The following table describes the parameters of the File Close command:

Parameter	Description
<i>name</i>	An identifier that you specified in File Open for this file. It is not the same as the file name.

Example

The following example is part of the FILEREAD.WIS demonstration program.

```
* Demonstrate some file commands
* Version 2.0
defaultfile = "wistest\fileread.wis"
filename = FileBrowse("Test of file info/read commands", defaultfile,
0,
"Scripts (*.wis)|*.wis|All (*.*)|*.*")
If filename # "" Then
msg = "Info: " : FileInfo(filename)
File Open f_test, filename
File Read f_test, r_test, 10
Ins msg, "Ascii: " : r_test
File Pointer f_test, 0
File Read f_test, r_test, 10, DF_BS
Ins msg, "Backslash: " : r_test
File Pointer f_test, 0
File Read f_test, r_test, 10, DF_HEX
File Pointer f_test, 0
File Read f_test, r_test, 10, DF_FT
Ins msg, "FT: " : r_test
```



```
MessageBox msg  
File Close f_test  
EndIf
```

Related Script Commands

[File Open](#), [File Pointer](#), [File Read](#), [File Write](#)

File Copy

Syntax

File Copy *source, dest, [overwrite], [err_var]*

Synonyms

FL CP

Description

The File Copy command copies a file to another file with a different path name. If a file with the destination name already exists, it is overwritten if the overwrite flag is set to True.

Parameters

The following table describes the parameters of the File Copy command:

Parameter	Description
<i>source</i>	Full path name of the file to be copied.
<i>dest</i>	Full path name of the destination for the copy to be pasted.
<i>overwrite</i>	True to overwrite the destination if it exists. The default is False.
<i>err_var</i>	If defined this variable will be set to 0 if the copy succeeds or 1 if the File Copy fails. If this is not defined a script error will be generated on failure.

Example

The following example copies the session called winteg1.wis to a file called winteg3.wis:

```
* Make a copy of session and call it winteg3.wis if it does not  
existFile Copy AppDir:"winteg1.wis",AppDir:"winteg3.wis"
```

The next example copies an excel file as a backup file.

```
* Note that the overwrite flag is set to True, so that the existing  
file is overwritten:  
* Overwrite old backup excel file  
File Copy "c:\info.xls", "c:\info.bak", True
```

Related Script Commands

[File Create](#), [File Move](#)

Version

5.0 err_var parameter added

File Create

Syntax

File Create *filename*, [*status_var*]

Synonyms

FL CR

Description

The File Create command creates a new local file.

Parameters

The following table describes the parameters of the File Create command:

Parameter	Description
<i>filename</i>	The full path name of the location of the new file.
<i>status_var</i>	A variable that receives a status code. See the following Status table for more information.

Values for status_var

Value	Description
0	File creation successful.
1	An unspecified error occurred.
2	The file could not be located.
3	All or part of the path is invalid.
4	The permitted number of open files was exceeded.
5	The file could not be accessed.
8	There are no more directory entries.
10	There was a hardware error.

Value	Description
11	File is used exclusively (locked) by another application.
13	The disk is full.

Example

The following example creates the file "pcnew.txt" if it does not already exist:

```
* Create the pcnew file if it does not exist
If FileExist("C:\ptest\pcnew.txt") Then
    MessageBox "ptest\pcnew.txt already exists"
Else
    File Create "C:\ptest\pcnew.txt"
EndIf
```

Related Script Commands

[DirExist](#), [File Copy](#), [File Delete](#), [FileExist](#), [FileInfo](#)

File CreateDir

Syntax

File CreateDir *directory*, [*status_var*], [*create_parents*]

Synonyms

FL CD

Description

The File CreateDir command creates a new local directory.

Parameters

The following table describes the parameters of the File CreateDir command:

Parameter	Description
<i>directory</i>	The full path name of the location of the new directory.
<i>status_var</i>	A variable that receives a status code. See the following Status table for more information
<i>create_parents</i>	Set to true to create any parent directory of the new directory that do not already exists. Set to false to give an error if the parent directory does not exists

Values for status_var

Value	Description
0	No error occurred
2	Invalid directory path
5	The directory already exists

Example

The following example creates the directory "C:\ptest" if it does not already exist:

```
* Create the pctest directory if it does not exist
If DirExist("C:\ptest") Then
  MsgBox "The directory pctest exists"
Else File CreateDir "C:\ptest"
EndIf
```

Related Script Commands

[DirExist](#), [File Create](#), [File Delete](#), [File DeleteDir](#), [FileExist](#)

Version

4.2.1 Added create_parents flag

File Delete

Syntax

File Delete *filename*, [*status_var*]

Synonyms

FL DL

Description

The File Delete command deletes a local file.

Parameters

The following table describes the parameters of the File Delete command:

Parameter	Description
<i>filename</i>	The full path name of the file to delete.
<i>status_var</i>	A variable that receives a status code. See the following Status table for more information.

Values for status_var

Value	Description
0	File deleted successfully
1	An unspecified error occurred.
2	The file could not be located.
3	All or part of the path is invalid.
4	The permitted number of open files was exceeded.
5	The file could not be accessed.
8	There are no more directory entries.
10	There was a hardware error.

Value	Description
11	File is used exclusively (locked) by another application.
13	The disk is full.

Example

The following example deletes the file "C:\pctest\pcnew.txt" if it exists:

```
* Delete the pcnew file if it exists
If FileExist("C:\pctest\pcnew.txt") Then
File Delete"C:\pctest\pcnew.txt"
MessageBox "pctest\pcnew.txt deleted"
EndIf
```

Related Script Commands

[File Create](#), [File CreateDir](#), [File DeleteDir](#), [FileExist](#), [FileInfo](#)

File DeleteDir

Syntax

File DeleteDir *directory*, [*status_var*]

Synonyms

FL DD

Description

The File DeleteDir command deletes a local directory.

Parameters

The following table describes the parameters of the File DeleteDir command:

Parameter	Description
<i>directory</i>	The full path name of the directory to delete.
<i>status_var</i>	A variable that receives a status code. See the following Status table for more information

Values for status_var

Value	Description
<i>0</i>	Directory deleted successfully
<i>1</i>	An unspecified error occurred.
<i>2</i>	The file could not be located.
<i>3</i>	All or part of the path is invalid.
<i>4</i>	The permitted number of open files was exceeded.
<i>5</i>	The file could not be accessed.
<i>8</i>	There are no more directory entries.
<i>10</i>	There was a hardware error.

Value	Description
11	File is used exclusively (locked) by another application.
13	The disk is full.

Example

In the following example, the "C:\ptest" directory is deleted if it exists:

```
* Delete the pctest directory if it exists
If DirExist("C:\ptest") Then
File DeleteDir "C:\ptest"
MessageBox "ptest directory deleted"
EndIf
```

Related Script Commands

[File Create](#), [File CreateDir](#), [File Delete](#), [FileExist](#), [FileInfo](#)

File Move

Syntax

File Move *source*, *new_name*, [*overwrite*], *err_var*

Synonyms

FL MV

Description

The File Move command moves or renames a file.

Parameters

The following table describes the parameters of the File Move command:

Parameter	Description
<i>source</i>	Full path name of the file to copy.
<i>new_name</i>	Full path name of the new name of the file.
<i>overwrite</i>	Specifies whether to overwrite a file that exists. Set to "True" to the destination file. The default is "False".
<i>err_var</i>	If defined this variable will be set to 0 if the File Move succeeds or 1 if the File Move fails. If this is not defined a script error will be generated on failure.

Example

In the following example, the File Move command is used to rename a file:

```
* Rename file
File Move AppDir:"custlet.doc",AppDir:"oldlet.doc"
```

In the next example, the program segment uses the wIntegrate directory as the destination directory.

The wIntegrate directory does not need to be specified, as it is always the default when a path is not specified. Remember that the

```
file name must still be in quotes.  
* Move file from root directory to wIntegrate directory  
* overwriting any file with the new name  
File Move "c:\info.xls", AppDir:"info.xls", True
```

Related Script Commands

[File Copy](#)

Version

5.0 err_var added

File Open

Syntax

File Open *name, filename, [ok_var], [status_var]*

Synonyms

FL OP

Description

The File Open command opens a local file. Use File Close to close the file when finished with other File operations.

Parameters

The following table describes the parameters of the File Open command:

Parameter	Description
<i>name</i>	An identifier you give to the file. You will use this identifier rather than the file name in other File commands.
<i>filename</i>	The actual name of the file. You must include the full path if the file is not in the wintegrate directory.
<i>ok_var</i>	A variable you specify which will contain one of the following status codes after the execution of the File Open command: 0 - File Open failed. 1 - File Open successful for read/write operations. 2 - File Open successful for read-only operations.
<i>status_var</i>	Reason code for why a file failed to open. See Table below

Values for status_var

Value	Description
0	No error occurred.

Value	Description
1	An unspecified error occurred.
2	The file could not be located.
3	All or part of the path is invalid.
4	The permitted number of open files was exceeded.
5	The file could not be accessed.
8	There are no more directory entries.
10	There was a hardware error.
11	File is used exclusively (locked) by another application.
13	The disk is full.

Example

The following example is taken from the FILEREAD.WIS demonstration program.

```
* Demonstrate some file commands
* Version 2.0

defaultfile = "wistest\fileread.wis"
filename = FileBrowse("Test of file info/read commands", defaultfile,
0, "Scripts (*.wis)|*.wis|All (*.*)|*.*")

If filename # "" Then
msg = "Info: " : FileInfo(filename)
File Open f_test, filename
File Read f_test, r_test, 10
Ins msg, "Ascii: " : r_test
```

Related Script Commands

[File Close](#), [File Pointer](#), [File Read](#), [File Write](#)

File Pointer

Syntax

File Pointer *name, position*

Synonyms

FL PN

Description

The File Pointer command moves the pointer a number of bytes into the file. 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.

Parameters

The following table describes the parameters of the File Pointer command:

Parameter	Description
<i>name</i>	The identifier for this file, specified in File Open.
<i>position</i>	The number of bytes into the file to position the pointer.

Example

The following example is taken from the FILEREAD.WIS demonstration program.

```
File Pointer f_test, 0
File Read f_test, r_test, 10, DF_BS
Ins msg, "Backslash: " : r_test
```

Related Script Commands

[File Close](#), [File Open](#), [File Read](#), [File Write](#)

File Read

Syntax

File Read *name, variable, [no_bytes], [options]*

Synonyms

FL RD

Description

The File Read command reads data from a file.

The Unicode character set has two bytes per character, so each byte of data read is placed in the low-byte of a character to make it backward compatible with previous releases.

Characters can be read in as Unicode using the DF_UTF8 or DF_UTF16 format if the file to be read is in the correct format.

Parameters

The following table describes the parameters of the File Read command:

Parameter	Description
<i>name</i>	The identifier for this file, specified in File Open.
<i>variable</i>	The variable to read the data into.
<i>no_bytes</i>	The number of bytes to read. The default reads the whole file or 64K, whichever is smaller. See the options parameter for DF_UTF8 and DF_UTF16 to see how they modify the actual number of bytes read.
<i>options</i>	Specifies different options for reading the data. See the following Options table for more information.

Values for options

Value	Description
<i>DF_BS</i>	Reads data as a backslash format string.
<i>DF_Hex</i>	Reads data as two-character Hex values.
<i>DF_FT</i>	Reads data in File Transfer format.
<i>DF_NoZero</i>	Strips the "zero" character from the data.
<i>DF_UTF8</i>	Reads the data from the file in UTF 8 (Unicode transformation format where each character is stored in 1 to 4 bytes). This option will move the pointer to the start of the next UTF-8 character in the input stream and will read extra bytes to ensure it always reads at least one full character from the file.
<i>DF_UTF16</i>	Reads the data from the file in UTF 16 Little Endian (Unicode transformation format where each character is represented by two or four bytes). For this format the no_bytes is taken as the number of characters to read. If the file is an odd length the last character will have the top byte set to zero.

Example

The following example is from the FILEREAD.WIS demonstration script.

```
defaultfile = "wistest\fileread.wis"
filename = FileBrowse("Test of file info/read commands",
defaultfile, 0, "Scripts (*.wis)|*.wis|All (*.*)|*.*")
If filename # "" Then
msg = "Info: " : FileInfo(filename)
File Open f_test, filename
File Read f_test, r_test, 10
Ins msg, "Ascii: " : r_test
File Pointer f_test, 0
File Read f_test, r_test, 10, DF_BS
```

```
Ins msg, "Backslash: " : r_test
File Pointer f_test, 0
File Read f_test, r_test, 10, DF_HEX

Ins msg, "Hex: " : r_test
File Pointer f_test, 0
File Read f_test, r_test, 10, DF_FT
Ins msg, "FT: " : r_test
MessageBox msg
File Close f_test
EndIf
```

Related Script Commands

[File Open](#), [File Close](#)

Version

5.2 Unicode notes. DF_UTF8 & DF_UTF16 formats.

File ReadLine

Syntax

File ReadLine *name, variable, [no_bytes]*

Synonyms

FL RL

Description

This command Reads a line of text up until the next cr, lf, cr/lf or no_bytes have been read.

The line read is automatically converted from the local machines current code page to the Unicode character set.

Parameters

The following table describes the parameters of the File ReadLine command:

Parameter	Description
<i>name</i>	The file identifier from File Open
<i>variable</i>	The name of the variable to read the line of text in to
<i>no_bytes</i>	The maximum number of bytes to read. Default 256

Example

Count the lines in the file opened as f_rltest

```
cnt = 0
Loop
While Not(EndOfFile(f_rltest))
File ReadLine f_rltest, line
cnt += 1
Repeat
```

Related Script Commands

[File Read](#)

Version

4.0.3 Original

5.2 Unicode notes

File Write

Syntax

File Write *name, variable, [options]*

Synonyms

FL WR

Description

The File Write command writes the value of a variable to a file.

The Unicode character set has two bytes per character, so each character of data written has the top byte removed to make this command backward compatible with previous releases.

To write Unicode characters to a file use the DF_UTF8 or DF_UTF16 options.

Parameters

The following table describes the parameters of the File Write command:

Parameter	Description
<i>name</i>	The identifier for the file, specified in File Open.
<i>variable</i>	A variable that you want to write to the file.
<i>options</i>	Specifies different options for writing the data. See the following Options table for more information.

Values for options

Value	Description
<i>DF_BS</i>	Writes data after converting it from backslash format.
<i>DF_Hex</i>	Writes data after converting from Hex values.

Value	Description
<i>DF_FT</i>	Writes data after converting it from File Transfer format.
<i>DF_UTF8</i>	Writes the data in UTF-8 (Unicode transformation format that writes each character in 1 to 4 bytes).
<i>DF_UTF16</i>	Write the data in UTF-16 Little Endian (Unicode transformation format that writes each character as a 2 or 4 byte code). This is the format that is used internally for strings in version 5.2 and later.

Example

In the following example, a file is opened, written to, and then closed:

```
* Open a file, write to it, then close it
var1 = "wIntegrate application development"
If FileExist("C:\pctest\pcnew.txt") Then
File Open new, "C:\pctest\pcnew.txt" OK
If OK Then
File Write new, var1
Else MessageBox "Cannot write to file"
EndIf
* Open pcnew.txt in notepad to show the line written to it
Dialog RunProgram
Set Filename "notepad"
Set Arguments "c:\pctest\pcnew.txt"
Invoke
File Close new
Else EndScript
EndIf
```

Related Script Commands

[File Close](#), [File Open](#), [File Pointer](#), [File Read](#)

Version

5.2 Unicode notes. DF_UTF8 & DF_UTF16 formats.

FileBrowse

Syntax

FileBrowse [*title*], [*filename*], [*option*], [*types*], [*extension*]

Description

The FileBrowse command displays a Windows file selection dialog that allows the user to browse directories and files in order to open a file or choose a file to save data.

Parameters

The following table describes the parameters of the FileBrowse command:

Parameter	Description
<i>title</i>	The title to display in the title bar of the File Browse dialog box.
<i>filename</i>	The name of the default file (the file you want to appear in the "Filename" field).
<i>option</i>	The option to open or to save the file dialog. See below.
<i>types</i>	One or more file types. List the name of the file type first, then the extension type separated by " "
<i>extension</i>	The default extension to use if a file name without an extension is entered in the File name field.

Values for option

Value	Description
0	Opens a file; the default title is "Open File". A Must exist prompt appears if the file name doesn't exist.

Value	Description
1	Saves a file; the default title is "Save file". An Overwrite prompt appears if the file entered already exists.
+2	Add 2 to the above options to suppress the must exists/overwrite prompt.

Return Value

file name chosen or "" if cancel was selected

Example

The following example is part of the FILEREAD.WIS demonstration program.

```
* Demonstrate some file commands
* Version 2.0
defaultfile = "wistest\fileread.wis"
filename = FileBrowse("Test of file info/read commands", defaultfile,
0, "Scripts (*.wis)|*.wis|All (*.*)|*.*")
```

Related Script Commands

[FileList](#), [FolderBrowse](#)

Version

4.1.1 Option to prevent prompt

5.0 Suppress prompt option

FileExist

Syntax

FileExist *filename*

Description

The FileExist function checks if a file exists, and returns "True" if the file exists.

Parameters

The following table describes the parameters of the FileExist command:

Parameter	Description
<i>filename</i>	The full path name of the file.

Return Value

True if file exists, otherwise False

Example

In the following example, the file "pcnew" is created if it does not already exist:

```
* Create the pcnew file if it does not exist
If FileExist("C:\ptest\pcnew.txt") Then
  MsgBox "ptest\pcnew.txt already exists"
Else File Create "C:\ptest\pcnew.txt"
EndIf
```

Related Script Commands

[DirExist](#), [File Create](#), [File Delete](#), [FileInfo](#)

FileInfo

Syntax

FileInfo *filename*, [*options*]

Description

The FileInfo function returns information about a file. If the file does not exist, it returns null.

Parameters

The following table describes the parameters of the FileInfo command:

Parameter	Description
<i>filename</i>	The full path name of the file.
<i>options</i>	Specifies the type of information to return. See the following Options table for more information.

Values for options

Value	Description
1	Returns the file length (number of characters).
2	Returns the date of the latest update of the file.
4	Returns the time of the latest update of the file.
8	Returns the attributes, which can be one or more of the following: A - Archived, H - Hidden, R - Read only, D - Subdirectory, S - System file, V - Volume id

Return Value

Information about the file as specified by the options parameter or "" if the file does not exist

Example

The following example shows how to use FileInfo:

```
* Get information on pctest.txt
If FileExist("C:\pctest\pctest.txt") Then
  MsgBox "": FileInfo("C:\pctest\pctest.txt")
Else
  MsgBox "C:\pctest\pctest.txt cannot be found"
EndIf
```

The next example is taken from the FILEREAD.WIS demonstration program.

```
* Demonstrate some file commands
* Version 2.0
defaultfile = "wistest\fileread.wis"
filename = FileBrowse("Test of file info/read commands",
  defaultfile, 0,
  "Scripts (*.wis)|*.wis|All (*.*)|*.*")
If filename # "" Then
  msg = "Info: " : FileInfo(filename)
  File Open f_test, filename
  File Read f_test, r_test, 10
  Ins msg, "Ascii: " : r_test
```

Related Script Commands

[File Create](#), [File CreateDir](#), [File Delete](#), [File DeleteDir](#), [FileExist](#)

FileList

Syntax

FileList *filespec*, [*options*]

Description

The FileList function checks a directory and returns a list of the files or subdirectories matching the criteria that you specify. Each item in the returned list is separated by a carriage return.

Parameters

The following table describes the parameters of the FileList command:

Parameter	Description
<i>filespec</i>	The type of file extension to return. This can be a wild card expression to match files; any one character or "*" plus optional characters.
<i>options</i>	Specifies other matching options. See the following Options table for more information.

Values for options

Value	Description
0	Return files only.
1	Returns files and directories.
2	Returns files, and includes directories in square brackets.
3	Returns files, and includes directories with ">" prefix.
4	Returns only subdirectories.
8	Returns only hidden files.
16	Returns only system files.

Value	Description
32	Do not include current directory.
64	Strip the extension from the files returned in the list

Return Value

List of files returned

Example

the following example, the program segment uses option 4 to return only the subdirectories:

```
* Get information on files
MessageBox FileList("C:\*.*", 4)
```

In the next example, the program segment shows option 3 added to option 4 to create option 7.

```
* The message box returned by this option is similar to the first
example, but includes the ">" prefix.
* Get information on files
MessageBox FileList("C:\*.*", 7)
```

Related Script Commands

[FileBrowse](#)

Version

5.0 Strip extension option added

Files

Syntax

Files *list_to*

Description

The Files command sends a list of the files in the current account to a specified list box or combo box.

The Files command is a Server Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

The following table describes the parameters of the Files command:

Parameter	Description
<i>list_to</i>	The name of the dialog box and list box to send the results to. The dialog box and list box names are separated by a period, e.g., Dialog.listbox.

Example

In the following example, a partial script shows the creation of a list box within a dialog box, and how the Files command uses the dialog and the list box name.

```
* Create the CUSTOMER Dialog Box
DLib = 'Library "':ScriptFile:'';'
Library 'wintsys\lib\server'
DialogBox Create Cust 15,4,254,120
Caption "Customers"
Style WS_CAPTION | WS_VISIBLE | WS_SYSMENU | WS_THICKFRAME |
WS_MINIMIZEBOX | WS_MAXIMIZEBOX | DS_MODALFRAME | WS_TABSTOP |
WS_GROUP
LText "Files",None,23,2,18,13
ListBox FILELIST,3,15,59,75,WS_BORDER|WS_VSCROLL|WS_TABSTOP
Files Cust.FILELIST
```

FileTempName

Syntax

FileTempName [*Folder*]

Description

This function creates a temporary file and returns its full path name. The file created should be deleted when it has been finished with.

Parameters

The following table describes the parameters of the FileTempName command:

Parameter	Description
<i>Folder</i>	The folder to create the file in. If this argument is omitted the file will be created in the systems temporary folder or the wIntegrate User Directory if no temporary folder has been setup.

Return Value

The return value is the full path name of the file created. If the file cannot be created the return value is "".

Related Script Commands

[File Delete](#)

Version

4.0.3 Original

FindHWND

Syntax

FindHWND *classname, title*

Description

The FindHWND function returns a Windows handle for the window with a specified window class name and/or title. If the class name is omitted or "", only the title is checked. If the title is omitted or "", only the class name is checked.

Parameters

The following table describes the parameters of the FindHWND command:

Parameter	Description
<i>classname</i>	Name of a Windows Class.
<i>title</i>	Title of the window.

Return Value

The window handle of the found window or 0 if it is not found.

Related Script Commands

[GetFocusHWND](#), [HWND](#)

FolderBrowse

Syntax

FolderBrowse [*Prompt*], [*Options*], [*StartFolder*]

Description

This function displays a standard Windows folder selection dialog that allows the user to browse and select a folder. It may also be used to browse for computers or printers.

Parameters

The following table describes the parameters of the FolderBrowse command:

Parameter	Description
<i>Prompt</i>	a text message displayed in the dialog to prompt the user (the default is no prompt)
<i>Options</i>	See below
<i>StartFolder</i>	The name of the folder to show as the initial location selected for the browse.

Values for Options

Value	Description
0	Browse only for folders (default)
1	Only return file system folders (BIF_RETURNONLYFSDIRS)
2	Don't display network folders below domain level (BIF_DONTGOBELOWDOMAIN)
?	Show an edit field for the user to type in an item - only works with later versions of Windows (BIF_EDITBOX)

Value	Description
8	Only return file system ancestors (BIF_RETURNFSANCESTORS)
4096	Browse for computers (BIF_BROWSEFORCOMPUTER)
8192	Browse for printers (BIF_BROWSEFORPRINTER)
16384	Browse for everything, including files (BIF_BROWSEINCLUDEFILES)

Return Value

The name of the folder selected or "" if the dialog was cancelled.

Example

Display a user-selected folder

```
MessageBox FolderBrowse( )
```

Allow user to select a printer name

```
Printer = FolderBrowse("Please select a printer", 8192)  
If Printer = "" Then MessageBox "User cancelled!"
```

Related Script Commands

[FileBrowse](#)

Version

4.0.1 Original

5.1 Added start folder parameter

FontInfo

Syntax

FontInfo *option, param*

Description

The function will return the list of available fonts and the font sizes for a specified font

Parameters

The following table describes the parameters of the FontInfo command:

Parameter	Description
<i>option</i>	Specify information to return. 0 - list of fonts, 1 - list of sizes for the specified font
<i>param</i>	Modifier depending on option. See below for option 0. Use the font name for option 2.

Values for param

The following values are used when the option is set to 0.

Value	Description
<i>0</i>	List all fonts
<i>1</i>	List fixed pitch fonts only
<i>2</i>	List variable pitch fonts only

Return Value

cr separated list of the information requested by the option and param parameters.

Example

Display a message box of fixed pitched fonts

```
MessageBox FontInfo(0,1)
```

Version

5.0 Original

Ftp Close

Syntax

Ftp Close

Synonyms

FTP CL

Description

The Ftp Close command closes a file on a remote FTP site. Since you can only have one remote file open at any one time for a single FTP session, it closes the current open file. Before you can open a file, you must first establish an FTP connection using the Ftp Connect command and you must open the file using the Ftp Open command.

Parameters

None

Example

Examples In the following example, the program segment opens the file "people", reads it, and then closes it using the Ftp Close command:

```
Ftp Open "people", 0, Response
If Response Then
  MessageBox Response
Else
  Ftp Read PeopleData
  MessageBox PeopleData
  Ftp Close
EndIf
```

Related Script Commands

[Ftp Connect](#), [Ftp Open](#)

Ftp Connect

Syntax

Ftp Connect *hostname, username, password, [response]*

Synonyms

FTP CN

Description

The Ftp Connect command closes an FTP (File Transfer Protocol) session with a remote FTP site. You can only have one FTP connection per session.

Parameters

The following table describes the parameters of the Ftp Connect command:

Parameter	Description
<i>hostname</i>	Name of the remote FTP site. You can specify either a TCP/IP domain name or IP address. If no account is held on the remote FTP site, the login name can be set to "anonymous" and the login password set to your own domain name (or e-mail address). However, if you specify a blank name and password, an anonymous login will automatically be set up.
<i>username</i>	Login name.
<i>password</i>	Login password.
<i>response</i>	Variable containing the response from the connect attempt. If the parameter is specified, it will be set to NULL if the connect attempt was successful or an error message if unsuccessful.

Example

In the following example, the program segment opens an anonymous FTP connection:

```
* Login anonymously
Ftp Connect "192.106.24.252", "", ""
```

In the next example, the program segment opens an FTP connection to "MyFtpHost":

```
Ftp Connect "MyFtpHost", "Me", "wqx1r4", Response
If Response Then
  MsgBox Response
Else
  Print FtpGetDir()
Ftp Disconnect
```

Related Script Commands

[Ftp Disconnect](#)

Ftp CreateDir

Syntax

Ftp CreateDir *directory*

Synonyms

FTP CD

Description

The Ftp CreateDir command creates a directory on a remote FTP site. You must first create an FTP connection using the Ftp Connect command before you can create a directory.

Parameters

The following table describes the parameters of the Ftp CreateDir command:

Parameter	Description
<i>directory</i>	The name of the directory to create. The directory name can be a full path, relative path or single subdirectory name.

Example

The following example shows how to create a directory named "tmp":

```
If Not(FtpDirExist("/usr/users/me/tmp")) Then
Ftp CreateDir "/usr/users/me/tmp"
EndIf
```

Related Script Commands

[Ftp Connect](#), [Ftp DeleteDir](#), [FtpDirExist](#)

Ftp Delete

Syntax

Ftp Delete *filename*

Synonyms

FTP DL

Description

The Ftp Delete command deletes a file on a remote FTP site. You must first create an FTP connection using the Ftp Connect command before you can delete a file.

Parameters

The following table describes the parameters of the Ftp Delete command:

Parameter	Description
<i>filename</i>	The name of the remote file to delete. The file name can be specified with or without a full or relative path.

Example

The following table describes the parameters of the Ftp Delete command:

```
If FtpExist("/usr/users/me/tempfile") Then
Ftp Delete "/usr/users/me/tempfile"
EndIf
```

Related Script Commands

[Ftp Connect](#), [Ftp DeleteDir](#), [FtpExist](#)

Ftp DeleteDir

Syntax

Ftp DeleteDir *directory*

Synonyms

FTP DD

Description

The Ftp DeleteDir command deletes a directory on a remote FTP site. You must first create an FTP connection using the Ftp Connect command before you can delete a directory.

Parameters

The following table describes the parameters of the Ftp DeleteDir command:

Parameter	Description
<i>directory</i>	The name of the directory to delete. The directory name can be a full path, relative path or single subdirectory name.

Example

The following example deletes the "tmp" directory:

```
Ftp SetDir "/usr/users/me"  
If FtpDirExist("tmp") Then  
Ftp DeleteDir "tmp"  
EndIf
```

Related Script Commands

[Ftp Connect](#), [Ftp CreateDir](#), [Ftp Delete](#), [FtpDirExist](#)

Ftp Disconnect

Syntax

Ftp Disconnect

Synonyms

FTP DC

Description

The Ftp Disconnect command closes an FTP session with a remote FTP site. You must first create an FTP connection using the Ftp Connect command before you can close the session.

Parameters

None

Example

In the following example, the program segment opens an FTP connection and then closes it:

```
Ftp Connect "MyFtpHost", "Me", "wqxlr4", Response
If Response Then
    MessageBox Response
Else
    Print FtpGetDir()
Ftp Disconnect
```

Related Script Commands

[Ftp Connect](#)

Ftp Get

Syntax

Ftp Get *filename*, *target*, [*type*], [*response*]

Synonyms

FTP GT

Description

The Ftp Get command retrieves a file from a remote FTP site. You must first create an FTP connection using the Ftp Connect command before you can retrieve a file.

Parameters

The following table describes the parameters of the Ftp Get command:

Parameter	Description
<i>filename</i>	The name of the remote file to retrieve. You can specify filename with or without a full or relative path.
<i>target</i>	The local file name where the retrieved file will be saved. If a local file with the same name exists, it will be overwritten.
<i>type</i>	The type of file to retrieve: See below.
<i>response</i>	A variable which will contain "" if successful or a string describing the error if unsuccessful.

Values for type

Value	Description
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.

Value	Description
<i>1</i>	Binary file.

Example

The following example retrieves the "/etc/hosts" file:

```
If FtpExist("/etc/hosts") Then
If Not(FileExist("c:\windows\hosts")) Then
Ftp Get "/etc/hosts", "c:\windows\hosts", 0, Response
If Response Then MessageBox Response
EndIf
EndIf
```

Related Script Commands

[Ftp Connect](#), [Ftp Put](#)

Ftp Move

Syntax

Ftp Move *oldname, newname*

Synonyms

FTP MV

Description

The Ftp Move command moves or renames a file on a remote FTP site.

You must first create an FTP connection using the Ftp Connect command before you can move a file.

Parameters

The following table describes the parameters of the Ftp Move command:

Parameter	Description
<i>oldname</i>	The name of the existing remote file to move/rename. File names can be specified with or without a full or relative path.
<i>newname</i>	The new name to assign.

Example

In the following example, the program segment renames "tempfile" to "permfile":

```
Ftp SetDir "/usr/users/me"  
If FtpExist("tempfile") Then  
Ftp Move "tempfile", "permfile"  
EndIf
```

Related Script Commands

[Ftp Connect](#)

Ftp Open

Syntax

Ftp Open *filename*, [*option*], [*response*]

Synonyms

FTP OP

Description

The Ftp Open command opens a file on a remote FTP site. You can only have one remote file open at any one time for a single FTP session. You must first create an FTP connection using the Ftp Connect command before you can open a file.

Parameters

The following table describes the parameters of the Ftp Open command:

Parameter	Description
<i>filename</i>	The name of the remote file to open. filename can be specified with or with-out a full or relative path.
<i>option</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 table below
<i>response</i>	A variable which will be set to "True" if successful or "False" if unsuccessful.

Values for option

Value	Description
0	Read ASCII text (the default). 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.

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.

Example

In the following example, the program segment reads the contents of the remote data file "people":

```
Ftp Open "people", 0, Response
If Response Then
  MsgBox Response
Else
  Ftp Read PeopleData
  MsgBox PeopleData
Ftp Close
EndIf
```

Related Script Commands

[Ftp Close](#), [Ftp Connect](#), [Ftp Read](#), [Ftp Write](#)

Ftp Pointer

Syntax

Ftp Pointer *position*

Synonyms

FTP PN

Description

The Ftp Pointer command sets the current file position for reading from a file on a remote FTP site. You must first create an FTP connection using the Ftp Connect command and open the file using the Ftp Open command before you can use this command.

Parameters

The following table describes the parameters of the Ftp Pointer command:

Parameter	Description
<i>position</i>	The number of bytes from the beginning of the file. A value of 0 sets the pointer to the beginning of the file and a value of -1 sets it to the end.

Example

In the following example, the program segment reads the file "people" twice:

```
Ftp Open "people", 0, Response
If Response Then
  MessageBox Response
Else
  Ftp Read PeopleData
  Ftp Pointer 0
  Ftp Read PeopleDataCopy
  Ftp Close
EndIf
```

Related Script Commands

[Ftp Close](#), [Ftp Connect](#), [Ftp Open](#), [Ftp Read](#), [Ftp Write](#)

Ftp Put

Syntax

Ftp Put *filename, target, [option], [response]*

Synonyms

FTP PT

Description

The Ftp Put command sends a file to a remote FTP site. You must first create an FTP connection using the Ftp Connect command before you can transfer a file.

Parameters

The following table describes the parameters of the Ftp Put command:

Parameter	Description
<i>filename</i>	The name of the local file to send. filename can be specified with or without a full or relative path. If filename already exists on the remote machine, it will be overwritten.
<i>target</i>	The remote file name where the sent file will be saved
<i>option</i>	Specifies the type of file to transfer. See below
<i>response</i>	A variable which is set to "" if successful or a string describing the error if unsuccessful.

Values for option

Value	Description
0	ASCII text file (the default). Select the text 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.

Value	Description
<i>1</i>	Binary file.

Example

In the following example, the program segment sends the ASCII file “savworld.c” to the remote machine and changes the name to "savtheworld.c":

```
Ftp Put "c:\dev\savworld.c", "savtheworld.c", 0, Response
If Response Then
  MsgBox Response
Else
  Print "File updated"
```

Related Script Commands

[Ftp Connect](#), [Ftp Disconnect](#), [Ftp Get](#)

Ftp Read

Syntax

Ftp Read *variable*, [*bytes*], [*options*]

Synonyms

FTP RD

Description

This command reads data from a file on a remote FTP site.

If the number of bytes to read is omitted then up to 32 kilobytes will be read. The maximum number of bytes that can be read is 64 kilobytes.

An FTP connection must be established (using Ftp Connect) and the file must have been opened first (using Ftp Open) for this command to be successful.

The file data conversions are useful for manipulating files containing binary data (especially NULs, ie. character zeros).

The Unicode character set has two bytes per character, so each byte of data read is placed in the low-byte of a character to make it backward compatible with previous releases.

Parameters

The following table describes the parameters of the Ftp Read command:

Parameter	Description
<i>variable</i>	The name of the variable into which data will be read.
<i>bytes</i>	The number of bytes to read. If the number of bytes to read is omitted then up to 32 Kilobytes will be read.
<i>options</i>	See below

Values for options

Value	Description
<i>0</i>	No conversion (default)
<i>DF_BS (1)</i>	Convert data to backslash format
<i>DF_HEX (2)</i>	Convert data to 2-character hex format
<i>DF_FT (3)</i>	Convert data to File Transfer format

Example

Read in names from remote data file

```
Ftp Open "people", 0, Response
If Response Then
  MsgBox Response
Else
  Ftp Read PeopleData
  MsgBox PeopleData
Ftp Close
EndIf
```

Related Script Commands

[Ftp Close](#), [Ftp Connect](#), [Ftp Open](#), [Ftp Write](#)

Version

4.0.1 Original version

5.2 Unicode notes

Ftp SetDir

Syntax

Ftp SetDir *directory*

Synonyms

FTP SD

Description

The Ftp SetDir command sets the current working directory on a remote FTP site. You must first create an FTP connection using the Ftp Connect command before you can rename a file.

Parameters

The following table describes the parameters of the Ftp SetDir command:

Parameter	Description
<i>directory</i>	The name of the new current directory. The directory name can be a full path, relative path or single subdirectory name.

Example

The following example sets the working directory to "/usr/users/me" and then deletes the "tmp" directory:

```
Ftp SetDir "/usr/users/me"  
If FtpDirExist("tmp") Then  
Ftp DeleteDir "tmp"  
EndIf
```

Ftp Write

Syntax

Ftp Write *variable, options*

Synonyms

FTP WR

Description

The Ftp Write command writes data to a file on a remote FTP site. You must first create an FTP connection using the Ftp Connect command and open the file using the Ftp Open command before using this command.

The Unicode character set has two bytes per character, so each character of data written has the top byte removed to make this command backward compatible with previous releases.

Parameters

The following table describes the parameters of the Ftp Write command:

Parameter	Description
<i>variable</i>	The name of the variable containing data to be written.
<i>options</i>	Conversion for data before writing. See below.

Values for options

Value	Description
<i>0</i>	No conversion (default)
<i>DF_BS (1)</i>	Convert data from back slash format
<i>DF_HEX (2)</i>	Convert data from 2-character hex format
<i>DF_FT (3)</i>	Convert data from File Transfer format

Example

In the following example, the program segment writes a series of names in the "people" remote data file:

```
PeopleData = "Smith" : cr : "Carruthers" : cr : "Crawford" : cr
Ftp Open "people", 0, Response
If Response Then
  MessageBox Response
Else
  Ftp Write PeopleData
Ftp Close
EndIf
```

Related Script Commands

[Ftp Close](#), [Ftp Connect](#), [Ftp Open](#), [Ftp Read](#)

Version

5.2 Unicode notes

FtpDirExist

Syntax

FtpDirExist *directory*

Description

The FtpDirExist function determines if a specified directory exists. It returns "True" if the file exists or "False" if it does not exist. You must first create an FTP connection using the Ftp Connect command before you can use this function.

Parameters

The following table describes the parameters of the FtpDirExist command:

Parameter	Description
<i>directory</i>	The name of the directory to check for existence. You must specify the correct case since this function is case sensitive. You can specify a directory with or without a full or relative path. However, paths can only be used if the remote FTP site uses "/" or "\" as the directory delimiter characters. Tip - Ftp SetDir will enable you to avoid the use of paths.

Return Value

True if the directory exists on the FTP server, otherwise false.

Example

The following example checks to see if the directory named "tmp" exists before creating it:

```
If Not(FtpDirExist("/usr/users/me/tmp")) Then
Ftp CreateDir "/usr/users/me/tmp"
EndIf
```

Related Script Commands

[Ftp Connect](#), [Ftp CreateDir](#), [Ftp DeleteDir](#), [Ftp SetDir](#), [FtpExist](#)

FtpExist

Syntax

FtpExist *filename*

Description

The FtpExist function determines if a specified file exists or not. It returns "True" if the file exists or "False" if it does not exist. You must first create an FTP connection using the Ftp Connect com-mand before you can use this function.

Parameters

The following table describes the parameters of the FtpExist command:

Parameter	Description
<i>filename</i>	The name of the file to check for existence. You must specify the correct case since this function is case sensitive. The filename can be specified with or without a full or relative path. However, paths can only be used if the remote FTP site uses "/" or "\" as directory delimiter characters. Tip - Ftp SetDir will enable you to avoid the use of paths.

Return Value

True if the file exists on the ftp server, otherwise false

Example

Delete tempfile if it exists

```
If FtpExist("/usr/users/me/tempfile") Then
  Ftp Delete "/usr/users/me/tempfile"
EndIf
```

Related Script Commands

[Ftp Connect](#), [FtpDirExist](#)

FtpGetDir

Syntax

FtpGetDir

Description

The FtpGetDir function returns the name of the current working directory on a remote FTP site. You must first create an FTP connection using the Ftp Connect command before you can use this function.

Parameters

None

Return Value

The name of the current directory on the FTP server

Example

In the following example, the program segment retrieves the name of the current working directory on "MyFtpHost":

```
Ftp Connect "MyFtpHost", "Me", "wqxlr4", Response
If Response Then
  MsgBox Response
Else
  Print FtpGetDir()
Ftp Disconnect
```

Related Script Commands

[Ftp Connect](#), [FtpList](#)

FtpInfo

Syntax

FtpInfo *FileName*, *Options*

Description

This function provides information about a specified file on a remote FTP site.

An FTP connection must be established (using `FtpConnect`) for this function to be successful.

If this function is called in conjunction with the opening of a file for manipulation of its contents, it is recommended that `FtpInfo` is called before `Ftp Open` or after `Ftp Close`.

It seems that not all the information returned by this function can be relied upon (the information returned depends on the remote FTP site). An obvious use of this function is to get the length of a file before retrieving it so that a progress bar maybe displayed during transfer (see the `wIntegrate FTP File Manager` for an example of this).

Parameters

The following table describes the parameters of the `FtpInfo` command:

Parameter	Description
<i>FileName</i>	the name of the remote file to retrieve information about
<i>Options</i>	See below

Values for Options

Value	Description
1	Return file length
2	Return last modification date

Value	Description
<i>4</i>	Return last modification time
<i>8</i>	Return file attributes (see below)

Return Value

The requested information on the remote file

The file attribute letters returned are:

Value	Description
<i>N</i>	Normal
<i>A</i>	Archive
<i>H</i>	Hidden
<i>R</i>	Read-only
<i>D</i>	Directory
<i>S</i>	System

Example

Display full details of a remote file

```
MessageBox FtpInfo("/usr/users/me/myfile")
```

Test the length of a remote file before retrieval

```
FileLength = FtpInfo("foo.txt", 1)  
If FileLength > 0 Then Ftp Get "foo.txt", "c:\Download\foo.txt"
```

Version

4.0.1 Original

FtpList

Syntax

FtpList *path, options*

Description

The FtpList function returns a list of the files and/or subdirectories in the current working directory matching the criteria that you specify. Each item in the returned list is separated by a carriage return. You must first create an FTP connection using the Ftp Connect command before you can use this function.

Parameters

The following table describes the parameters of the FtpList command:

Parameter	Description
<i>path</i>	A wild-card expression for file matching.
<i>options</i>	See below

Values for options

Options for returning list. You can specify options 1, 2, or 3 and add 4 or 32 to these options.

Value	Description
<i>1</i>	Include directories.
<i>2</i>	Include directories square brackets.
<i>3</i>	Prefix directories with >.
<i>+4</i>	List directories only.
<i>+32</i>	Omit current directory (.).

Return Value

list of files and directories specified by options

Example

In the following example, the program segment lists only the directories (option 2 + 4):

```
Ftp SetDir("/usr/users")
Users = FtpList("*.*", 6)
```

In the next example, the two functions produce the same result:

```
MessageBox FtpList("/etc")
MessageBox FtpList("/etc/*.*.")
```

Related Script Commands

[Ftp Connect](#)

FTToAscii

Syntax

FTToAscii *string*

Description

The FTToAscii function 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 FTToAscii command:

Parameter	Description
<i>string</i>	The string to convert

Return Value

A copy of the string in ascii format

Related Script Commands

[AsciiToBS](#), [AsciiToFT](#), [AsciiToHex](#), [BStoAscii](#), [HexToAscii](#)

Get

Syntax

Get *var_name*

Description

The Get function returns the current value of a session variable.

Parameters

The following table describes the parameters of the Get command:

Parameter	Description
<i>var_name</i>	The name of a wIntegrate session variable.

Return Value

The current value of the session variable

Example

In the following example, the Get function returns the current icon bar script:

```
* Display a message box with the current icon bar script
MessageBox Get(IconBarScript)
```

The next example is taken from the AP.WIS demonstration program.

```
Set F1 = Get(F9) : "E"
Set F2 = Get(F9) : "F"
Set F3 = Get(F9) : "S"
Set F4 = Get(F9) : "C"
```

Related Script Commands

[Dialog](#), [Set](#), [Store](#)

GetFocus

Syntax

GetFocus [*dboxname*]

Description

The GetFocus function returns the name of the control that currently has the focus. If the focus is on an unnamed control, on the dialog box or is outside of the dialog box, the function returns "".

If the dialog box name is omitted the name of the dialog box which has the focus or contains the control with the focus is returned. If no shown dialog box has the focus "" is returned.

Parameters

The following table describes the parameters of the GetFocus command:

Parameter	Description
<i>dboxname</i>	The dialog box in which you want to return the control that has the focus.

Return Value

Name of the control with the focus or ""

Example

Set field to "0" if the PushButton with name Zero and caption "&Zero" is run using Alt-Z.

```
name = GetFocus(Order)
If name # "Zero" Then
name = "Order." : name
$name = 0
EndIf
```

Check if any control on the order dialog has the focus

```
If GetFocus() = "Order" Then
    Print "Order has the focus"
EndIf
```

Version

4.1.0 dbxname parameter made optional

GetFocusHWnd

Syntax

GetFocusHWnd

Description

The GetFocusHWnd function returns the window handle of the wIntegrate window that has the focus. It returns 0 if no window has the focus.

Parameters

None

Return Value

The handle of the wIntegrate window with the focus

Related Script Commands

[FindHWnd](#), [HWnd](#)

GetFromHost Next

Syntax

GetFromHost Next *to_var*, *number_chars*, [*timeout*]

Synonyms

GH NX

Description

The GetFromHost Next command captures the specified number of characters from the host and stores them in the specified variable. If the timeout expires before all of the characters have been received, the variable is set to the characters that have been received so far.

Parameters

The following table describes the parameters of the GetFromHost Next command:

Parameter	Description
<i>to_var</i>	The variable that stores the captured characters.
<i>number_chars</i>	The number of characters to capture.
<i>timeout</i>	The maximum time to wait for the characters. This is specified in the same way as in the Wait commands.

Example

In the following example, the program segment shows the use of the GetFromHost Next command:

```
GetFromHost Next mydata,4
If Length(mydata) < 4 Then MessageBox "Didn't get enough data"
```

Related Script Commands

[GetFromHost RawData](#)

GetFromHost RawData

Syntax

GetFromHost RawData *to_var*, *number_chars*, [*timeout*], [*reset_timeout*], [*end_chars*]

Synonyms

GH RD

Description

The GetFromHost RawData directly captures the specified number of characters received from the host and stores them in the specified variable.

This routine directly captures the data, stopping it from going through the emulation and effecting the screen.

If the timeout expires before all the characters have been received, the variable is set to the characters that have been received so far.

If the end_chars parameter is specified the command will return if any of the character in end_chars are received from the host before the specified number of characters have been received.

Parameters

The following table describes the parameters of the GetFromHost RawData command:

Parameter	Description
<i>to_var</i>	The variable that stores the captured characters.
<i>number_chars</i>	The number of characters to capture.
<i>timeout</i>	The maximum time to wait for the characters. This is specified in the same way as in the Wait commands.

Parameter	Description
<i>reset_timeout</i>	Reset the time out whenever a character is received. This means that the command will finish after the timeout period has been exceeded after the last character was sent up from the host. Default False (0).
<i>end_chars</i>	If specified the command will stop when it receives any of the characters in this string. The end characters must have a code values in the range 0 to 255.

Example

In the following example, the program segment demonstrates the GetFromHost RawData command:

```
GetFromHost RawData mydata,4
If Length(mydata)< 4 Then MessageBox "Didn't get enough data"
```

Wait for a tab or cr character within the next 20 chars, timeing out if there is a 2 second gap between received characters:

```
GetFromHost RawData var, 20,"2s",True, tab:cr
If Right(var,1) = tab Then
    MessageBox "Tab delimited"
Else If Right(var,1) = cr Then
    MessageBox "Return delimited"
Else MessageBox "Timed out"
```

Related Script Commands

[GetFromHost Next](#)

Version

4.0.1 reset_timeout and end_chars parameters added

GetIconColumn

Syntax

GetIconColumn

Description

The `GetIconColumn` function returns the current column position during the definition of a toolbar. The value returned can be used with the `SetIconColumn` command to line up icons on an icon bar which consists of more than one row.

The `GetIconColumn` command is an Icon Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

None

Return Value

The current column position during the creation of an icon bar

Version

4.0 Original

GetMachineType

Syntax

GetMachineType

Description

The **GetMachineType** function returns the current host machine type and sets the global variables **Server_Version** and **Machine_Type**.

The **GetMachineType** command is a Server Library command. For more information, see "Appendix A - Scripting Libraries and Modules".

Parameters

None

Example

The following example displays the machine type in a message box

```
Library "wintsys\lib\server"  
GetMachineType  
If Server_Error Then  
  MessageBox ServerErrMsg(Server_Error)  
Else MessageBox "" : Machine_type  
EndIf  
CloseServer
```

Related Script Commands

[CloseServer](#), [ServerErrMsg](#)

Global

Syntax

Global *var_name* [= *value*] ...

Synonyms

GL

Description

The Global command creates up to five global script variables.

A global script variable is one that retains its value between running scripts. Each wIntegrate session has its own set of global variables.

Up to five variables can be defined at a time in a single global command.

Use the Destroy command to remove up to five global variables from memory.

Parameters

The following table describes the parameters of the Global command:

Parameter	Description
<i>var_name</i>	The name of the variable to create.
<i>value</i>	The initial value of the variable. If this is omitted, the variable value is null.

Example

The following example is part of the PRESTORE.WIS demonstration program.

```
* Prestore keystrokes
* Version 2.3, 20 June 1994
* Check The "R" check box, and then press a button (P1 - P9) to
* to start recording, press the button again to stop recording
* Uncheck the "R" check box, and then press the button again to
* play the keystrokes. The "D" button shows the current definitions.
If Not(IsDialog(Prestore)) Then
DLib = 'Library "'':ScriptFile:'';'
Global Prestore_State,Prestore_Text
```

Related Script Commands

[Destroy](#)

Green

Syntax

Green *color*

Description

The Green function returns the green intensity of a specified RGB (Red Green Blue) color. On a color monitor, colors are displayed as varying intensities of red, green, and blue dots. Intensity is the amount of red, green, or blue on a scale of 0 through 255.

Parameters

The following table describes the parameters of the Green command:

Parameter	Description
<i>color</i>	The color name or number.

Return Value

The amount of green in the color (0 to 255)

Example

This returns 255 as yellow contains 100% green

```
Print Green(RGB_Yellow)
```

Related Script Commands

[Blue](#), [ChooseColor](#), [Red](#), [RGB](#)

HCall

Syntax

HCall *subname*, *parms*, *waitToFin*

Description

The HCall command calls a host subroutine. The HCall command is a Server Library command.

For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

The following table describes the parameters of the HCall command:

Parameter	Description
<i>subname</i>	The name of the host subroutine that you want to call.
<i>parms</i>	The parameters of the subroutine, passed as a string.
<i>waitToFin</i>	Set this flag to: True - If you want the script to wait for the subroutine to finish. False - If you want the script to continue before the subroutine has finished.

HDelete

Syntax

HDelete *filename, item*

Description

The HDelete command deletes an item or record from a host file. The HDelete command is a Server Library command.

For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

The following table describes the parameters of the HDelete command:

Parameter	Description
<i>filename</i>	The name of the file.
<i>item</i>	The name of the item (record) to delete.

Help TopicId

Syntax

Help TopicId *id*, [*filename*]

Synonyms

HP TI

Description

The Help TopicId command displays the Help topic from the windows help file with the given topic number.

Parameters

The following table describes the parameters of the Help TopicId command:

Parameter	Description
<i>id</i>	Id number for help topic.
<i>filename</i>	Name of the help file (the default is winteg.hlp).

Example

The following example shows the use of the Help TopicId command:

```
Help TopicId 2501;* Show help on query builder
```

Related Script Commands

[Help TopicName](#)

Help TopicName

Syntax

Help TopicName *name*, [*filename*]

Synonyms

HP TN

Description

The Help TopicName command display the Help topic from the windows help file with the given topic name.

Parameters

The following table describes the parameters of the Help TopicName command:

Parameter	Description
<i>name</i>	Name of the topic to lookup.
<i>filename</i>	Name of the help file (the default is winteg.hlp).

Related Script Commands

[Help TopicId](#)

HexToAscii

Syntax

HexToAscii *string*

Description

The HexToAscii function converts a 2-character hex string to ASCII format. In Hex format, every byte is a two-byte hexadecimal version, so a particular byte position can be located directly.

Parameters

The following table describes the parameters of the HexToAscii command:

Parameter	Description
<i>string</i>	The string to convert to ASCII text.

Return Value

Copy of the string converted to ASCII

Example

Read in data from file in hex format

```
File Read f, rec,,DF_Hex
byte = Mid(rec, 10, 2) ;* Get 5th bytes (2 hex digits * 5 = 10)
* Check if it's an A
If HexToAscii(byte) = "A" Then
...

```

Related Script Commands

[AsciiToBS](#), [AsciiToFT](#), [AsciiToHex](#), [BStoAscii](#), [FTToAscii](#)

Host Break

Syntax

Host Break

Synonyms

HT BR

Description

This command sends the break signal appropriate to the current communications to the host.

Parameters

None

Host Get

Syntax

Host Get *variable*

Synonyms

HT GT

Description

The Host Get command is the PC side of a simple data transfer scheme.

Once this command is run the script waits until it receives data in the correct format and then sets the variable.

It is used by various host routines. e.g. WIN.ASSIGN, WIN.DBMETHOD.

Parameters

The following table describes the parameters of the Host Get command:

Parameter	Description
<i>variable</i>	The name of the script variable to receive the data from the host

Related Script Commands

[Host Send](#)

Host Send

Syntax

Host Send *text*

Synonyms

HT SN

Description

The Host Send command is the PC side of a simple data transfer to the host.

It sends the text given to it to the host in a special format. This is used by various host routines. e.g. WIN.EVAL, WIN.DBMETHOD.

Parameters

The following table describes the parameters of the Host Send command:

Parameter	Description
<i>text</i>	The data to send to the host

Related Script Commands

[Host Get](#)

HotSpot Add

Syntax

HotSpot Add *left, top, right, bottom, [click_text], [double_text], [style], [fore_color], [back_color]*

Synonyms

HS AD

Description

The HotSpot Add command adds a hot spot to a dialog box. The position of the hot spot is specified by character cells on the host screen (usually 0-79 across and 0-23 down).

Parameters

The following table describes the parameters of the HotSpot Add command:

Parameter	Description
<i>left</i>	The screen column position to start the hot spot.
<i>top</i>	The screen row position to start the hot spot.
<i>right</i>	The column position for the hot spot to extend on the right.
<i>bottom</i>	The row position for the hot spot to extend on the bottom.
<i>click_text</i>	The action to take place when the hot spot is clicked once. To run a script command instead of sending the text to the host prefix the script command with "\m".
<i>double_text</i>	The action to take place when the hot spot is clicked twice. If the text for a click is preceded by "\m", it runs as a script, otherwise data is sent to the host.
<i>style</i>	The style of the HotSpot. See below

Parameter	Description
<i>fore_color</i>	Foreground color (the default is COL_Black (0)). See the following
<i>back_color</i>	Background color (the default is COL_White (15)). See the following Colors table for fore_color for more information.

Values for style

Value	Description
<i>0</i>	Hotspot is transparent. The default is 0.
<i>1</i>	Hotspot is raised with a grey/white border.
<i>2</i>	Hotspot is sunk, with a grey/white border.
<i>3</i>	Hotspot looks like a text box with a 2 pixel grey/white and black border.
<i>64</i>	overrides the screen color. Use it by itself or add it to other flag numbers.

Values for fore_color

Value	Description
<i>COL_Black</i>	0
<i>COL_Blue</i>	1
<i>COL_Green</i>	2
<i>COL_Cyan</i>	3
<i>COL_Red</i>	4
<i>COL_Magenta</i>	5
<i>COL_Brown</i>	6

Value	Description
<i>COL_LightGray</i> or <i>COL_LightGrey</i>	7
<i>COL_Gray</i> or <i>COL_Grey</i>	8
<i>COL_LightBlue</i>	9
<i>COL_LightGreen</i>	10
<i>COL_LightCyan</i>	11
<i>COL_LightRed</i>	12
<i>COL_LightMagenta</i>	13
<i>COL_Yellow</i>	14
<i>COL_White</i>	15

Example

The following example adds hot spots to the window.

```
HotSpot Add 0,0,39,11, "Top Left Hotspot"  
HotSpot Add 40,0,79,11,"Top Right Hotspot"  
HotSpot Add 0,12,79,23, "\mMessageBox 'click on the top half of the  
screen'", "\mMessageBox 'double click on the top half of the screen'"
```

Related Script Commands

[HotSpot AutoDelete](#), [HotSpot Delete](#), [HotSpot DeleteAll](#), [Hotspot Display](#), [HotSpot Off](#), [HotSpot On](#)

HotSpot AutoDelete

Syntax

HotSpot AutoDelete *flag*

Synonyms

HS AU

Description

The HotSpot AutoDelete command automatically deletes hot spots whenever it receives a form feed from the host.

Parameters

The following table describes the parameters of the HotSpot AutoDelete command:

Parameter	Description
<i>flag</i>	Can be one of the following: True - Automatically deletes hot spots whenever a from feed is received from the host. False - Does not delete hot spots.

Related Script Commands

[HotSpot Add](#), [HotSpot Delete](#), [HotSpot DeleteAll](#), [Hotspot Display](#), [HotSpot Off](#), [HotSpot On](#)

HotSpot Delete

Syntax

HotSpot Delete *left, top, right, bottom*

Synonyms

HS DL

Description

The HotSpot Delete command deletes a hot spot at the specified position.

Parameters

The following table describes the parameters of the HotSpot Delete command:

Parameter	Description
<i>left</i>	The starting screen column position of the hot spot.
<i>top</i>	The starting screen row position of the hot spot.
<i>right</i>	The column position the hot spot extends to.
<i>bottom</i>	The row position that the hot spot extends to.

Example

In the following example, the HotSpot Delete command deletes two hot spots:

```
HotSpot Delete 0,0,39,11 ; * Top Left Hotspot  
HotSpot Delete 40,0,79,11 ; * Top Right Hotspot
```

Related Script Commands

[HotSpot Add](#), [HotSpot AutoDelete](#), [HotSpot DeleteAll](#), [Hotspot Display](#), [HotSpot Off](#), [HotSpot On](#)

HotSpot DeleteAll

Syntax

HotSpot DeleteAll

Synonyms

HS DA

Description

The HotSpot DeleteAll command deletes all currently defined hot spots.

Parameters

None

Example

The following example creates two hot spots and then deletes them using the HotSpot DeleteAll command:

```
HotSpot Add 0,0,39,11, "Top Left Hotspot"
HotSpot Add 40,0,79,11,"Top Right Hotspot"
.
.
.
HotSpot DeleteAll
```

Related Script Commands

[HotSpot Add](#), [HotSpot AutoDelete](#), [HotSpot Delete](#), [Hotspot Display](#), [HotSpot Off](#), [HotSpot On](#)

HotSpot Display

Syntax

HotSpot Display *style, fore_col, back_col*

Synonyms

HS DS

Description

The HotSpot Display command displays hot spots in the colors that you specify. You must first create hot spots with HotSpot Add.

Parameters

The following table describes the parameters of the HotSpot Display command:

Parameter	Description
<i>style</i>	The style of the hotspot:
<i>fore_col</i>	Designates the foreground color. Use color numbers or constants.
<i>back_col</i>	Designates the background color. Use color numbers or constants.

Values for style

Value	Description
<i>0</i>	Hotspot is transparent. The default is 0.
<i>1</i>	Hotspot is raised with a grey/white border.
<i>2</i>	Hotspot is sunk, with a grey/white border.
<i>3</i>	Hotspot looks like a text box with a 2 pixel grey/white and black border.
<i>64</i>	Overrides the screen color. Use it by itself or add it to other flag numbers.

Example

The following are examples of different ways to use HotSpot Display command:

```
HotSpot Display 64;* Display hotspots black on white
```

```
HotSpot Display 64, COL_Red, COL_Yellow;* Display as red on yellow
```

```
HotSpot Display ;* Make hotspots invisible
```

```
HotSpot Display 65, COL_Black,COL_LightGrey;* Raised hotspots
```

```
HotSpot Display 3 ;* Text box effect with no coloring
```

Related Script Commands

[HotSpot Add](#), [HotSpot AutoDelete](#), [HotSpot Delete](#), [HotSpot DeleteAll](#), [HotSpot Off](#), [HotSpot On](#)

HotSpot Off

Syntax

HotSpot Off

Synonyms

HS OF

Description

The HotSpot Off command turns off hot spot processing after processing has been turned on by the HotSpot On command.

Parameters

None

Related Script Commands

[HotSpot Add](#), [HotSpot AutoDelete](#), [HotSpot Delete](#), [HotSpot DeleteAll](#), [Hotspot Display](#), [HotSpot On](#)

HotSpot On

Syntax

HotSpot On

Synonyms

HS ON

Description

The HotSpot On command turns on hot spot processing after processing has been turned off by the HotSpot Off command.

Parameters

None

Related Script Commands

[HotSpot Add](#), [HotSpot AutoDelete](#), [HotSpot Delete](#), [HotSpot DeleteAll](#), [Hotspot Display](#), [HotSpot Off](#)

HourGlass Off

Syntax

HourGlass Off

Synonyms

HG OF

Description

The HourGlass Off command turns off an hourglass cursor turned on by HourGlass On. The hourglass is only removed when the number of HourGlass Off statements matches the number of HourGlass On statements.

Tip To turn off all hourglass processing at once, use the HourGlass Smash command.

Parameters

None

Example

The following example shows the use of the hourglass cursor while running a report.

```
HourGlass On
Enter "SORT CUSTOMER NAME BY NAME"
Print ""
HourGlass Off
```

Related Script Commands

[CursorShape](#), [HourGlass On](#), [HourGlass Smash](#)

HourGlass On

Syntax

HourGlass On

Synonyms

HG ON

Description

The HourGlass On command switches on the HourGlass cursor and disables the mouse. Use on lengthy operations where no other windows operation can be selected while the process is running. The HourGlass is shared between all running session windows. To turn the hourglass off, use HourGlass Off once for every call to HourGlass On.

Tip To turn off all hourglass processing at once, use the HourGlass Smash command.

Parameters

None

Example

The following example shows the use of the hourglass cursor while running a report.

```
HourGlass On
Enter "SORT CUSTOMER NAME BY NAME"
Print ""
HourGlass Off
```

Related Script Commands

[CursorShape](#), [HourGlass Off](#), [HourGlass Smash](#)

HourGlass Smash

Syntax

HourGlass Smash

Synonyms

HG SM

Description

The HourGlass Smash command turns off the HourGlass no matter how many HourGlass On statements have been used. It should only be used to restore the cursor if a test script has mis-matched the number of HourGlass On and HourGlass Off statements.

Parameters

None

Related Script Commands

[CursorShape](#), [HourGlass Off](#), [HourGlass On](#)

HRead

Syntax

HRead *var_name, filename, item*

Description

The HRead command reads an item or record from a host file. You must then check the Server_Error value to verify that the record exists and if it was read.

The HRead command is a Server Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

The following table describes the parameters of the HRead command:

Parameter	Description
<i>var_name</i>	The name of a script variable to which HRead reads data.
<i>filename</i>	The name of the file to read.
<i>item</i>	The name of the item (record) to read.

Example

The following example is a part of the PRODM2.WIS demonstration program.

```
* Validate Id, read record if it exists
Sub OnValidateId()
Library 'wintsys\lib\server'
If ProdM.ProdRef Then
HRead r_prod 'WIN.PROD',ProdM.ProdRef
ProdM_Mode = 8
If Server_Error = 5 Then
Server_Error = 0
r_prod = ''
Rep r_prod,2,3
ProdM_Mode = 1
EndIf
```

Related Script Commands

[HReadU](#), [HReadV](#), [HReadVU](#), [HWrite](#)

HReadU

Syntax

HReadU *var_name, filename, item, fldno*

Description

The HReadU command reads a single record from a host file and locks the item. You must then check the Server_Error value to verify if the item exists and was read and locked. Use HRelease to release the lock the item locked by HReadU.

The HReadU command is a Server Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

The following table describes the parameters of the HReadU command:

Parameter	Description
<i>var_name</i>	The name of a script variable to which HRead reads data.
<i>filename</i>	The name of the file to read.
<i>item</i>	The name of the item (record) to read.
<i>fldno</i>	The field number to read.

Related Script Commands

[HRead](#), [HReadV](#), [HReadVU](#), [HRelease](#), [HWriteU](#)

HReadV

Syntax

HReadV *var_name, filename, item, fldno*

Description

The HReadV command reads a single field from a host file.

The HReadV command is a Server Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

The following table describes the parameters of the HReadV command:

Parameter	Description
<i>var_name</i>	The name of a script variable to which HRead reads data.
<i>filename</i>	The name of the file to read.
<i>item</i>	The name of the item (record) to read.
<i>fldno</i>	The field number to read.

Example

The following example code is taken from the DDE_SCR.WIS demonstration program.

```
Sub GetAndSend(HostFile, HostItem, HostAtt, $LinkName, LinkItem)
HReadv Value, HostFile, HostItem, HostAtt
If Server_Error Then
  MsgBox ServerErrMsg(Server_Error) : " " : HostFile : ", " :
  HostItem
Else
  DDE Poke $LinkName, LinkItem, Value
EndIf
EndSub
```

Related Script Commands

[HRead](#), [HReadU](#), [HReadVU](#), [HRelease](#), [HWriteV](#)

HReadVU

Syntax

HReadVU *var_name, filename, item, fldno*

Description

The HReadVU command reads a single field from a host file and locks the item. You must then check the Server_Error value to verify if the item exists and was read and locked. Use HRelease to release the lock on the item locked by HReadVU.

The HReadVU command is a Server Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

The following table describes the parameters of the HReadVU command:

Parameter	Description
<i>var_name</i>	The name of a script variable to which HReadVU reads data.
<i>filename</i>	The name of the file to read.
<i>item</i>	The name of the item (record) to read.
<i>fldno</i>	The field number to read. HReadV Parameters

Related Script Commands

[HRead](#), [HReadU](#), [HReadV](#), [HRelease](#), [HWriteVU](#)

HRelease

Syntax

HRelease *filename, item*

Description

The HRelease command releases a lock from a host item.

The HRelease command is a Server Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

The following table describes the parameters of the HRelease command:

Parameter	Description
<i>filename</i>	The name of the host file.
<i>item</i>	The name of the item or record to release.

Related Script Commands

[HReadU](#), [HReadV](#), [HReadVU](#), [HWriteU](#), [HWriteV](#), [HWriteVU](#)

HWnd

Syntax

HWnd *dboxname*, [*control*]

Description

The HWnd command returns the Windows handle of the specified dialog box or control.

Parameters

The following table describes the parameters of the HWnd command:

Parameter	Description
<i>dboxname</i>	Dialog box name.
<i>control</i>	Name of the control.

Return Value

Window handle or 0 if the dialog box or control is not shown or does not exist.

HWrite

Syntax

HWrite *var_name*, *filename*, *item*

Description

The HWrite command writes a record to a host file and unlocks the file that was previously locked by HReadU or HReadVU.

The HWrite command is a Server Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

The following table describes the parameters of the HWrite command:

Parameter	Description
<i>var_name</i>	The name of a script variable to which HWrite writes data.
<i>filename</i>	The name of the file to write.
<i>item</i>	The name of the item (record) to write.

Example

The following example is part of the PRODM2.WIS demonstration program.

```
Sub OnWrite()  
Library 'wintsys\lib\server'  
am = char(254)  
r_prod = ProdM.Name:am:Mid(ProdM.GroupCode,2):am:ProdM.Price  
HWrite r_prod, 'WIN.PROD', ProdM.ProdRef  
If Server_Error Then  
  MessageBox ServerErrMsg(Server_Error)  
Else  
  ClearEntry  
EndIf  
EndSub
```

Related Script Commands

[HRead](#), [HWriteU](#), [HWriteV](#), [HWriteVU](#)

HWriteU

Syntax

HWriteU *var_name, filename, item*

Description

The HWriteU command writes a record to a host file and leaves the file locked. Use HRelease to release the lock on the item locked by HWriteU.

The HWriteU command is a Server Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

The following table describes the parameters of the HWriteU command:

Parameter	Description
<i>var_name</i>	The name of a script variable to which HWriteU writes data.
<i>filename</i>	The name of the file to write.
<i>item</i>	The name of the item (record) to write

Related Script Commands

[HReadU](#), [HRelease](#), [HWrite](#), [HWriteV](#), [HWriteVU](#)

HWriteV

Syntax

HWriteV *var_name, filename, item, fldno*

Description

The HWriteV command writes a single field to a host file.

The HWriteV command is a Server Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

The following table describes the parameters of the HWriteV command:

Parameter	Description
<i>var_name</i>	The name of a script variable to which HWriteVU writes data.
<i>filename</i>	The name of the file to write to.
<i>item</i>	The name of the item (record) to write to.
<i>fldno</i>	The field number to write to.

Related Script Commands

[HReadV](#), [HRelease](#), [HWrite](#), [HWriteU](#), [HWriteVU](#)

HWriteVU

Syntax

HWriteVU *text, filename, item, fldno*

Description

The HWriteVU command writes a single field to a host file and leaves the file locked. Use HRelease to release the lock the item locked by HWriteVU.

The HWriteVU command is a Server Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

The following table describes the parameters of the HWriteVU command:

Parameter	Description
<i>text</i>	The new text for the field.
<i>filename</i>	The name of the file to write to.
<i>item</i>	The name of the item (record) to write to.
<i>fldno</i>	The field number to write to.

Related Script Commands

[HReadVU](#), [HRelease](#), [HWrite](#), [HWriteU](#), [HWriteV](#)

IconBar

Syntax

IconBar

Description

The IconBar function returns the name of the current iconbar. You can assign a script (like the one in the example) to a function key so the icon bar can be accessed from a key.

Parameters

None

Return Value

The name of the current iconbar

Example

The following example code sets the input focus to the icon bar.

```
*  
barname = IconBar()  
SetFocus $barname
```

IConv

Syntax

IConv *string, format*

Description

The IConv command converts the current date or time into the internal date or time format.

Parameters

The following table describes the parameters of the IConv command:

Parameter	Description
<i>string</i>	The date or time to convert.
<i>format</i>	Specifies how to convert the string. See Below

Values for format

Value	Description
<i>MT</i>	convert to time format.
<i>D</i>	convert to date format. The date order is taken from the Control Panel settings, or you can specify the order. Example: “DDMY” convert the date in day month year order.
<i>MD</i>	Masked decimal. The conversion format is: MDn[f] where n is the number of decimal places and f is the optional scaling factor.
<i>MB</i>	Binary to decimal. The MB conversion converts a string (up to 32 characters) of binary digits into an integer. If the string contains characters other than 1 or 0, this function returns "".

Value	Description
<i>MX</i>	Hexadecimal to decimal. The MX conversion converts a string of (up to 8) hexadecimal digits (0-9, A-F or a-f) into an integer. If the string contains characters which are not hexadecimal digits, this function returns "".
<i>SA<key></i>	This uses a simple algorithm to hide the value of the string and convert it into a format that consists of 7 bit non-control characters. Note: This is not a secure encryption method it is aimed to hide text from a casual viewing. <key> can be any text and is used as part of the scrambling process

Return Value

Returns the converted string or "" if the conversion fails

Example

The following example converts FE to an integer:

```
* Set value to 254
value = Oconv("FE", "MX")
```

The following example converts the time to internal format, then prints it to a message box.

```
MessageBox "The internal time is " : Iconv("08:30:48","MT")
```

Related Script Commands

[Date](#), [Oconv](#), [Stamp](#), [Time](#)

Version

4.2.1 New format "SA"

If

Syntax

If *expr* **Then**

Description

The If/Then/Else/EndIf command executes one of two blocks of statements based on a conditional expression. If expression is true, the program executes the first group of statements. If the expression is false, the program executes the second group of statements.

Remember the following points when writing If/Then/Else/EndIf statements:

The Else clause is optional.

An If/Then/Else/EndIf structure may occupy either a single line, or several lines.

Remember the following points when coding single- and multi-line statements:

The single-line form does not require an EndIf before the Else.

In the multi-line form, each clause must contain at least one statement.

Parameters

The following table describes the parameters of the If command:

Parameter	Description
<i>expression</i>	An expression that evaluate to true or false.
<i>Then</i>	The word "Then". This is an mandatory part of the statement.

Example

The following example is taken from the COPYMENU.WIS demonstration program.

```
* Copy Text to Notepad
Sub CopyTextNotepad()
```

```
If Not(IsApp("notepad.exe")) Then
Dialog RunProgram
Set Filename = 'notepad.exe'
Set Arguments = ''
Invoke
EndIf
title_text = "Untitled - Notepad"
If IsTask(title_text) Then
Invoke EditCopy
Activate title_text
SendKeys "^V" ;* Paste key for notepad
Else
MessageBox "This example copies to ":title_text:" only",
"Copy To Notepad"
EndIf
EndSub
```

Related Script Commands

[And, Or](#)

ImageList Add

Syntax

ImageList Add *name, filename, [mask], [flags], [index_var]*

Synonyms

IMAGELIST AD

Description

The ImageList Add command adds a bitmap image to an image list.

Parameters

The following table describes the parameters of the ImageList Add command:

Parameter	Description
<i>name</i>	Image list name.
<i>filename</i>	Name of bitmap file to add.
<i>mask</i>	Color to use as mask (use "" for no mask.).
<i>flags</i>	See below.
<i>index_var</i>	A variable which will be set with the image index number of the new image or -1 if unsuccessful.

Values for flags

This can be one or more of the following added together

Value	Description
<i>1</i>	Load image as a monochrome image.
<i>32</i>	Map first pixel to Window background.
<i>64</i>	Use system default size.
<i>4096</i>	Map button edge colors.

Related Script Commands

[ImageList AddIcon](#), [ImageList Delete](#), [ImageList GetIconSize](#), [ImageList GetImageCount](#), [ImageList GetImageInfo](#), [ImageList Load](#), [ImageList New](#), [ImageList Remove](#)

ImageList AddIcon

Syntax

ImageList AddIcon *name, filename, [index_var]*

Synonyms

IMAGELIST AI

Description

The ImageList AddIcon command adds an image from an icon file to an imagelist.

Parameters

The following table describes the parameters of the ImageList AddIcon command:

Parameter	Description
<i>name</i>	Image list name.
<i>filename</i>	Name of the bitmap file to add.
<i>index_var</i>	A variable which will be set with the image index number of the new image or -1 if unsuccessful.

Related Script Commands

[ImageList Add](#), [ImageList Delete](#), [ImageList GetIconSize](#), [ImageList GetImageCount](#), [ImageList GetImageInfo](#), [ImageList Load](#), [ImageList New](#), [ImageList Remove](#)

ImageList Delete

Syntax

ImageList Delete *name*

Synonyms

IMAGELIST DL

Description

The ImageList Delete command deletes an imagelist.

Parameters

The following table describes the parameters of the ImageList Delete command:

Parameter	Description
<i>name</i>	The name of the imagelist to delete.

Related Script Commands

[ImageList Add](#), [ImageList AddIcon](#), [ImageList GetIconSize](#), [ImageList GetImageCount](#), [ImageList GetImageInfo](#), [ImageList Load](#), [ImageList New](#), [ImageList Remove](#)

ImageList GetIconSize

Syntax

ImageList GetIconSize *name, width_var, depth_var*

Synonyms

IMAGELIST GS

Description

The ImageList GetIconSize command retrieve the width and depth (height) of the images stored in the image list.

Parameters

The following table describes the parameters of the ImageList GetIconSize command:

Parameter	Description
<i>name</i>	Image list name.
<i>width_var</i>	Variable to set with the width.
<i>depth_var</i>	Variable to set with the depth or height.

Related Script Commands

[ImageList Add](#), [ImageList AddIcon](#), [ImageList Delete](#), [ImageList GetImageCount](#), [ImageList GetImageInfo](#), [ImageList Load](#), [ImageList New](#), [ImageList Remove](#)

ImageList GetImageCount

Syntax

ImageList GetImageCount *name, count_var*

Synonyms

IMAGELIST GC

Description

The ImageList GetImageCount command returns the number of images in an imagelist.

Parameters

The following table describes the parameters of the ImageList GetImageCount command:

Parameter	Description
<i>name</i>	Image list name.
<i>count_var</i>	Variable to set with the result.

Related Script Commands

[ImageList Add](#), [ImageList AddIcon](#), [ImageList Delete](#), [ImageList GetIconSize](#), [ImageList GetImageInfo](#), [ImageList Load](#), [ImageList New](#), [ImageList Remove](#)

ImageList GetImageInfo

Syntax

ImageList GetImageInfo *name, image_no, planes_var, bpp_var*

Synonyms

IMAGELIST GI

Description

The ImageList GetImageInfo command retrieves information on an image.

Parameters

The following table describes the parameters of the ImageList GetImageInfo command:

Parameter	Description
<i>name</i>	Name of the image list.
<i>image_no</i>	Image index to retrieve information.
<i>planes_var</i>	Variable to set with the number of planes in the bitmap.
<i>bpp_var</i>	Variable to set with the bits per pixel in the bitmap.

Related Script Commands

[ImageList Add](#), [ImageList AddIcon](#), [ImageList Delete](#), [ImageList GetIconSize](#), [ImageList GetImageCount](#), [ImageList Load](#), [ImageList New](#), [ImageList Remove](#)

ImageList Load

Syntax

ImageList Load *name*, *filename*, [*width*], [*max_images*], [*mask_color*], [*type*], [*flags*], [*ok_var*]

Synonyms

IMAGELIST LD

Description

The ImageList Load command loads an image list from a bitmap or icon.

Parameters

The following table describes the parameters of the ImageList Load command:

Parameter	Description
<i>name</i>	The handle for the image list.
<i>filename</i>	File name to load.
<i>width</i>	Width of each image (default 0)
<i>max_images</i>	Maximum number of images that the list can contain (the default is 0).
<i>mask_color</i>	Color for transparency mask (the default is -1). See the following Colors table for more information.
<i>type</i>	The type of image: 0 - Bitmap (the default), 1 - Icon, 2 - Cursor.
<i>flags</i>	See below
<i>ok_var</i>	A variable which will be set with True if successful or False if unsuccessful.

Values for flags

Can be one or more of the following added together:

Value	Description
<i>1</i>	Load image as a monochrome image.
<i>32</i>	Map first pixel to Window background.
<i>64</i>	Use system default size.
<i>4096</i>	Map button edge colors.

Related Script Commands

[ImageList Add](#), [ImageList AddIcon](#), [ImageList Delete](#), [ImageList GetIconSize](#), [ImageList GetImageCount](#), [ImageList GetImageInfo](#), [ImageList New](#), [ImageList Remove](#)

ImageList New

Syntax

ImageList New *name, width, depth, [colors], [mask], [initial], [grow], [ok_var]*

Synonyms

IMAGELIST NW

Description

The ImageList New command creates a new image list with no images.

Parameters

The following table describes the parameters of the ImageList New command:

Parameter	Description
<i>name</i>	The handle to the image list
<i>width</i>	Width of each image
<i>depth</i>	Depth or height of each image.
<i>colors</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>mask</i>	Image list mask.
<i>initial</i>	Initial size of the image list (default 4).
<i>grow</i>	Maximum size of the image list (the default is 0).
<i>ok_var</i>	A variable which will be set with "True" if successful or "False" if unsuccessful.

Related Script Commands

[ImageList Add](#), [ImageList AddIcon](#), [ImageList Delete](#), [ImageList GetIconSize](#), [ImageList GetImageCount](#), [ImageList GetImageInfo](#), [ImageList Load](#), [ImageList Remove](#)

ImageList Remove

Syntax

ImageList Remove *name, index_number*

Synonyms

IMAGELIST RM

Description

The ImageList Remove command deletes an image from an image list.

Parameters

The following table describes the parameters of the ImageList Remove command:

Parameter	Description
<i>name</i>	Image list name.
<i>index_number</i>	Index of image to delete.

Related Script Commands

[ImageList Add](#), [ImageList AddIcon](#), [ImageList Delete](#), [ImageList GetIconSize](#), [ImageList GetImageCount](#), [ImageList GetImageInfo](#), [ImageList Load](#), [ImageList New](#)

Index

Syntax

Index *string, sub_string, [occurrence]*

Description

The Index function returns the character position of a substring in a string.

Parameters

The following table describes the parameters of the Index command:

Parameter	Description
<i>string</i>	The string or text to search.
<i>sub_string</i>	The text to find in the string.
<i>occurrence</i>	The occurrences of sub_string to be located. The default is 1.

Return Value

The position of the occurrence of string or 0 if not found

Example

The following example is from the C:\winteg\wintsys\script\LOGON.WIS script.

```
*** LearnInfo event subroutines ***
Sub BrowseForFile()
fname = LearnInfo.FileName
If fname = "" Then fname = AppDir:"*.wis"
If Index(fname, ".",1) = 0 Then fname := ".wis"
If Mid(fname,2,1) # ":" Then fname = AppDir:fname
newname = FileBrowse("Filename to save the learned script", fname, 1,
"Scripts (*.wis)|*.wis")
If newname # "" Then LearnInfo.filename = newname
EndSub
```


Related Script Commands

[Field](#), [InString](#)

InfoBox

Syntax

InfoBox *text*, [*position*]

Synonyms

IB

Description

The InfoBox command displays the "Host Information" pop-up window with the text you specify. If an info box is already displayed, the text changes. When the text parameter is null, the info box is closed.

Parameters

The following table describes the parameters of the InfoBox command:

Parameter	Description
<i>text</i>	The text for the info box to display.
<i>position</i>	The position for the info box: T - Displays at the top of the screen. C - Displays at the center of the screen (the default). B - Displays at the bottom of the screen.

Example

The following example displays an info box for ten seconds, then closes it by using InfoBox with the text parameter as null.

```
* Show the install info box, then close it
InfoBox "Program is installing", B
Wait Delay "10s"
InfoBox ""
```

Related Script Commands

[MessageBox](#)

Ins

Syntax

Ins *var_name*, *line*, [*tab*], *string*

Description

The Ins command inserts a string into a variable at a specified location.

Parameters

The following table describes the parameters of the Ins command:

Parameter	Description
<i>var_name</i>	The variable name where the string will be inserted.
<i>line</i>	The line where the string will be inserted. If line is -1, the string appends to the variable.
<i>tab</i>	The tab number where the string will be inserted. If tab is -1, the string appends to the specified line. If tab is omitted, the entire line is replaced.
<i>string</i>	The string to insert into the variable.

Example

The following example is part of the AP.WIS demonstration program.

```
* Update listbox with current keys
Sub NormalKeys()
If ap_normal = "" Then
ap_normal = "J,Left Arrow,Cursor left"
Ins ap_normal,-1, "K,Right Arrow,Cursor right"
Ins ap_normal,-1, "B,Up Arrow,Cursor Up"
Ins ap_normal,-1, "N,Down Arrow,Cursor Down"
```

Related Script Commands

[Del](#), [Extract](#), [Rep](#)

InString

Syntax

InString *string*, *sub_string*, [*start_position*]

Description

The InString function returns the character position of the first occurrence of a of substring in a string.

Parameters

The following table describes the parameters of the InString command:

Parameter	Description
<i>string</i>	The text string in which to search for the sub string.
<i>sub_string</i>	The text to find in the string.
<i>start_position</i>	The character position from which to start the search. The default is 1.

Return Value

The position the sub string was found or 0

Example

The following example returns 10 since the "e" is in the tenth position. Notice how the first "e" is not returned since it was specified to start searching from the seventh character.

```
* Set the variable position to 10
pos = InString("wIntegrate", "e",7)
```

Related Script Commands

[Configure](#), [Get](#), [Set](#), [Show](#), [Store](#)

Invoke

Syntax

Invoke *name*

Synonyms

IN

Description

The Invoke command executes a menu command. You must set the local variables for the menu commands with the Set command.

See the "Reference - Script Menu Options" section of this manual for information on the Menu Options available.

Parameters

The following table describes the parameters of the Invoke command:

Parameter	Description
<i>name</i>	The name of the menu command to execute.

Example

In the following example, the program segment invokes the Copy command from the edit menu.

```
This example is part of the COPYMENU.WIS demonstration script.  
If IsTask(title_text) Then  
  Invoke EditCopy
```

The next example is the ED_AEBAT.WIS demonstration program.

```
* Demonstration script to edit the autoexec.bat file  
* Version 2.0  
Dialog FileEdit  
Set Filename = "c:\autoexec.bat"  
Invoke
```

Related Script Commands

[Configure](#), [Get](#), [Set](#), [Store](#), [Update](#), [Show](#)

IsApp

Syntax

IsApp *app_name*

Description

The IsApp function checks if an application is running.

Parameters

The following table describes the parameters of the IsApp command:

Parameter	Description
<i>app_name</i>	The name of the application's executable.

Return Value

True if the application is running, otherwise false

Example

The following example is taken from the COPYMENU.WIS demonstration program.

```
* Copy Text to Notepad
Sub CopyTextNotepad()
If Not(IsApp("notepad.exe")) Then
Dialog RunProgram
Set Filename = 'notepad.exe'
Set Arguments = ''
Invoke
EndIf
```

Related Script Commands

[IsTask](#)

IsCommandBar

Syntax

IsCommandBar *name*

Description

This function returns true if *name* is a command bar currently in memory.

Parameters

The following table describes the parameters of the IsCommandBar command:

Parameter	Description
<i>name</i>	The name of the command bar

Return Value

True if the command bar exists, otherwise false

Version

4.0 Original

IsCommandBarShown

Syntax

IsCommandBarShown *name*

Description

This function returns true if *name* is a command bar that currently has been shown.

Parameters

The following table describes the parameters of the IsCommandBarShown command:

Parameter	Description
<i>name</i>	The name of the command bar

Return Value

True if the command bar is shown, otherwise false

Version

4.0 Original

IsControl

Syntax

IsControl *dlg_name, ctrl_name*

Description

This function is used to check if a control has been created on a dialog box.

Parameters

The following table describes the parameters of the IsControl command:

Parameter	Description
<i>dlg_name</i>	The name of the dialog box.
<i>ctrl_name</i>	The name of the control

Return Value

True if the dialog box and control have been defined otherwise false.

Example

Check if keystrip dialog box has a control F16

```
If IsControl(KeyStrip, F16) Then KeyStrip.F16 = label
```

Related Script Commands

[IsDialog](#)

Version

5.1 Original version

IsDialog

Syntax

IsDialog *name*

Description

The IsDialog function checks if a dialog box exists.

Parameters

The following table describes the parameters of the IsDialog command:

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

Return Value

True if the dialog is in memory, otherwise false

Example

The following example is part of the AP.WIS demonstration program.

```
* Advanced Pick function keys help
* Version 2.0
If Not(IsShown(AP_Keys)) Then
If Not(IsDialog(AP_Keys)) Then CreateAPKeys
DialogBox Window AP_Keys
EndIf
EndScript
```

Related Script Commands

[IsShown](#), [IsControl](#)

IsEvent

Syntax

IsEvent *type, text*

Description

The IsEvent function returns "True" if a specified event exists or "False" if it does not exist.

Parameters

The following table describes the parameters of the IsEvent command:

Parameter	Description
<i>type</i>	The event type. This can be any of the events created with the Event command and is specified as their name or abbreviation: • OnHostText or HT • OnExit or OE • OnRestore or OR • OnPortClose or OP • OnPortOpen or OO (wIntegrate 98 only)
<i>text</i>	Event text (for type = OnHostText only).

Return Value

True if event has been defined otherwise false

Example

In the following example, the event is deleted if it exists:

```
If IsEvent(OnHostText, ":") Then Event Delete OnHostText ":"
```

Related Script Commands

[Event Delete](#), [Event OnExit](#), [Event OnHostText](#), [Event OnPortClose](#), [Event OnPortOpen](#), [Event OnRestore](#)

IsFile

Syntax

IsFile *Name*

Description

This function returns true if the *Name* specified is an open file identifier.

Parameters

The following table describes the parameters of the IsFile command:

Parameter	Description
<i>Name</i>	The identifier for this file, specified in File Open.

Return Value

The return value is 1 if the *Name* is an open file identifier otherwise 0.

Example

Close *f_test* if we haven't already

```
If IsFile(f_test) Then File Close f_test
```

Version

4.0.3 Original

IsFullPath

Syntax

IsFullPath *file_name*

Description

This function checks if a string is in the format of a full path name of a file.

Parameters

The following table describes the parameters of the IsFullPath command:

Parameter	Description
<i>file_name</i>	The string to check

Return Value

True (1) if the string specified a full path name

Example

Expand path to the user directory if it is not a full path

```
If Not(IsFullPath(file_name)) Then file_name = UserDir : file_name
```

Related Script Commands

[IsRemotePath](#)

Version

5.0 Original

IsMenu

Syntax

IsMenu *menu*

Description

The IsMenu function returns "True" if a menu exists or "False" if it does not exist.

Parameters

The following table describes the parameters of the IsMenu command:

Parameter	Description
<i>menu</i>	The name of the menu to verify.

Return Value

True if menu exists otherwise false.

Example

Delete Run menu if it exists

```
If IsMenu(Run) Then
Menu Delete Run
EndIf
```

Related Script Commands

[IsMenuItem](#)

IsMenuItem

Syntax

IsMenuItem *menu_item*, [*option*]

Description

The IsMenuItem function returns if a menu item exists. It returns "True" if a menu item exists.

Parameters

The following table describes the parameters of the IsMenuItem command:

Parameter	Description
<i>menu_item</i>	The name of the menu item to verify
<i>option</i>	Options to specify: 0 - Checks only soft menu items loaded by scripts, 1 - Checks also for built-in or extension menu options

Return Value

True if the menu item exists, otherwise False

Example

The following example checks if a menu item exists and adds it if it does not.

```
If Not (IsMenuItem(MyOpt1)) Then
Menu AddItem Edit, MyOpt1,"Copymenu", "script
'example\script\copymenu'"
EndIf
```

The next example attaches a menu after checking for a menu item.

```
If IsMenuItem(RunReceiveFile,1) Then
Menu Attach Run, RunReceiveFile, "&Receive Binary File..."
EndIf
```

Related Script Commands

[IsMenu](#), [IsOnMenu](#)

IsNumber

Syntax

IsNumber *string*

Description

The IsNumber function returns True if string is a number. Numbers are defined as digits zero through nine only.

Parameters

The following table describes the parameters of the IsNumber command:

Parameter	Description
<i>string</i>	The text to check

Return Value

True if the string contains only digits, otherwise false

Example

In the following example, if val is a number then it increments by 10:

```
If IsNumber(val) Then val += 10
```

IsObject

Syntax

IsObject *name*

Description

The IsObject function returns True if an object exists.

Parameters

The following table describes the parameters of the IsObject command:

Parameter	Description
<i>name</i>	Object name

Return Value

Returns true if the object exists

IsOnMenu

Syntax

IsOnMenu *menu, position*

Description

The IsOnMenu function checks if a menu item or a menu item at a specific position exists. It returns "True" if a menu item exists.

Parameters

The following table describes the parameters of the IsOnMenu command:

Parameter	Description
<i>menu</i>	The name of the menu to check.
<i>position</i>	This parameter has two uses. Either the name of the menu item in quotes or the position of the item on the menu.

Return Value

True if the specified item is on the menu, otherwise False

Example

The following example shows the use of the IsOnMenu function:

```
If Not(IsOnMenu(Edit,"MyOpt1")) Then
Menu AddItem Edit, MyOpt1, "Copymenu", "script
'example\script\copymenu'"
EndIf
```

Related Script Commands

[IsMenu](#), [IsMenuItem](#)

IsRemotePath

Syntax

IsRemotePath *file_name*

Description

This function checks if a file name string specifies a path on a remote drive.

Parameters

The following table describes the parameters of the IsRemotePath command:

Parameter	Description
<i>file_name</i>	The string to check

Return Value

True (1) if the file name specifies a remote path otherwise False (0).

Example

Check if the "G" drive is a network drive

```
If IsRemotePath("G:\") Then Print "G is a network drive"
```

Related Script Commands

[IsFullPath](#)

Version

5.0 Original

IsShown

Syntax

IsShown *name*

Description

The IsShown function returns "True" if a dialog box is currently being displayed.

Parameters

The following table describes the parameters of the IsShown command:

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

Return Value

True if the dialog is shown, False otherwise

Example

The following example is taken from the PRODM2.WIS demonstration program.

```
* Check Dialog is not already running
If IsShown(ProdM) Then EndScript
* Create the dialog box
CreateDialog
```

Related Script Commands

[IsDialog](#)

IsTask

Syntax

IsTask *task_name*

Description

The IsTask function returns "True" if a task is running.

Parameters

The following table describes the parameters of the IsTask command:

Parameter	Description
<i>task_name</i>	The name of the task to check. The task name must be exactly as it displays in the Windows task list.

Return Value

True if the task is running, otherwise false

Example

The following example is part of the COPYMENU.WIS demonstration program.

```
title_text = "Untitled - Notepad"
If IsTask(title_text) Then
  Invoke EditCopy
  Activate title_text
  SendKeys "^V" ;* Paste key for notepad
Else
  MessageBox "This example copies to ":title_text:" only", "Copy To
  Notepad"
EndIf
```

Related Script Commands

[Activate](#), [IsApp](#)

IsTrigger

Syntax

IsTrigger *name*

Description

The IsTrigger function returns "True" if a trigger of the specified name exists.

Parameters

The following table describes the parameters of the IsTrigger command:

Parameter	Description
<i>name</i>	The name of the trigger.

Return Value

True if the trigger exists, otherwise false

Example

In the following example, the trigger "errbox" is deleted if it exists:

```
* Delete errbox trigger if it has been defined
If IsTrigger(errbox) Then Trigger Delete errbox
```

Related Script Commands

[Trigger Add](#), [Trigger Delete](#), [Trigger DeleteAll](#)

IsVScreen

Syntax

IsVScreen *name*

Description

This function checks if a virtual screen of the specified name has been defined.

Parameters

The following table describes the parameters of the IsVScreen command:

Parameter	Description
<i>name</i>	The name of the virtual screen

Return Value

This function returns true if the specified virtual screen exists.

Version

4.1.1 Original

Items

Syntax

Items *list_to, filename, selection, output*

Description

The Items command sends the output of fields to a list box.

The Items command is a Server Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

The following table describes the parameters of the Items command:

Parameter	Description
<i>list_to</i>	This parameter is made up of the dialog box name and the list box name, separated by a period, e.g., Dialog.List
<i>filename</i>	The name of the host file to select items from.
<i>selection</i>	A SELECT statement to choose items to list in the list box.
<i>output</i>	Specifies the output from the SELECT statement.

Example

The following example shows a list box within a dialog box, and how the Items command uses the dialog and the list box name.

```
Items Cust.ITEMLIST, "CUSTOMER", 'SELECT CUSTOMER WITH NAME LIKE  
"...S..."', 'NAME'
```

Key ClearAll

Syntax

Key ClearAll

Synonyms

KEY CA

Description

This command clears all down all the soft key definitions.

Parameters

None

Related Script Commands

[Key ClearSet](#)

Key ClearSet

Syntax

Key ClearSet *key_set*

Synonyms

KEY CS

Description

This command clears down the specified group of key definitions.

Parameters

The following table describes the parameters of the Key ClearSet command:

Parameter	Description
<i>key_set</i>	The number of the key set to clear down. See below

Values for key_set

The key set numbers are:

Value	Description
<i>0</i>	Edit Keys (e.g. Arrow keys, Insert, Delete)
<i>1</i>	Function keys (e.g. F1,F2, F11)
<i>2</i>	Mouse buttons. (e.g. MouseLeft, MouseRight)
<i>3</i>	Numeric key pad keys
<i>4</i>	Alpha numeric keys. (e.g. A,B, 1, 2, !, \$)

Related Script Commands

[Key ClearAll](#)

Key LoadEmulation

Syntax

Key LoadEmulation

Synonyms

KEY LE

Description

This command loads the key definitions from the current emulation.

Parameters

None

Key Run

Syntax

Key Run *key_name*

Synonyms

KEY RN

Description

The Key Run command runs a key definition, and is the same as pressing the actual key. If the key definition is a macro, it will run the script.

Parameters

The following table describes the parameters of the Key Run command:

Parameter	Description
<i>key_name</i>	The name of the key to run

Example

The following example runs the F2 key.

```
Key Run "F2"
```

KeyboardLocked

Syntax

KeyboardLocked

Description

This function checks if the keyboard is locked.

Parameters

None

Return Value

True (1) if the keyboard is locked otherwise False (0).

Related Script Commands

[Display Command](#)

Version

5.0 Origianl

KeyIn

Syntax

KeyIn *var_name*, [*timeout*]

Synonyms

KI

Description

This command reads the next key from the keyboard input buffer and sets it into a variable.

The LocalMode On command must be run before this command. The LocalMode command has options that affect the way the keys are stored in the buffer and therefore how they are returned by this command.

Parameters

The following table describes the parameters of the KeyIn command:

Parameter	Description
<i>var_name</i>	The name of the variable in which to store the resulting key stroke
<i>timeout</i>	The time to wait for a key to be pressed. If the timeout is omitted or 0 the KeyIn command will wait forever

Example

The following example shows the name of the key pressed

```
Display FF
Display Text "LocalMode and KeyIn commands"
Display At 3,2
LocalMode On,1,7
```

```
Display Text "Press any key (Escape exits)"
```

```
Loop
KeyIn ch
```

```
Display At 10,10
Display Text ch
Display ClearEOL
Until ch = "Escape"
Repeat

LocalMode off
Display At 0,20
Display Text "Finished"
```

This example collect the character entered until return is pressed or 2 seconds elapse

```
Display FF
Display Text "Enter your name quickly (it won't be echoed) "
LocalMode On
name = ""
Loop
KeyIn ch,"2s"
Until ch = cr Or ch = ""
name := ch
Repeat
If ch = "" Then
MessageBox "Not fast enough"
Else
MessageBox "Hello ":name
EndIf

LocalMode off
```

Related Script Commands

[LocalMode](#)

Version

4.1.1 Original

5.1.1 Added synonym

LastErrorText

Syntax

LastErrorText

Description

The LastErrorText function returns the last error text displayed in the File Transfer Monitor after completion of a File Import, Export, Spool or Bridge Copy. The text returned from this function is valid only immediately after a file transfer has finished and returned "Error" in the ReturnValue variable.

This function is also used to return the error text after the Update command has failed.

Note: The text is to give the user a more meaningful message after a failure. The actual text for each error may change in future releases.

Parameters

None

Return Value

The text of the error message.

Example

The following example script runs an import, then returns a message box if the File Transfer Monitor returns an error.

```
Dialog RunImportFile
Set File = "WIN.ORDER"
Set DosFile="c:\WINTeG\wo.txt"
Invoke
If ReturnValue = "Error" Then
  MessageBox "Import failed - ":LastErrorText()
EndIf
```

Related Script Commands

[Show](#), [Invoke](#), [Update](#)

Version

5.2 Return error text for Update command

Left

Syntax

Left *string*, *number_chars*

Description

The Left function returns the left-most character(s) of a string.

Parameters

The following table describes the parameters of the Left command:

Parameter	Description
<i>string</i>	The string to check for the left-most character(s).
<i>number_chars</i>	The number of characters to return

Return Value

The left hand characters from the string

Example

The following example sets the variable val to "wIn".

```
val = Left("wIntegrate", 3)
```

Related Script Commands

[Mid](#), [Right](#)

Length

Syntax

Length *string*

Description

The Length function returns the number of characters in a string.

Parameters

The following table describes the parameters of the Length command:

Parameter	Description
<i>string</i>	The string to check

Return Value

The number of characters in the string

Example

The following example sets the variable var to 10.

```
var = "wIntegrate"  
num_chars = Length(var)
```

Library

Syntax

Library *filename, option*

Synonyms

LB

Description

The Library command appends another script to the current script. User-created commands and functions in the library can therefore be created once and used in any number of scripts.

Parameters

The following table describes the parameters of the Library command:

Parameter	Description
<i>filename</i>	Filename of library.
<i>option</i>	Describes how the library will be appended. See the following

Values for option

The following table describes the options for the Library command. You can only use the S and U options from within a stored library subroutine and function.

Value	Description
<i>A</i>	Append to current script (the default).
<i>L</i>	Load script into memory. The library script is stored in memory. Future usage of the library will copy the script from memory to the end of the current script.

Value	Description
<i>S</i>	Load subroutines into memory. This loads the library into memory and adds all the subroutines and functions as extensions to the wIntegrate script language. (i.e., the subroutines and functions can be used in the same way as any other built-in script commands.) Future usage of the library command for this file will do nothing, because the subroutines are accessed directly.
<i>M</i>	Cache current script. This is similar to the L option but it stores a copy of the script in which the library statement is run. The filename in this case is used as an identifier and does not relate to anything on disk.
<i>C</i>	Cache subroutines in memory. This is similar to the S option but stores the current script.
<i>U</i>	Unload library and any subroutines. Each time a library is stored, a usage count is increased. The unload option decreases the usage count. When it reaches 0, the library is removed from memory. If any of the routines from the library are in use, the library will not be unloaded completely. The exception is that a subroutine from the library can unload the library which will stop the current script if there is no other library usage. All libraries/sub-routines/functions stored are available in any of the currently running wIntegrate sessions.

LicenseInfo

Syntax

LicenseInfo *Options*

Description

This function returns information on the license this copy of wIntegrate is running with. These details can be seen on the Splash screen and help about dialog box.

Parameters

The following table describes the parameters of the LicenseInfo command:

Parameter	Description
<i>Options</i>	See below

Values for Options

Value	Description
<i>0</i>	User name
<i>1</i>	Organization
<i>2</i>	Serial number
<i>3</i>	User limit

Return Value

The information requested by the options parameter

Version

4.1 Original

Local

Syntax

Local *var_name* [= *value*], ...

Synonyms

LC

Description

The Local command creates up to five local script variables.

These variables exist only until the end of the user subroutine or function. If the variable name matches an existing variable, then the existing variable is not changed.

Up to five local variables can be defined in one local command.

To set global variables (these exist from one script to the next), use the Global command.

Parameters

The following table describes the parameters of the Local command:

Parameter	Description
<i>var_name</i>	The name of the variable to create.
<i>value</i>	The initial value of the variable. If this is omitted, the variable value is null.

Example

The following example comes from the PIECHART.WIS demonstration script.

```
* Sins and Cosines scaled by 1000 looked up from table
Define sinpc(percent)
Local val
If percent <= 25 Then
val = Extract(PCS, percent+1)
Else If percent <= 50 Then
val = Extract(PCS, 51-percent)
```

```
Else If percent <= 75 Then
val = -Extract(PCS, percent-49)
Else
val = -Extract(PCS, 101-percent)
EndIf
return val
```

Related Script Commands

[Define](#), [Sub](#), [Global](#)

LocalMode

Syntax

LocalMode *On/Off*, [*key_coding*], [*soft_key_option*]

Description

The LocalMode On command puts the wIntegrate terminal into a local mode where characters typed are not sent to the host until LocalMode Off is run.

While LocalMode is on the keystrokes can be retrieved by using the EditInput or KeyIn command.

The [*key_coding*] and [*soft_key_option*] parameters control the value retrieved by the KeyIn command. If the [*key_coding*] parameter is set to any other value than the default (0). The keyboard buffer is flushed when the LocalMode on and LocalMode off statements are executed.

Parameters

The following table describes the parameters of the LocalMode command:

Parameter	Description
<i>On/Off</i>	On turns local mode on, Off turn local mode off
<i>key_coding</i>	Specify how the keys are stored in the keyboard buffer for use with the KeyIn command: 0 - The ascii characters of the keys are stored (the default), 1 - The name of the key (as used to define a softkey) is stored.
<i>soft_key_option</i>	See table below.

Values for soft_key_option

The soft key option can be 0 or the following options added together.

Value	Description
<i>1</i>	Intercept the edit keys (e.g. LeftArrow, Insert, Escape)

Value	Description
2	Intercept the function keys (e.g. F1, F2)
4	Intercept the keys from the numeric keypad
255	Intercept all keys (includes all the above options)

Example

See the `KeyIn` command for examples

Related Script Commands

[EditInput](#), [KeyIn](#)

Version

4.1.1 Added `key_coding` and `soft_key_option` parameters

LocalPrint

Syntax

LocalPrint *flag*

Description

This LocalPrint command turns the local (PC) printer on and off.

The LocalPrint command is a Server Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

The following table describes the parameters of the LocalPrint command:

Parameter	Description
<i>flag</i>	"True" for on. "False" for off.

Locate

Syntax

Locate *string*, *sub_string*, [*options*], [*delimiter*], [*ins_pos_var*]

Description

The Locate function returns the field position of a substring within a specified string.

If the sort options are used the searched string must be sorted to match the sort option specified or the substring may not be found even if it is in the string.

Note: The right sorts "AR" and "DR" are a right justified text sort. This means that longer entries are seen as being larger than shorter entries regardless of their contents. To sort numbers, remove any trailing zeros and ensure they have the same number of decimal places and be the same sign.

Parameters

The following table describes the parameters of the Locate command:

Parameter	Description
<i>string</i>	The string to search.
<i>sub_string</i>	The text within the string to find.
<i>options</i>	Options for the locate, see below. The default is "E".
<i>delimiter</i>	The character that is the field delimiter. The default is a carriage return.
<i>ins_pos_var</i>	The name of a variable to hold the position where the substring would need to be inserted into the string. If the sub_string was found in the string this is the same as the returned field position. If no sort options were specified and the substring is not found this will be the number of the field after the last field in the string.

Values for options

The options are specified as a case insensitive string containing the following values.

Value	Description
<i>AL</i>	String is sorted in ascending left order.
<i>AR</i>	String is sorted in ascending right order.
<i>DL</i>	String is sorted in descending left order
<i>DR</i>	String is sorted in descending right order.
<i>E</i>	The entire field must match. This is always set when using the sort options.
<i>F</i>	The first characters of the field matches the substring
<i>U</i>	Convert both strings to upper case before the comparison

Return Value

The number of the field containing the specified sub string or 0 if the sub string is not found

Example

The following example is taken from the WC.WIS demonstration program.

```
Sub RunCmnd()
pos = Locate(DialogList(wc,Cmnd), wc.Cmnd, "EU")
If pos = 0 Then DialogBox AddToList wc,cmnd,wc.cmnd,1
Execute wc.Cmnd
If IsShown(wc) Then DialogBox SetText wc,Cmnd, ''
EndSub
```

Build a sorted string of names

```
name_list = ""
pos = Locate(name_list,"Tim","AL",, ins_pos)
Ins name_list,ins_pos, "Tim"
pos = Locate(name_list,"Alan","AL",, ins_pos)
Ins name_list,ins_pos, "Alan"
```

```
pos = Locate(name_list,"David","AL",, ins_pos)
Ins name_list,ins_pos, "David"

MessageBox name_list, "Sorted List of Names"
```

Related Script Commands

[Field](#)

Version

5.1 Sort options and insert position variable

Loop

Syntax

Loop

Description

The Loop/Repeat command repeats any contained statements while or until a specified condition is met, depending on whether you use the While or Until clause. statements can precede and/or follow the test condition. Repeat is required and finishes the Loop block.

Parameters

None

Example

The following example is from the EFFECT.WIS demonstration program.

```
* Demonstrate wIntegrate's Effects
* Version 2.0
Display FF
Display At 29,0
Display Text "Effects Colour Chart"
x = 0
y = 0
j = 1
Loop
EffectName = ""
if j & 1 Then EffectName = "Dim"
if j & 2 Then EffectName := "Reverse"
if j & 4 Then EffectName := "Underline"
if j & 8 Then EffectName := "Flash"
if j & 16 Then EffectName := "Bold"
if j & 32 Then EffectName := "Secret"
j += 1
y+=1
if y = 22 Then
y = 1
x += 26
EndIf
Display At x+25,y
Display Effect "Normal"
Display At x,y
Display Text Left(EffectName,24)
```

```
Until j = 64  
Repeat  
Display At 0,22  
Display Effect "Normal"
```

Related Script Commands

[And, Or](#)

Mail Address

Syntax

Mail Address [*To*], [*Cc*], [*Bcc*], [*Options*]

Synonyms

ML AD

Description

This command displays a mail system dialog box allowing the user to address a message by selecting address book addresses.

Parameters

The following table describes the parameters of the Mail Address command:

Parameter	Description
<i>To</i>	the name of the variable that will receive the names of the primary recipients
<i>Cc</i>	the name of the variable that will receive the names of the copy recipients
<i>Bcc</i>	the name of the variable that will receive the names of the blind copy recipients
<i>Options</i>	See below

Values for Options

Value	Description
0	(default)
4	Disable mail system user interaction

Example

Allow user to select other recipients before sending message

```
To = "Demi;Jodie"  
Subject = "Gossip!"  
Text = "Psssst.... guess who I saw with Michelle last night?"  
Mail Address To, Cc, Bcc  
Mail Send To, Cc, Bcc, Subject, Text
```

Version

4.0.1 Original

Mail Delete

Syntax

Mail Delete *Id*, [*Options*]

Synonyms

ML DL

Description

This command deletes a mail message.

Parameters

The following table describes the parameters of the Mail Delete command:

Parameter	Description
<i>Id</i>	the ID of the message to delete. This ID will have been retrieved using the MailFindNext or MailFindAll functions.
<i>Options</i>	See below

Values for Options

Value	Description
0	(default)
4	Disable mail system user interaction

Example

Delete all messages from Sabrina

```
Ids = MailFindAll("Sabrina", 16)
If Ids # "" Then
Count = Count(Ids, CR) + 1
i = 1
Loop
```

```
While i &lt;= Count  
  Id = Extract(Ids, i)  
  Mail Delete Id  
  i += 1  
Repeat  
EndIf
```

Related Script Commands

[MailFindNext](#), [MailFindAll](#)

Version

4.0.1 Original

Mail LookUp

Syntax

Mail LookUp [*Name*], [*Address*], [*Options*]

Synonyms

ML LU

Description

This command returns details about a mail recipient given a full or partial name.

Parameters

The following table describes the parameters of the Mail LookUp 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 Name 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>Options</i>	See below

Values for Options

Value	Description
<i>0</i>	(default)
<i>1</i>	Display full details
<i>4</i>	Disable mail system user interaction

Example

Display Sheila's full name and address

```
Name = "Sheila"  
Mail LookUp Name, Address  
MessageBox Name : CR : Address
```

Display Address Book

```
Mail LookUp
```

Display Sandra's full details

```
Name = "Sandra"  
Mail LookUp Name, , 1
```

Check Sally's name before sending a message

```
Name = "Sally"  
Mail LookUp Name  
If Name # "" Then  
Mail Send Name, , , "Greetings!", "How about a drink after work?"  
EndIf
```

Version

4.0.0 Introduced

4.0.1 Changed from function to command and added extra parameters.

Mail Read

Syntax

Mail Read *Id*, [*Sender*], [*Subject*], [*Text*], [*Attachments*], [*To*], [*Cc*], [*Time*], [*State*], [*Options*]

Synonyms

ML RD

Description

This command reads a mail message.

Parameters

The following table describes the parameters of the Mail Read command:

Parameter	Description
<i>Id</i>	the ID of the message to read. This ID will have been retrieved using the MailFindNext or MailFindAll functions.
<i>Sender</i>	the name of the variable that will receive the message sender's name.
<i>Subject</i>	the name of the variable that will receive the title or subject of the message.
<i>Text</i>	the name of the variable that will receive the actual message text.

Parameter	Description
<i>Attachments</i>	the name of the variable that will receive the full path names of any attachment files. Multiple attachment files will be separated by a semi-colon, eg. "C:\Temp\Murder1.txt;C:\Temp\NYPDBlue.txt". On reading a message with attachments, the attachments are copied from the mail system into temporary files. It is good practice to delete these files when no longer needed. If the Attachments parameter is not specified, then no temporary files will be copied.
<i>To</i>	the name of the variable that will receive the names of the primary recipients. Multiple recipients will be separated by a semi-colon, eg. "Rachel;Chandler;Monica".
<i>Cc</i>	the name of the variable that will receive the names of the copy recipients.
<i>Time</i>	the name of the variable that will receive the time of the message.
<i>State</i>	the name of the variable that will receive the message state (see below).
<i>Options</i>	See below

Values for State

Value	Description
<i>0</i>	(default)
<i>1</i>	Don't mark as read
<i>2</i>	Make message text first attachment
<i>4</i>	Disable mail system user interaction

Values for Options

Value	Description
0	Read
1	Unread
2	File(s) attached

Example

Display first unread message

```
Id = MailFindNext("", , 1)
If Id # "" Then
Mail Read Id, , Subject, Text
MessageBox Subject : CR : CR : Text
EndIf
```

Display attachment file from specific message

```
Id = MailFindNext("", "Map of Treasure", 32)
If Id # "" Then
Mail Read Id, , , Attachment
Run Attachment
* We really ought to delete the file after viewing it
Else
MessageBox "Sorry, no treasure!"
EndIf
```

Version

4.0.1 Original

Mail Send

Syntax

Mail Send [*To*], [*Cc*], [*Bcc*], [*Subject*], [*Text*], [*Attachments*], *Options*, *Response*

Synonyms

ML SN

Description

This command sends a mail message.

Parameters

The following table describes the parameters of the Mail Send command:

Parameter	Description
<i>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). Recipient names can be full names, partially unresolved names or full email addresses, eg. "Mary Doe", "Mary", "marydoe@widgets.com", respectively. Multiple recipients may be specified by separating them with a semi-colon, eg. "Denise;Teresa;Marilyn".
<i>Cc</i>	the names of the copy recipients
<i>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>Options</i>	See below
<i>Response</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 Options

Value	Description
0	(default)
1	Review message before sending
4	Disable mail system user interaction (overrides 1)

Example

Tell grunts to attend a meeting

```
Mail Send "Bill;Ben", , , "Meeting at 10:30 today in my Office",  
"Attend or else!"
```

Ask managers to attend a meeting

```
To = "Algernon;Emily;Nigel"  
Cc = "Caterers"  
Subject = "Meeting at 12:00 today at the Swimming Pool Bar"  
Text = "Dear Friends" : CR : CR  
Text := "Would you mind attending this meeting?" : CR  
Text := "Thank you so much." : CR : CR  
Text := "Your friend, Uriah."  
Mail Send To, Cc, , Subject, Text, "c:\Temp\Country Club Map.gif", 1,  
Response  
If Response # "" Then  
  MessageBox Response  
EndIf
```

Version

4.0.0 Introduced

4.0.1 Split original To recipients into To, CC and BCC by adding extra parameters.

MailAvailable

Syntax

MailAvailable [*Response*]

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 MailAvailable command:

Parameter	Description
<i>Response</i>	If an error occurs this parameter will contain text describing what went wrong.

Return Value

The return value is 1 if connection was successful or 0 if not.

Example

Send a message

```
If MailAvailable(Error) Then
Mail Send "Sarah", , , "Rendezvous", "Coffee machine at 10:05?"
Else
MessageBox Error
EndIf
```

Version

4.2.1 Original

MailFindAll

Syntax

MailFindAll *Pattern, Options*

Description

This function finds all available mail messages matching specified criteria. Once the messages have been found they can be read or deleted, using Mail Read or Mail Delete , 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 MailFindAll 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 the Options parameter. If Pattern is not supplied (or no search location is specified in the Options parameter), the function returns the next message meeting the search criteria specified in the Options parameter but without performing any pattern matching. Pattern matching is not case-sensitive.
<i>Options</i>	See below

Values for Options

Options maybe combined by addition.

Value	Description
0	Search all messages (default)
1	Search unread messages only
2	Original order
4	Disable mail system user interaction
8	Reverse order of results
16	Match sender
32	Match subject
64	Match attachment
128	Match time

Return Value

The return value will be a carriage-return (CR) separated list

Example

Count the number of unread messages on the subject of Dilbert

```
Ids = MailFindAll("Dilbert", 33)
MessageBox Count(Ids, CR) + 1
```

List the titles of all unread messages

```
Ids = MailFindAll( , 1)
If Ids # "" Then
Count = Count(Ids, CR) + 1
Subjects = ""
i = 1
Loop
While i <= Count
Id = Extract(Ids, i)
Mail Read Id, , Subject
Subjects := Subject : CR
```

```
i += 1
Repeat
MessageBox Subjects
Else
MessageBox "No unread messages"
EndIf
```

Version

4.0.1 Original

MailFindNext

Syntax

MailFindNext *Id*, [*Pattern*], [*Options*]

Description

This function finds the next available mail message matching specified criteria. It is intended to be used iteratively so that all mail messages maybe enumerated. Once a message has been found it can be read or deleted, using Mail Read or Mail Delete , 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 MailFindNext 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 Options parameter. If Pattern is not supplied (or no search location is specified in the Options parameter), the function returns the next message meeting the search criteria specified in the Options parameter but without performing any pattern matching. Pattern matching is not case-sensitive.
<i>Options</i>	See below

Values for Options

Options maybe combined by addition.

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 system user interaction
<i>16</i>	Match sender
<i>32</i>	Match subject
<i>64</i>	Match attachment
<i>128</i>	Match time

Return Value

The id of the next mail message matching the pattern and options or "" if there are no more matches

Example

Search for, and display, the first unread message from Pamela

```
Id = MailFindNext("", "Pam", 17)
If Id # "" Then
Mail Read Id, , Subject, Text
MessageBox Subject : CR : CR : Text
EndIf
```

Delete the fourth message of June 22nd

```
Id = ""
i = 1
Loop
Id = MailFindNext(Id, "/06/22", 128)
While i < 4 And Id # ""
i += 1
Repeat
If Id # "" Then
Mail Delete Id
EndIf
```

Version

4.0.1 Original

MakeCommandBar

Syntax

MakeCommandBar *name*

Description

The **MakeCommandBar** command takes the toolbar defined between **CreateIconBar** and **EndCreateIconBar** and adds it to the list of toolbars and displays it.

The **MakeCommandBar** command is a **Icon Library** command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

The following table describes the parameters of the **MakeCommandBar** command:

Parameter	Description
<i>name</i>	The name of the toolbar to display

Version

4.0 Original

MapChars

Syntax

MapChars *text*, [*conversion*]

Description

The MapChars function allows characters to be mapped from the host character set to the PC character set and vice versa as specified in the Terminal Emulation. MapChars returns text converted for the current emulation.

Parameters

The following table describes the parameters of the MapChars command:

Parameter	Description
<i>text</i>	Text to be converted
<i>conversion</i>	Specifies which conversion to use. Values are: TRUE (default)- Host-to-PC conversion. FALSE - PC -to-host conversion.

Return Value

The converted text

Match

Syntax

Match *value, pattern*

Description

The Match function compares a string with a pattern and returns True if it matches.

Parameters

The following table describes the parameters of the Match command:

Parameter	Description
<i>value</i>	The text to match.
<i>pattern</i>	The matching pattern. See table below.

Values for pattern

The format of the pattern string made up of repeating the following:
[num]{A|N|X|'lit'}:

Value	Description
<i>num</i>	Number stands for the number of repetitions of the type of character in the type field. If num is not set, it defaults to 1. If it is set to 0, it searches for zero or more occurrences of the type.
<i>A</i>	Alphabetic characters
<i>N</i>	Numeric
<i>X</i>	Any character.
<i>'lit'</i>	You can also use a literal text string inside of quotes.

Return Value

True if the text matches the pattern, otherwise false

Example

The following example is part of the PRODM2.WIS demonstration program.

```
* Validate Price
Sub OnValidatePrice()
If Not(Match(ProdM.Price,"1N0N")) Or ProdM.Price < 1 Then
    MsgBox 'Price must be an integer greater than zero', "Product
Maintenance", MB_ICONHAND | MB_OK, ProdM
    DialogBox InputOK, ProdM, False
Else
    DialogBox InputOK ProdM, True
    If ProdM_Mode < 8 Then CheckWrite 4
EndIf
EndSub
```

Menu AddItem

Syntax

Menu AddItem *menu, item, text, command, [position]*

Synonyms

MN AI

Description

The Menu AddItem command creates a menu item and adds it to a menu. An ampersand (&) in the text makes the letter after it underlined as a shortcut to select the item.

Parameters

The following table describes the parameters of the Menu AddItem command:

Parameter	Description
<i>menu</i>	The name of the menu this item will be attached to.
<i>item</i>	The name you give to this item you are adding.
<i>text</i>	The text to display for this menu item.
<i>command</i>	The script command to run when this item is selected.
<i>position</i>	This is either a menu item name in quotes or the position number of the menu item which follows the added item.

Example

The following example is taken from the COPYMENU.WIS demonstration program.

```
Menu Create CopyMenu
AddItem TextToNotepad, "Text to &Notepad", CopyLib : "CopyTextNotepad"
AddItem TextToWrite, "&Text to Write", CopyLib : "CopyTextWrite"
AddItem BitmapToPbrush, "Bitmap to &Paintbrush", CopyLib :
"CopyBitmapPbrush"
AddItem TableToExcel, "Table to &Excel", CopyLib : "CopyTableExcel"
AddItem TextToWord, "Text To &Word", CopyLib : "CopyTextWord"
```

```
AddItem BitToWord, "&Bitmap To Word", CopyLib : "CopyBitmapWord"  
AddItem ScreenToWord, "&Screen To Word", CopyLib : "CopyScreenWord"  
AddItem ScreenToPrinter, "Screen To P&rinter", CopyLib :  
"CopyScreenPrinter"
```

Related Script Commands

[Menu DeleteItem](#)

Menu Attach

Syntax

Menu Attach *menu*, *attach_item*, [*text*], [*position*]

Synonyms

MN AT

Description

The Menu Attach command attaches an existing menu, menu item or menu command to a menu.

Parameters

The following table describes the parameters of the Menu Attach command:

Parameter	Description
<i>menu</i>	The name of the existing menu.
<i>attach_item</i>	The name of the item to attach. See table below.
<i>text</i>	Text to display on the menu, representing the attached item.
<i>position</i>	The position for the attached item. The new item is attached before the menu item name or item number that you specify here. The default position is the end of the menu.

Values for attach_item

The item to attach to the menu is one of the following:

Value	Description
<i>sub menu</i>	The name of a sub menu to attach.
<i>item</i>	The name of the menu item to attach.

Value	Description
<i>menu option</i>	The name of a script menu option to attach. See the "Reference - Script Menu Options" section of this manual for the menu option names.

Example

Attach the standard File Open option to a custom host menu

```
Menu Attach HostMenu, FileOpen, "Open File", "custmaint"
```

Related Script Commands

[Menu Detach](#)

Menu Command

Syntax

Menu Command *menu_name, menu_item, command*

Synonyms

MN CM

Description

The Menu Command command changes the action of a menu item when it is clicked on.

Parameters

The following table describes the parameters of the Menu Command command:

Parameter	Description
<i>menu_name</i>	The name of the menu where the item exists.
<i>menu_item</i>	The name of the menu item to change.
<i>command</i>	The new command for the item.

Related Script Commands

[Menu AddItem](#), [Menu DeleteItem](#)

Menu Create

Syntax

Menu Create *menu*

Synonyms

MN CR

Description

The Menu Create command creates an empty menu and puts the script into menu creation mode. You must use the Menu EndCreate command to end creation mode.

Parameters

The following table describes the parameters of the Menu Create command:

Parameter	Description
<i>menu</i>	The name of the new menu to create.

Example

The following example is taken from the COPYMENU.WIS demonstration program. It shows the code from the creation of the menu to the completion of the menu using EndCreate.

```
Menu Create CopyMenu
AddItem TextToNotepad, "Text to &Notepad", CopyLib : "CopyTextNotepad"
AddItem TextToWrite, "&Text to Write", CopyLib : "CopyTextWrite"
AddItem BitmapToPbrush, "Bitmap to &Paintbrush", CopyLib :
"CopyBitmapPbrush"
AddItem TableToExcel, "Table to &Excel", CopyLib : "CopyTableExcel"
AddItem TextToWord, "Text To &Word", CopyLib : "CopyTextWord"
AddItem BitToWord, "&Bitmap To Word", CopyLib : "CopyBitmapWord"
AddItem ScreenToWord, "&Screen To Word", CopyLib : "CopyScreenWord"
AddItem ScreenToPrinter, "Screen To P&rinter", CopyLib :
"CopyScreenPrinter"
AddItem PieChart, "Create Pie &Chart", "Script 'exam
ple\script\parse';Script 'example\script\piechart'"
AddItem BarChart, "Create B&ar Chart", "Script 'exam
ple\script\parse';Script 'example\script\barchart'"
EndCreate
```

Related Script Commands

[Menu EndCreate](#)

Menu Delete

Syntax

Menu Delete *menu*, *delete_sub_menus*

Synonyms

MN DL

Description

The Menu Delete command deletes a menu. If the menu is attached to any other menu, it must be detached first with the Menu Detach command. You must use Menu Delete to delete the menu from the PC memory.

If the delete sub menus flag is set to true any sub menus of this menu will be deleted if they are only attached to this menu.

Parameters

The following table describes the parameters of the Menu Delete command:

Parameter	Description
<i>menu</i>	The name of the menu to delete.
<i>delete_sub_menus</i>	True to recursively delete menu attached to this menu. Default is False

Example

The following example is code from the COPYMENU.WIS demonstration program.

```
Menu Create CopyMenu
.
.
.
EndCreate
Menu Popup CopyMenu, Mouse(Get_X), Mouse(Get_Y)
Menu Delete CopyMenu
EndScript
```

Related Script Commands

[Menu Create](#), [Menu Detach](#)

Version

4.1.0 Added delete sub menus optional flag

Menu DeleteItem

Syntax

Menu DeleteItem *menu, position*

Synonyms

MN DI

Description

The Menu DeleteItem command deletes a menu item, detaches a submenu or a wIntegrate command.

Parameters

The following table describes the parameters of the Menu DeleteItem command:

Parameter	Description
<i>menu</i>	The name of the menu from which to delete an item.
<i>position</i>	The position of the item or the item name in quotes.

Example

The following example deletes the "logoff" item from the menu.

```
Menu DeleteItem HostMenu, "logoff"
```

Related Script Commands

[Menu AddItem](#)

Menu Detach

Syntax

Menu Detach *menu, position*

Synonyms

MN DT

Description

The Menu Detach command detaches a menu item, submenu or wIntegrate command.

Parameters

The following table describes the parameters of the Menu Detach command:

Parameter	Description
<i>menu</i>	The name of the menu from which to detach an item.
<i>position</i>	The position of the item or the item name in quotes.

Example

The following example detaches an item from the menu.

```
Menu Detach HostMenu, "logoff"
```

Related Script Commands

[Menu Attach](#)

Menu Enable

Syntax

Menu Enable *menu, position, flag*

Synonyms

MN EN

Description

The Menu Enable command enables or disables a menu position.

Parameters

The following table describes the parameters of the Menu Enable command:

Parameter	Description
<i>menu</i>	The name of the menu.
<i>position</i>	The position of the item or the item name in quotes.
<i>flag</i>	Can be one of the following: "True" - Enables a menu item. "False" - Disables a menu item.

Example

The following example disables the "Login" item on a menu named "HostMenu":

```
Menu Enable HostMenu, "Login", False
```

Related Script Commands

[Menu Create](#), [MenuEnabled](#)

Menu EndCreate

Syntax

Menu EndCreate *menu*

Synonyms

MN EC

Description

The Menu EndCreate command stops the creation of the current menu. Start creation of the menu script with the Menu Create command. You can use just "EndCreate" to end creation mode.

Parameters

The following table describes the parameters of the Menu EndCreate command:

Parameter	Description
<i>menu</i>	The name of the menu created with Menu Create.

Example

The following example is taken from the COPYMENU.WIS demonstration program.

```
Menu Create CopyMenu
AddItem TextToNotepad, "Text to &Notepad", CopyLib : "CopyTextNotepad"
.
.
.
AddItem BarChart, "Create B&ar Chart", "Script
'example\script\parse';Script
'example\script\barchart'"
EndCreate
```

Related Script Commands

[Menu Create](#)

Menu Help

Syntax

Menu Help *menu*, {*submenu* / *item*}, *help_text*

Synonyms

MN HL

Description

The Menu Help command adds a line of help text on the status bar when the menu option is selected. The menu items on the wIntegrate main menu show help on the status line.

Parameters

The following table describes the parameters of the Menu Help command:

Parameter	Description
<i>menu</i>	The name of the menu.
{ <i>submenu</i> / <i>item</i> }	The name of the sub menu or item to explain in the help text.
<i>help_text</i>	The text to appear in the status bar.

Example

The following example displays information in the status bar when the Logon item is clicked on:

```
Menu Help HostMenu, Logon, "Brings up login box"
```

Menu Main

Syntax

Menu Main {*menu* / *Default*}

Synonyms

MN MN

Description

The Menu Main command replaces the main menu with a specified menu.

Use the "Default" parameter to clear all current menus and display the default wintegrate menus.

Note: The default wIntegrate menus do not include the file transfer items under the Run menu. To add these to the default main menu, use the line "Script 'wintsys\script\newsess'".

Parameters

The following table describes the parameters of the Menu Main command:

Parameter	Description
<i>{menu / Default}</i>	The name of the new main menu. The word Default clears all current menus and displays the default wintegrate menus.

Example

Menu Main MyMenu

* Reset menus

Menu Main Default

The next example adds file transfer functionality to the default wIntegrate menus.

* Add file transfer routines to the menus

Script "wintsys\menu\ftmenu"

Related Script Commands

[Menu AddItem](#), [Menu Attach](#), [Menu Create](#)

Menu New

Syntax

Menu New *menu*

Synonyms

MN NW

Description

The Menu New command creates an empty menu.

Tip: Use Menu Create to create a full menu.

Parameters

The following table describes the parameters of the Menu New command:

Parameter	Description
<i>menu</i>	The name of the new menu.

Example

The following example creates a new menu called "HostMenu":

```
Menu New HostMenu
```

Related Script Commands

[Menu Create](#)

Menu OnClose

Syntax

Menu OnClose *menu, command*

Synonyms

MN OC

Description

The Menu OnClose command runs a script command when the menu closes. Use it to inform the host when a menu is closed. This command can only be used when the complete menu tree is closed.

Parameters

The following table describes the parameters of the Menu OnClose command:

Parameter	Description
<i>menu</i>	The name of the menu that is closed.
<i>command</i>	The script command to run when the menu closes.

Example

The following example prints a message on the wIntegrate screen when the menu is closed:

```
Menu OnClose HostMenu, "Enter 'HostMenu is closed'"
```

Related Script Commands

[Menu OnOpen](#)

Menu OnOpen

Syntax

Menu OnOpen *menu, command*

Synonyms

MN OO

Description

The Menu OnOpen command runs a command when the menu opens. Use it to initialize a menu that has options which can be grayed or checked.

Parameters

The following table describes the parameters of the Menu OnOpen command:

Parameter	Description
<i>menu</i>	The name of the menu that is opened.
<i>command</i>	The script command to run when the menu opens.

Example

The following example enables Bridge Copy if there is more than one session open.

```
Menu OnOpen Run "Menu Enable Run, RunBridgeCopy"  
(Sessions(0) > 1) "
```

Related Script Commands

[Menu OnClose](#)

Menu PopUp

Syntax

Menu PopUp *menu*, *x*, *y*, [*flag*]

Synonyms

MN PU

Description

The Menu PopUp command creates a pop-up menu which disappears with a mouse click outside the menu area. You must create the menu with Menu Create before you can make it a popup menu.

Parameters

The following table describes the parameters of the Menu PopUp command:

Parameter	Description
<i>menu</i>	The name of the popup menu.
<i>x</i>	The column position to start the popup menu.
<i>y</i>	The row position to start the popup menu.
<i>flag</i>	The flags for Menu PopUp are: 2 - Allows the right mouse button to select menu items. 4 - Centers the menu around the x position. 8 - Right-aligns the menu to the x position.

Example

The following example comes from the COPYMENU.WIS demonstration program. Notice that

```
Menu PopUp is used after the menu is created.
Menu Create CopyMenu
AddItem TextToNotepad, "Text to &Notepad", CopyLib : "CopyText
Notepad"
AddItem TextToWrite, "&Text to Write", CopyLib : "CopyTextWrite"
AddItem BitmapToPbrush, "Bitmap to &Paintbrush", CopyLib :
"CopyBitmapPbrush"
AddItem TableToExcel, "Table to &Excel", CopyLib : "CopyTableEx
```

```
cel"  
AddItem TextToWord, "Text To &Word", CopyLib : "CopyTextWord"  
AddItem BitToWord, "&Bitmap To Word", CopyLib : "CopyBitmapWord"  
AddItem ScreenToWord, "&Screen To Word", CopyLib : "CopyScreen  
Word"  
AddItem ScreenToPrinter, "Screen To P&rinter", CopyLib : "Copy  
ScreenPrinter"  
AddItem PieChart, "Create Pie &Chart", "Script 'exam  
ple\script\parse';Script 'example\script\piechart'"  
AddItem BarChart, "Create B&ar Chart", "Script 'exam  
ple\script\parse';Script 'example\script\barchart'"  
EndCreate  
Menu Popup CopyMenu, Mouse(Get_X), Mouse(Get_Y)
```

Related Script Commands

[Menu Create](#)

Menu Separator

Syntax

Menu Separator *menu*, [*position*]

Synonyms

MN SP

Description

The Menu Separator command adds a horizontal bar to a menu.

Parameters

The following table describes the parameters of the Menu Separator command:

Parameter	Description
<i>menu</i>	The name of the menu.
<i>position</i>	The position of the item or the item name in quotes. Default is the end of the menu.

Example

The following example adds a horizontal line to the COPYMENU.WIS demonstration program.

```
The line appears after the first menu item, "Text to Notepad".
Menu Create CopyMenu
AddItem TextToNotepad, "Text to &Notepad", CopyLib : "CopyTextNotepad"
Menu Separator 2
```

Menu SetCheck

Syntax

Menu SetCheck *menu, position, flag*

Synonyms

MN SC

Description

The Menu SetCheck command sets or removes a menu item check mark. To verify whether a menu item is checked, use the MenuChecked function.

Parameters

The following table describes the parameters of the Menu SetCheck command:

Parameter	Description
<i>menu</i>	The name of the menu.
<i>position</i>	The position of the item or the item name in quotes.
<i>flag</i>	Can be one of the following: True - Sets the check, False - Clears a check.

Example

The following example demonstrates the use of the SetCheck command:

```
Menu SetCheck HostMenu, "EditMode", True
```

Related Script Commands

[MenuChecked](#)

MenuChecked

Syntax

MenuChecked *menu, position*

Description

The MenuChecked function verifies whether a menu item is checked and returns "True" if it is checked. To set a menu item as checked or unchecked, use the Menu SetCheck command.

Parameters

The following table describes the parameters of the MenuChecked command:

Parameter	Description
<i>menu</i>	The name of the menu where the item exists.
<i>position</i>	The position of the item you want to verify. The position can be the menu item name or item number.

Return Value

True if the menu is checked, otherwise false

Example

The following example checks if a script is recording and stops it if "True":

```
* Check if script is recording and stop recording if "true"
If MenuChecked(Edit, "EditRecord") Then
Invoke EditRecord
EndIf
```

Related Script Commands

[Menu SetCheck](#)

MenuEnabled

Syntax

MenuEnabled *menu, position*

Description

The MenuEnabled function checks if a menu item is enabled, and returns "True" if the menu item is enabled.

Parameters

The following table describes the parameters of the MenuEnabled command:

Parameter	Description
<i>menu</i>	The name of the menu to check.
<i>position</i>	The position of the item or the item name in quotes.

Return Value

True if the menu is enabled, otherwise false

Example

The following example prints a message on the status bar if the Run Script menu item is enabled:

```
If MenuEnabled(Run, "RunScript") Then  
Print "Run Script enabled"
```

Related Script Commands

[MenuEnabled](#)

MenuGUI Active

Syntax

MenuGUI Active *menu_option*, *active*

Synonyms

MG AC

Description

This command tells the application active state of a current GUI for a menu option.

Parameters

The following table describes the parameters of the MenuGUI Active command:

Parameter	Description
<i>menu_option</i>	The name of the menu option.
<i>active</i>	True to inform the application the GUI is active otherwise False.

Version

5.0 Original

MenuGUI Command

Syntax

MenuGUI Command *menu_option, command, [arg1...argN]*

Synonyms

MG CM

Description

This command runs commands that are used by the GUI for a specific menu option.

Parameters

The following table describes the parameters of the MenuGUI Command command:

Parameter	Description
<i>menu_option</i>	The name of the menu option
<i>command</i>	The name of the command to run
<i>arg1...argN</i>	Parameters required for the command

Version

5.0 Original

MenuGUI DefaultMap

Syntax

MenuGUI DefaultMap *menu_option*, *set_flag*, [*get_flag*]

Synonyms

MG DM

Description

This command specifies the default mapping between the session variables and the representation in the GUI for the menu. Any variables that are not specifically mapped with the MenuGUI MapVariable command use this mapping.

Parameters

The following table describes the parameters of the MenuGUI DefaultMap command:

Parameter	Description
<i>menu_option</i>	The name of the menu option.
<i>set_flag</i>	A number specifying the required mapping. See below.
<i>get_flag</i>	This flag has the same meaning as the set flag. If it is omitted it default to the same value as the set flag.

Values for set_flag

Value	Description
<i>0</i>	No mapping
<i>1</i>	Map to the default attribute of the control with the same name as the variable in the dialog box that has the same name as the menu option

Value	Description
2	Call a script function to set or get the variable

Related Script Commands

[MenuGUI MapVariable](#), [MenuGUI Initialise](#), [MenuGUI Update](#)

Version

5.0 Original

MenuGUI GetScript

Syntax

MenuGUI GetScript *menu_option*, *var_name*, [*from_memory*]

Synonyms

MG GS

Description

This command returns a script that can be used to set the variables for the specified menu option.

Parameters

The following table describes the parameters of the MenuGUI GetScript command:

Parameter	Description
<i>menu_option</i>	The name of the menu option
<i>var_name</i>	The name of the variable to receive the script.
<i>from_memory</i>	True to return a script of the current settings stored in the application. False to return a script of the settings from the menu gui.

Related Script Commands

[MenuGUI Save](#)

Version

5.0

MenuGUI Initialise

Syntax

MenuGUI Initialise *menu_option*

Synonyms

MG IN

Description

This command initialises the GUI for a menu option. It copies the current value of the session variables for the option to the GUI and sets the active state to true.

Parameters

The following table describes the parameters of the MenuGUI Initialise command:

Parameter	Description
<i>menu_option</i>	The menu option to initialise the GUI for.

Related Script Commands

[MenuGUI Active](#), [MenuGUI Update](#)

Version

5.0 Original

MenuGUI Load

Syntax

MenuGUI Load *menu_option, file_name, err_var*

Synonyms

MG LD

Description

This command loads the settings into the specified menu GUI from a saved script. It will also make the specified GUI option active.

Parameters

The following table describes the parameters of the MenuGUI Load command:

Parameter	Description
<i>menu_option</i>	The name of the menu option
<i>file_name</i>	The name of the script to load the options from
<i>err_var</i>	The variable to receive an error code if the load fails. This value will be 0 if the load succeeds.

Related Script Commands

[MenuGUI Active](#), [MenuGUI Save](#)

Version

5.0 Original

MenuGUI LoadDefault

Syntax

MenuGUI LoadDefault *menu_option*, *leaf_name*

Synonyms

MG LF

Description

This command loads the default settings for the specified menu option. It is usually used on the Default button of a dialog box being used to provide a menu GUI.

Parameters

The following table describes the parameters of the MenuGUI LoadDefault command:

Parameter	Description
<i>menu_option</i>	The name of the menu option
<i>leaf_name</i>	The leaf name of the file to read the default from. This value is added on to the path to the default folder to find the default file.

Version

5.0 Original

MenuGUI LoadEmulationKeys

Syntax

MenuGUI LoadEmulationKeys *key_set*

Synonyms

MG LE

Description

This command loads the specified keys from the current emulation into the keyboard menu GUI.

Parameters

The following table describes the parameters of the MenuGUI LoadEmulationKeys command:

Parameter	Description
<i>key_set</i>	The keys to load from the emulation. See table below.

Values for key_set

Value	Description
<i>0</i>	Edit keys
<i>1</i>	Function keys
<i>2</i>	Mouse buttons
<i>3</i>	Numeric keypad
<i>4</i>	Alpha numeric keys

Version

5.0 Original

MenuGUI MapVariable

Syntax

MenuGUI MapVariable *menu_option*, *variable*, *set_flag*, [*get_flag*], [*set_extra*], [*get_extra*], [*prefix*]

Synonyms

MG MV

Description

This command sets up the mapping between a session variable and it's representation on the GUI displaying the menu option.

Parameters

The following table describes the parameters of the MenuGUI MapVariable command:

Parameter	Description
<i>menu_option</i>	The name of the menu option
<i>variable</i>	The name of the session variable
<i>set_flag</i>	The mapping to apply. See the table below.
<i>get_flag</i>	The mapping to use when getting the variable from the GUI. It has the same meaning as the <i>set_flag</i> . If omitted it default to the same as the <i>set_flag</i> .
<i>set_extra</i>	Extra information required by the <i>set_flag</i> .
<i>get_extra</i>	Extra information required by the <i>get_flag</i> . Defaults to the same value as the <i>set_extra</i> parameter.
<i>prefix</i>	The prefix used in a session variable that is removed in the Menu GUI.

Values for set_flag

Value	Description
0	There is no mapping
1	Map to default attribute of control with same name on the dialog box with the same name as the menu option.
2	Set/Get script functions are defined
3	As 1 but the property to use is specified by the set_extra/get_extra parameters
4	As 3 except the session value is 0 based and the GUI property is 1 based.
5	Map to control/property in a named dialog box. set_extra/get_extra parameters specify the dialog/control/property.
6	As 5 but the prefix parameter is added/removed from the value when setting/getting the data from the GUI
7	Map to control/property in the dialog box with the same name as the menu option with a prefix that is added/removed from the value when setting/getting the data from the GUI

Related Script Commands

[MenuGUI DefaultMap](#), [MenuGUI Initialise](#), [MenuGUI Update](#)

Version

5.0 Original

MenuGUI Save

Syntax

MenuGUI Save *menu_option, file_name, header, err_var*

Synonyms

MG SV

Description

This command saves the settings from the current menu gui to a file as a script.

Parameters

The following table describes the parameters of the MenuGUI Save command:

Parameter	Description
<i>menu_option</i>	The name of the menu option
<i>file_name</i>	The name of the file to save. If this is not a full path the path is expanded to be in the user directory.
<i>header</i>	The header to write as a comment at the beginning of the saved script
<i>err_var</i>	The variable to receive the error number in if the save fails. This value is 0 if the save succeeds.

Related Script Commands

[MenuGUI Load](#), [MenuGUI GetScript](#)

Version

5.0 Original

MenuGUI SetupComms

Syntax

MenuGUI SetupComms *port_name*

Synonyms

MG SC

Description

This command is used to ensure that the communications menu option and variables have been loaded.

Parameters

The following table describes the parameters of the MenuGUI SetupComms command:

Parameter	Description
<i>port_name</i>	The name of the communications port to set up.

Version

5.0 Original

MenuGUI Update

Syntax

MenuGUI Update *menu_option*, *err_var*, [*err_text_var*]

Synonyms

MG UP

Description

This command reads the information from the GUI for a menu option and sets the session variables.

Parameters

The following table describes the parameters of the MenuGUI Update command:

Parameter	Description
<i>menu_option</i>	The name of the menu option to update.
<i>err_var</i>	Variable to be set to error number if an error occurs or 0 if there is no error.
<i>err_text_var</i>	The name of a variable to receive a textual description of the error.

Related Script Commands

[MenuGUI Active](#), [MenuGUI Initialise](#)

Version

5.0 Original

MenuList

Syntax

MenuList

Description

The MenuList function returns a cr separated list of all the currently defined menus.

Parameters

None

Return Value

cr separated list of all the currently defined menus

Example

List all the current menus to the screen

```
* List all the current menus to the screen
* showing which are attached to the main menu
Display FF
Display Text "List of menus\r\n\r\n"
Display Text "Menu name"
Display At 30
Display Text "Attached to main?\r\n"

menus = MenuList()
num_menus = Count(menus,cr) + (menus # "")
j = 0
Loop
While j < num_menus
j+=1
menu_name = Extract(menus,j)
Display Text menu_name
Display At 30
on_menu = "No"
If IsOnMenu(mainmenu, menu_name) Then on_menu = "Yes"
Display Text on_menu:cr:lf
Repeat

Display LF
```

Version

4.1 First implemented

MessageBox

Syntax

MessageBox (*function*) **MessageBox** *message*, [*title*], [*flags*], [*attach_to*]

Description

The MessageBox command displays a message in a popup window and waits until the button is clicked. The message box is modal, which means that keystrokes and menu selections will not go to wIntegrate until the box is closed.

The MessageBox can be used as a command in which case any return value will be ignored, or as a function which returns a value depending on which of the buttons was pressed. See below.

Parameters

The following table describes the parameters of the MessageBox command:

Parameter	Description
<i>message</i>	The text of the message to display.
<i>title</i>	The title for the message box. If you do not specify, the
<i>flags</i>	Icon and Response buttons to display in the box. (See below)
<i>attach_to</i>	Name of the dialog box to attach to. An attached message

Values for flags

Separate button and icon flags with a vertical bar (|), e.g., MB_OK | MB_ICONHAND.

Value	Description
<i>MB_OK</i>	Displays an “OK” button.
<i>MB_OKCANCEL</i>	Displays “OK” and “Cancel” buttons.

Value	Description
<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.
<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.

Return Value

The text of the button pressed to close the message box is returned. i.e. "OK", "Cancel", "Yes", "No", "Retry", "Ignore", or "Abort"

Example

The following examples shows using the MessageBox as a command

```
* Inform user his email has been dispatched OK
MessageBox "Your Email has been sent", "Email Confirmation", MB_OK |
MB_ICONASTERISK
```

The following example shows the use of MessageBox as a function

```
reply = MessageBox("Are you sure?", , MB_YESNO | MB_ICONQUESTION)
If reply = "Yes" Then
    Print "Let's go!"
```

Related Script Commands

[MessageBox](#)

Mid

Syntax

Mid *string, start_position, [number_chars]*

Description

The Mid function returns section of the original string from a given position that is the specified number of characters long.

Parameters

The following table describes the parameters of the Mid command:

Parameter	Description
<i>string</i>	The string to check for character(s).
<i>start_position</i>	The position of the character to start from.
<i>number_chars</i>	The number of characters to return. If this is omitted, Mid returns all characters to the end of the string.

Return Value

The sub string specified by the start position and length

Example

The following example returns the string "nteg":

```
* Returns the string "nteg"
val = Mid(wIntegrate", 3,4)
* Returns the string "rate"
val = Mid("wIntegrate", 7)
Mid Parameters
```

Related Script Commands

[Left](#), [Right](#)

Mouse

Syntax

Mouse option

Description

The Mouse function returns information on the mouse position.

Parameters

The following table describes the parameters of the Mouse command:

Parameter	Description
<i>option</i>	Use the following options to return information about the mouse position

Values for option

Value	Description
<i>0</i>	(Get_X) The X coordinate in the screen position.
<i>1</i>	(Get_Y) The Y coordinate in the screen position.
<i>2</i>	(Get_Page) The screen back page.
<i>3</i>	Returns “True” if the mouse is over the session window.
<i>4</i>	Returns “True” if the mouse is over terminal window and on back page 0 (the current screen).
<i>5</i>	Returns x coordinate in 10th of a character
<i>6</i>	Returns y coordinate in 10th of a character

Return Value

The information about the mouse position as specified in the option parameter

Example

The following example sends the mouse coordinates to the host if it is on the current screen:

```
* Send the mouse coordinate to the host if on the current screen
If Mouse (4) Then Enter Mouse(Get_X) : "," : Mouse(Get_Y)
```

In the next example, a message box displays if the mouse is within 1.5 character of the left of the current screen:

```
If Mouse(5) < 15 Then MessageBox "The mouse is within 1.5 character of
the left of the screen"
```

MultiExec

Syntax

MultiExec *selection, options*

Description

The MultiExec command execute one or more host statement via the host server. This command is the only server command that does not open the server. You must use another server library command first, or use the OpenServer command before you use MultiExec.

The MultiExec command is a Server Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

The following table describes the parameters of the MultiExec command:

Parameter	Description
<i>selection</i>	One or more statements separated by carriage returns.
<i>options</i>	See below.

Values for options

Use one or more of the following options:

Value	Description
<i>F</i>	Display output in full screen.
<i>Q</i>	Quit the server after the command is run.
<i>P</i>	Send data to local (PC) printer.
<i>W</i>	Wait for a carriage return after running statements.

Example

The following example shows the use of the MultiExec command:

```
OpenServer  
sel = 'SSELECT WIN.ORDER WITH CUSTNAME LIKE "A..."'  
Ins sel,-1, "LIST WIN.ORDER WITH QUANTITY < 100"  
MultiExec sel, "F W Q"
```

MultiScript

Syntax

MultiScript *script_list, command_line, delimiter*

Synonyms

MT

Description

The MultiScript command runs all the script files specified in the *script_list* parameter one after the other. If the *script_list* is empty this command just returns immediately.

Parameters

The following table describes the parameters of the MultiScript command:

Parameter	Description
<i>script_list</i>	List of the script file names to be run.
<i>command_line</i>	The command line that will be passed to each of the scripts
<i>delimiter</i>	A single character delimiter that is used to separate the script file names. This defaults to the pipe character " ".

Example

Run two of the example scripts

```
MultiScript "example\script\effect.wis|example\script\display"
```

Re-run all of the load scripts for this session

```
MultiScript Get(LoadScript)
```

Version

4.1.1 Original

NextRow

Syntax

NextRow

Description

The NextRow command moves the next icon position to the next row. The number of rows for the icon bar is specified in CreateIconBar.

The NextRow command is an Icon Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

None

Example

The following example is from the "wintsys\iconbar\bar_tech.wis" script. This technical icon bar in wIntegrate has two rows of icons.

```
Library 'wintsys\lib\iconlib'
CreateIconBar Technical_bar,2
AddIcon 'image\i_copyto.bmp',"Show EditCopyTo"
AddIcon 'image\i_copyts.bmp',"Invoke EditCopySpecial"
AddIcon 'image\i_copyb.bmp',"Dialog EditCopyTo;Set CopyTo =
'Clipboard';Set
Format = 'Bitmap';Invoke EditCopySpecial"
.
.
.
NextRow
AddIcon 'image\i_copyt.bmp',"Dialog EditCopyTo;Set CopyTo =
'Clipboard';Set
Format = 'Table';Invoke EditCopySpecial"
```

Related Script Commands

[AddGap](#), [AddIcon](#), [AddLargeIcon](#), [AddWideIcon](#), [CreateIconBar](#), [EndCreateIconBar](#)

Not

Syntax

Not *expression*

Description

The Not function reverses a Boolean value.

Parameters

The following table describes the parameters of the Not command:

Parameter	Description
<i>expression</i>	Any valid expression.

Return Value

Reverses the boolean value of the given expression. 0 or "" is returned as 1. All other values are returned as 0.

Example

The following example is part of the COPYMENU.WIS demonstration program.

```
If Not(IsApp("notepad.exe")) Then
Dialog RunProgram
Set Filename = 'notepad.exe'
Set Arguments = ''
Invoke
EndIf
```

Notify

Syntax

Notify *script_command*

Description

The Notify command sets a script command to be run by the host server immediately after the next command sent to the server completes.

The Notify command is a Server Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

The following table describes the parameters of the Notify command:

Parameter	Description
<i>script_command</i>	Script command to run when the next server command finishes

Version

4.1.1 Original

Object Enum

Syntax

Object Enum *object_name*, [*enum_name*], [*global*], [*prefix*], [*filename*]

Synonyms

OBJECT EN

Description

The Object Enum commands defines or saves variables set to the value of the enumerated constants from within the type library of an automation object.

Parameters

The following table describes the parameters of the Object Enum command:

Parameter	Description
<i>object_name</i>	Name of the object to read the enumerated constants for
<i>enum_name</i>	Name of the enumerated constant or "" for all constants for this object
<i>global</i>	True to create global variables for this object (the default), False to create local variables.
<i>prefix</i>	Text to prefix the variables name with
<i>filename</i>	If specified variables are saved to this file instead of being defined

Example

Dynamically load constants for use with the Find Execute method

```
Object enum WordApp, "WdFindWrap",False
Object enum WordApp, "WdReplace", False
* The above lines defined the wdFindContinue and wdReplaceAll
constants used below
Find.Execute optParam,optParam,optParam,optParam,optParam,optParam,
True,wdFindContinue, optParam, optParam, wdReplaceAll
```

Create a file with all the enumerated constants for excel

```
Object New xl, "Excel.Sheet", err
If err = 0 Then
Object Enum xl, "", True, "", "c:\excel_const.wis"
Object Release xl
EndIf
```

Version

4.0.1 Original

Object Get

Syntax

Object Get *name, filename, prog_ID, error_var*

Synonyms

OBJECT GT

Description

This command will get an automation object from a file or retrieve an active object. When an object is no longer needed it should be released using the Object Release command.

Parameters

The following table describes the parameters of the Object Get 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 prog_ID specified is retrieved.
<i>prog_ID</i>	The automation object or its class ID in {}. If omitted the default object for the specified filename is returned.
<i>error_var</i>	Name of a variable to return the error code. See table below

Values for error_var

Value	Description
0	No error
1	Can't find the specified prog_ID
2	Unable to create the required object

Value	Description
3	Object does not have a Dispatch interface so can not be used for Automation
4	Unable to retrieve type information for the object
5	Unable to retrieve IUnknown interface
6	Unable to retrieve IPersistFile interface
7	Unable to load the file
8	Can't get bind context for this object
9	Syntax error in file name
10	No active object of specified type

Example

Get an Excel work book automation object from the Tab separated file c:\ab.txt

```
Object Get xl, "c:\ab.txt", "Excel.Sheet", err
```

Related Script Commands

[Object New](#), [Object Release](#), [Object Set](#)

Version

4.0 Original

Object New

Syntax

Object New *name*, *prog_ID*, *error_var*

Synonyms

OBJECT NW

Description

This command creates a new Automation object. When an object is no longer needed it should be released using the Object Release command.

Parameters

The following table describes the parameters of the Object New command:

Parameter	Description
<i>name</i>	The name of the object.
<i>prog_ID</i>	The automation object or its class ID in { }
<i>error_var</i>	Name of a variable to return the error code. See below

Values for error_var

Value	Description
0	No error
1	Can't find ProgId
2	Object create failed
3	Object does not have dispatch interface
4	Can't load type information for object

Example

Create a new Word application object and show it

```
Object New WordApp,"Word.Application", err  
If err = 0 Then WordApp.Visible = True
```

Related Script Commands

[Object Get](#), [Object Release](#), [Object Set](#)

Version

4.0 Original

Object Release

Syntax

Object Release *name*

Synonyms

OBJECT RL

Description

This command is used to release an automation object after it has been used. This will free any memory and resources used by this object.

Parameters

The following table describes the parameters of the Object Release command:

Parameter	Description
<i>name</i>	The name of the object.

Example

Release the range object after autofitting cells in Excel

```
Object set xlrange = xlsheet.range(data_start::":":data_end)
xlrange.columns.AutoFit
Object Release xlrange
```

Related Script Commands

[Object Get](#), [Object New](#), [Object Set](#)

Version

4.0 Original

Object Set

Syntax

Object Set *name* = *dispatch_object*, [*err_var*]

Synonyms

OBJECT ST

Description

This command is used to give an object name to an automation object that has been returned from a property or method from another automation object. When an object is no longer needed it should be released using the Object Release command.

Parameters

The following table describes the parameters of the Object Set command:

Parameter	Description
<i>name</i>	The name of the object.
=	An equals sign must be specified
<i>dispatch_object</i>	Property or method of another automation object which returns a dispatch interface.
<i>err_var</i>	The name of a variable to set with an error code if the object set fails. See table below for the return values.

Values for err_var

The values returned in the error variable are:

Value	Description
0	No error
1	Invalid dispatch value. e.g. A property or method has returned 0 for the dispatch interface.

Value	Description
2	Not a dispatch value. This is usually caused if a dispatch interface return value has been modified after it was returned.
3	Unable to retrieve the type information for the returned dispatch interface. Without this information the returned object can not be used.
4	Object exists. The name used in the first parameter has already been used.

Example

Auto fit columns in Excel

```
* Get the first work sheet in the Excel workbook, create a
* range object to address a group of cells and autofit them
Object set xlsheet = xlbook.sheets.item(1)
Object set xlrange = xlsheet.range(data_start:"":data_end)
xlrange.columns.AutoFit
Object Release xlrange
Object Release xlsheet
```

Open a file using UniObjects

```
* The Session object holds a previously opened UniObjects session
Object Set MyFile = Session.OpenFile("MY.BP"), errVar
If errVar Then
    MsgBox "Unable to open file MY.BP"
Else
    * Do something with the file here
EndIf
```

Related Script Commands

[Object Get](#), [Object New](#), [Object Release](#)

Version

5.1.1 Optional error variable added

OConv

Syntax

OConv *string, format*

Description

The Oconv function converts string or numeric data from internal format to display format based on conversion codes. If the input value or conversion code is invalid, Oconv returns "".

Parameters

The following table describes the parameters of the OConv command:

Parameter	Description
<i>string</i>	The string to convert.
<i>format</i>	See the following table for information on different format conversions:

Values for format

Value	Description
<i>MT</i>	Time
<i>D</i>	Date Format
<i>MD</i>	Masked Decimal
<i>MC</i>	Masked Character. MCU - Upper Case, MCL - Lower Case
<i>MB</i>	Decimal to binary
<i>MX</i>	Decimal to Hexadecimal
<i>SA<key></i>	Convert string from scrambled format into the original format. <key> must match the key used in the IConv function to scramble the data.

Time formats

You must use the MT format to specify a time format, then add any of the remaining time formats:

Time Format	Description
MT	This format must be specified, and the other formats can be added. This format returns a 24-hour time format like 14:45.
H	Converts the time into 12-hour format like 02:45PM.
S	Adds seconds to the time format like 14:45:30 (MTS format) or 02:45:30PM (MTHS format).
HS	This combines the H and S formats above, which results in this format - 02:45:30PM (MTHS format).

Date formats

You must use the D format to specify a date format, then add any of the remaining date formats for the conversion you want. You may use only the “D” format option, and the date is converted according to the Windows Control Panel setup. The syntax of the date formatting is as follows: D (year_digits) (format_string) (separator).

Date Format	Description
D	You must use “D” to specify a date format. When used alone, it returns a date in the format set in the Control Panel. Example: Dec 18 1995 (assuming Control Panel setting is Month, Day, Year).
year_digits	The number of digits of the year. The default is 4. Example: “D2” format displays Dec 18 95.
separator	The character you want to use to separate day, month, and year. Example: “D2/” displays 12/18/95.
Y	Returns only the year. Example: 1995

Date Format	Description
M	Returns only the month number. Example: 12
MA	Returns the name of the month. Example: December
MS	Returns the month name abbreviation. Example: Dec
MZ	Returns the month number without the leading zero. Example: 2 (Instead of “02” for February.)
D	Returns the day number with a leading zero. Example: 05 (Monday = 01, Sunday = 07.)
DZ	Returns the day of the month. Example: 18
W	Returns the day number without a leading zero. Example: 5 (Monday = 1, Sunday = 7.)
WA	Returns the name of the week day. Example: Friday
Q	Returns the current quarter of the year. Example: 4
J	Returns the day of the year. Example: 355 (for December 21.)

Masked Decimal

The conversion format is: MDn[f] [,] [\$] [P] [S] [Z]

Parameter	Description
n	The number of decimal places after the decimal point. If n is less than f, the value is rounded.
f	The optional scaling factor. If omitted, it is set to n.
,	Thousands delimiter (place a comma between thousands).
\$	Currency delimiter (prefix with the dollar symbol).
P	Data contains a point (scaling factor f is ignored).
S	Strip spaces and if specified, \$ (dollar sign) symbols from the data.

Parameter	Description
Z	Suppress leading zeros.

Return Value

Copy of string with the output format applied

Example

The following example shows a time conversion.

```
* Convert the internal time and show in a message box
MessageBox "Current time is: " : Oconv(Time(),"MTHS")
```

The following example shows an Oconv date conversion:

```
* Convert the internal date and show in a message box
MessageBox "Current date is: " : Oconv(Date(),"DWAMSDY")
```

Masked decimal conversions

```
val = Oconv(1234, "MD2"); * val = 12.34
val = Oconv(12345, "MD23"); * val = 12.35
val = Oconv(912345, "MD2$,"); * val = $9,123.45
val = Oconv(1234.5, "MD2P,"); * val = 1,234.50
val = Oconv($1,234.56, "MD$,PS"); * val = 1234.56
```

Get upper case version of a string

```
* Set Value to ABCDEF
Value = OConv("AbCdEf", "MCU")
```

Get lower case version of a string

```
* Set Value to abcdef
Value = OConv("AbCdEf", "MCL")
```

The following converts 5 into binary:

```
* Set Value to 00000101  
Value = Oconv(5, "MB")
```

The next example converts 289 to binary:

```
* Set Value to 0000000100100001  
Value = Oconv(289, "MB")
```

The following example converts 254 to hexadecimal:

```
* Set value to FE  
value = Oconv(254, "MX")
```

Related Script Commands

[Date](#), [Time](#), [Stamp](#), [Iconv](#)

Version

4.2.1 Added scramble format "SA"

OpenServer

Syntax

OpenServer

Description

The OpenServer command starts the host side of the wIntegrate server. It always attempts to start the server and does not return any error code so it is better to use the CheckServer command to start the server.

The OpenServer command is a Server Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

None

Version

4.0 Original

Or

Syntax

expr1 Or expr2

Description

The Or connective is used in If and Loop statements. It is used between expressions, the value is true if one of the expressions is true. It has a lower priority than the And connective.

Parameters

The following table describes the parameters of the Or command:

Parameter	Description
<i>expr1</i>	Expression that evaluates to a true or false value
<i>expr2</i>	Expression that evaluates to a true or false value

Example

The following example code is from the PRODM2.WIS demonstration program.

```
* Validate Price
Sub OnValidatePrice()
If Not(Match(ProdM.Price,"1N0N")) Or ProdM.Price < 1 Then
    MsgBox 'Price must be an integer greater than zero', "Product
Maintenance", MB_ICONHAND | MB_OK, ProdM
    DialogBox InputOK, ProdM, False
Else
    DialogBox InputOK ProdM, True
    If ProdM_Mode < 8 Then CheckWrite 4
EndIf
```

Related Script Commands

[And](#), [If](#), [Loop](#)

PathAddExt

Syntax

PathAddExt *file_name, extension*

Description

This function returns the file name given to it with the specified extension added if the file name does not already have an extension.

Parameters

The following table describes the parameters of the PathAddExt command:

Parameter	Description
<i>file_name</i>	The name of the file to add the extension to
<i>extension</i>	The file extension to add

Return Value

The file name with the given extension or the original file name if it already had an extension.

Example

Ensure that the file name always has the "txt" extension

```
text_file_name = PathAddExt(text_file_name, "txt")
```

Change the extension of a file to "wis"

```
file_name = PathAddExt(PathComponent(file_name,3), "wis")
```

Related Script Commands

[PathAppend](#), [PathComponent](#)

Version

6.0 Original version

PathAppend

Syntax

PathAppend *path*, *item*

Description

This function returns the given file path with the given folder/file name appended to it.

Parameters

The following table describes the parameters of the PathAppend command:

Parameter	Description
<i>path</i>	Path to append to.
<i>item</i>	item to append to the path. This can be a folder, leaf name or a path to a lower level folder or leaf name. If this is set to "" the original path is return with a trailing directory delimiter.

Return Value

The new path built from the parameters

Example

Get the path of test.txt in the User directory

```
test_file_name = PathAppend(UserDir, "test.txt")
```

Related Script Commands

[PathAddExt](#), [PathComponent](#)

Version

6.0 Original version

PathComponent

Syntax

PathComponent *file_name*, *option*

Description

This function returns the requested parts of the given file name depending on the option specified.

The parts returned are a combination of one or more of the folder holding the file name, the leaf name without it's extension and the file extension.

Parameters

The following table describes the parameters of the PathComponent command:

Parameter	Description
<i>file_name</i>	The original file name
<i>option</i>	The component(s) to return.

Values for option

Add together one or more of the following

Value	Description
<i>1</i>	The directory
<i>2</i>	The leaf name without the extension
<i>4</i>	The file extension

Return Value

The components requested

Example

Get the folder for a file

```
folder = PathComponent(file_name, 1)
```

Check if the files extension is "txt"

```
If PathComponent(file_name, 4) = "txt" Then Print file_name:" is a  
text file"
```

Get the leaf name with its extension

```
leaf = PathComponent(file_name, 6)
```

Related Script Commands

[PathAddExt](#), [PathAppend](#)

Version

6.0 Original version

Play Abort

Syntax

Play Abort

Synonyms

PLAY AB

Description

The Play Abort command permanently stops the current playback. The Play Start command will not restart it.

Note: Use a Dialog EditPlay command to start a playback after information has been recorded using the Capture command.

Parameters

None

Example

The following example shows the use of the Play Abort command:

```
reply = MessageBox("Continue playback?", "", MB_YESNO)
If reply = "No" Then
  Play Abort
Else Play Start
EndIf
```

Related Script Commands

[PlayInfo](#), [Play Start](#), [Play Stop](#)

Play Start

Syntax

Play Start

Synonyms

PLAY ST

Description

The Play Start command restarts the playback after the Play Stop command.

Note: Use a Dialog EditPlay command to start a playback after information has been recorded using the Capture command.

Parameters

None

Example

The following example shows the use of the Play Start command.

```
Play Stop
reply = MessageBox("Continue playback?", "", MB_YESNO)
If reply = "No" Then
Play Abort
Else Play Start
EndIf
```

Related Script Commands

[Play Abort](#), [PlayInfo](#), [Play Stop](#)

Play Stop

Syntax

Play Stop

Synonyms

PLAY SP

Description

The Play Stop command stops the current playback.

Note: Use a Dialog EditPlay command to start a playback after information has been recorded using the Capture command.

Parameters

None

Example

The following example shows the use of the Play Stop command:

```
Play Stop
reply = MessageBox("Continue playback?", "", MB_YESNO)
If reply = "No" Then
Play Abort
Else Play Start
EndIf
```

Related Script Commands

[Play Abort](#), [PlayInfo](#), [Play Start](#)

PlayInfo

Syntax

PlayInfo

Description

The PlayInfo command returns "True" if a playback is in progress.

Note: Use a Dialog EditPlay command to start a playback after information has been recorded using the Capture command.

Parameters

None

Return Value

True if the playback is currently playing, otherwise false.

Example

The following example uses PlayInfo to check if a playback is running, If it is, a message box displays asking the user to choose to continue.

```
Dialog EditPlay
Set Text = "True"
Invoke

If PlayInfo() Then Play Stop
reply = MessageBox("continue?", "", MB_YESNO)
If reply = "No" Then
    Play Abort
Else
    Play Start
EndIf
```

Related Script Commands

[Play Abort](#), [Play Start](#), [Play Stop](#)

PopupCalculator

Syntax

PopupCalculator *name, edit control, button control*

Description

The command pops up a window with the calculator control. It updates the value from an edit control and is right aligned to the specified button control. Pressing return or the equals button updates the edit control. Pressing escape or clicking outside the popup window closes it without updating the edit control

This command is implemented in the DboxLib script library.

Parameters

The following table describes the parameters of the PopupCalculator command:

Parameter	Description
<i>name</i>	The name of the dialog box with the edit control
<i>edit control</i>	The name of the edit control to update with the number from the calculator
<i>button control</i>	The name of the button control to right align the popup calculator with

Example

Add a popup calculator to dialog

```
Example Add a popup calculator to dialog
DialogBox Create CalcDemo
...
LText "Popup Calculator",None,93,52,66,12
EditText PopupEdit,93,68,65,12
GraphicButton AppDir:"Image\i_calc.bmp",PopupButton, 160, 68, 14, 12,
GBS_RAISED
OnEvent PopupButton,"Click",'Library
"wintsys\lib\dboxlib";PopupCalculator CalcDemo,PopupEdit,PopupButton'
...
EndCreate
```

* For the full script of this example see "example\script\calc.wis"

Version

4.2.1 Original

PortInfo

Syntax

PortInfo [*option*]

Description

This function returns information about the available communications ports.

Parameters

The following table describes the parameters of the PortInfo command:

Parameter	Description
<i>option</i>	Specifies the information to return. Defaults to 0.

Values for option

Value	Description
<i>0</i>	List of currently available communications ports
<i>1</i>	List of currently available serial communications ports

Return Value

cr separated list of information for the specified value of the option parameter.

Example

Check if serial communication port 2 is available

```
pos = Locate(PortInfo(1), "COM2")  
If pos Then Print "Serial port 2 is available"
```

Version

5.0 Original

Print

Syntax

Print *text*

Synonyms

PR

Description

The Print command displays text on the status bar. To erase the text printed on the status line, use another Print command with the text null ("").

Parameters

The following table describes the parameters of the Print command:

Parameter	Description
<i>text</i>	The text to print on the status line.

Example

The following example line comes from the DISPBOX.WIS demonstration program.

```
* Display message on Status line
Print "You answered " : Reply : ", so here we go"
```

PrinterInfo

Syntax

PrinterInfo *option*, [*printer*]

Description

This function returns information on the printers on the system.

Parameters

The following table describes the parameters of the PrinterInfo command:

Parameter	Description
<i>option</i>	A number identifying the information to return. See the table below.
<i>printer</i>	The name of the printer to check. This is only required for option 2.

Values for option

Value	Description
0	Return the name of the default printer
1	Return a list of all the printers on the system
2	Check that the printer named in the printer parameter can be accessed.

Return Value

The result required for the specified option.

Version

5.0 Original

5.1.1 Option 2 check printer

Random

Syntax

Random *option*

Description

This function returns a random number generated by the computer.

Parameters

The following table describes the parameters of the Random command:

Parameter	Description
<i>option</i>	Options specifying the range of the random number returned. See table below for details.

Values for option

Value	Description
<i>0</i>	Random number between 0 and 32767
<i>1</i>	Random number between 0 and 1 (note wIntegrate does not support floating point numbers directly so the decimal conversions in oconv would have to be used before the returned value can be used in any calculation)
<i>+n</i>	n is 2 to 32767. Random number between 1 and n
<i>-n</i>	n is a seed value for the random number generator. Repeating the same seed number will repeat the same set of numbers. The value returned is ""

Return Value

The return is random number depending on the option number.

Example

Return a value from 1 to 100

```
* Seed the random number generator based on the time and return a  
* value from 1 to 100  
dum = Random(-time())  
val = Random(100)
```

Also see the example script `race.wis`

Version

4.0.3 Original

Receive

Syntax

Receive *flag*

Synonyms

RC

Description

The Receive command starts or stops receiving host data.

Parameters

The following table describes the parameters of the Receive command:

Parameter	Description
<i>flag</i>	Can be one of the following: True - Starts the receipt of data, False - Stops the receipt of data.

Example

The following example turns receiving off:

```
* Turn receiving off
Receive Off
The next example turns receiving on:
* Turn receiving on
Receive On
```

Red

Syntax

Red *color*

Description

The Red function returns the red intensity of a specified RGB (Red Green Blue) color. On a color monitor, colors are displayed as varying intensities of red, green, and blue dots. Intensity is the amount of red, green, or blue on a scale of 0 through 255.

Parameters

The following table describes the parameters of the Red command:

Parameter	Description
<i>color</i>	The color name or number.

Return Value

The amount of red in the given color from 0 to 255.

Related Script Commands

[Blue](#), [ChooseColor](#), [Green](#)

Rep

Syntax

Rep *var_name*, *line*, [*tab*], *string*

Description

The Rep command replaces data at a line or tab position within a string.

Parameters

The following table describes the parameters of the Rep command:

Parameter	Description
<i>var_name</i>	The name of the variable.
<i>line</i>	The line number of the string to replace.
<i>tab</i>	The tab position number. If this is omitted, the whole line is replaced.
<i>string</i>	The new value.

Example

The following example is part of the PRESTORE.WIS demonstration program.

```
If Extract(Prestore_State,pno) Then
* Recording in progress
Capture Off $cname
Rep Prestore_text, pno, CaptureText($cname, DF_BS)
Capture Delete $cname
DialogBox SetText Prestore,$id,id
Rep Prestore_State,pno,0
Capture On $cname, "", "Keystrokes"
DialogBox SetText Prestore,$id,"R":pno
Rep Prestore_State,pno,1
Else
Type Extract(Prestore_text, pno)
EndIf
```

Related Script Commands

[Del](#), [Extract](#), [Ins](#)

Repeat

Syntax

Repeat

Description

This command is placed at the end of a block of commands started with the Loop statement.

When it is encountered the script continues from the next statement after the Loop.

Parameters

None

Related Script Commands

[Loop](#), [While](#), [Until](#)

ResolveEffect

Syntax

ResolveEffect *effect*, [*fore_color*], [*back_color*], [*combine*], [*def_fore*], [*def_back*], [*ignore_bold*], [*ignore_reverse*]

Description

The ResolveEffect function returns the colours, flashing and underline states used to display a particular effect on the screen.

Parameters

The following table describes the parameters of the ResolveEffect command:

Parameter	Description
<i>effect</i>	The name or number of the effect to return the information for.
<i>fore_color</i>	Foreground color for the effect. Defaults to value set in Setup Colors.
<i>back_color</i>	Background color for the effect. Defaults to value set in Setup Colors
<i>combine</i>	The combination flags. This is a value calculated from the advanced section of setup colors. It defaults to the current value for the effect.
<i>def_fore</i>	The current default foreground color (0-15). Default to value from Setup Colors.
<i>def_back</i>	The current default background color (0-15). Default to value from Setup Colors.
<i>ignore_bold</i>	Ignore bold effect. Defaults to value from Setup Colors.
<i>ignore_reverse</i>	Ignore reverse flag. Defaults to value from Setup Colors.

Return Value

The foreground color, background color, flashing flag and underline flag in a cr separated list.

Example

Check if current normal effect is green

```
If Extract(ResolveEffect(0),1) = RGB_LightGreen Then  
    Print "Foreground of normal effect is green"  
EndIf
```

Check real underline flag is set for underline effect

```
If Extract(ResolveEffect("UnderLine"),4) = 1 Then Print "Underline  
set"
```

Version

5.0 Original

Return

Syntax

Return *value*

Synonyms

RT

Description

The Return command exits the current user-defined function, and returns the value.

Parameters

The following table describes the parameters of the Return command:

Parameter	Description
<i>value</i>	The value of a variable to return.

Example

The following example comes from the PIECHART.WIS demonstration program.

```
If percent <= 25 Then
val = Extract(PCS, 26-percent)
Else If percent <= 50 Then
val = -Extract(PCS, percent-24)
Else If percent <= 75 Then
val = -Extract(PCS, 76-percent)
Else
val = Extract(PCS, percent-74)
EndIf
return val
```

Related Script Commands

[Define](#)

RGB

Syntax

RGB *red, green, blue*

Description

The RGB command defines a color by specifying color intensities of red, green, and blue on a scale of 0 to 255.colors. The RGB number is used by other wIntegrate commands to define a color.

Parameters

The following table describes the parameters of the RGB command:

Parameter	Description
<i>red</i>	The color intensity of red on a scale of 0 to 255.
<i>green</i>	The color intensity of green on a scale of 0 to 255.
<i>blue</i>	The color intensity of blue on a scale of 0 to 255.

Return Value

The internal color number that includes the given intensities of red, green, blue.

Example

The following example shows the RGB color bright red (255,0,0).

```
Draw Pen RGB(255,0,0)
Draw Brush 0,BRUSH_Null
Draw Line xl,yt,xl,yb
Draw Line xr,yb
```

Related Script Commands

[Blue](#), [Green](#), [Red](#)

Right

Syntax

Right *string, number_chars*

Description

The Right function returns the right-most character(s) of a string.

Parameters

The following table describes the parameters of the Right command:

Parameter	Description
<i>string</i>	The string to check for the right-most character(s).
<i>number_chars</i>	The number of characters to return.

Return Value

The specified right-most characters of the string

Example

The following example returns "ate" which are the last three characters of the string "wIntegrate":

```
* Set the variable val to "ate"
val = Right("wIntegrate", 3)
```

Related Script Commands

[Left](#), [Mid](#)

Run

Syntax

Run *path*, [*params*], [*ret_var*], [*show_opt*], [*verb*], [*def_dir*]

Description

The Run command runs a file or program on the local computer.

It uses the Windows associations between file extensions and programs to decide what program to run with the file.

Parameters

The following table describes the parameters of the Run command:

Parameter	Description
<i>path</i>	The path to the program or file name.
<i>params</i>	Additional parameters if path is a program.
<i>ret_var</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 below for the most common errors.
<i>show_opt</i>	The show option is a request. The applications will not necessarily start with the specified show state. The default is 0 (show and activate). See the table below.
<i>verb</i>	Either "Open" to open a document or run an application, or "Print" to print a file (Print is not supported by all applications).
<i>def_dir</i>	Default directory (default none) for program to start in

Values for ret_var

Value	Description
0	System was out of memory or executable file was corrupt.
2	File was not found
3	Path was not found.
31	No association found for file type, or verb not supported.
Others	Executable would not load or run under Windows.

Values for show_opt

Value	Description
0	Show and Activate
1	Minimize
2	Maximize
3	Hide
4	Show but do not activate

Example

The first example shows the most simple way to use the Run command. It opens up the Notepad application.

```
* Start up notepad
Run "notepad"
```

The next example opens an application and a file.

```
* Start excel with expenses sheet
Run "excel", "C:\expenses.xls"
```

This next example uses the `ret_var` to check errors. The last line of the program specifies that if the value of `ret_var` is less than or equal to 32, a message box appears:

```
* Alternative way to start excel sheet (with error checking)
Run "C:\expensis.xls","", ret_var
If ret_var <= 32 Then MsgBox "Couldn't open expenses sheet"
```

This next example opens and prints a document without displaying the application.

```
* Print word from windows document without showing it in the desktop
Run "c:\docs\letter.doc",,ret_var, 3, "Print"
```

Screen CopyPage

Syntax

Screen CopyPage *number*

Synonyms

SN CP

Description

The Screen CopyPage command copies the data from a backpage to the current screen.

The cursor remains at the bottom of the screen. The number of back pages is specified in Setup Terminal.

Parameters

The following table describes the parameters of the Screen CopyPage command:

Parameter	Description
<i>number</i>	The number of the back page to copy to the current screen.

Example

In the following example, the sixth back page is copied to the screen:

```
Screen CopyPage 6
```

Related Script Commands

[Screen Load](#), [Screen Off](#), [Screen On](#), [Screen Remove](#), [Screen Restore](#), [Screen Save](#), [Screen ScrollRegion](#), [Screen Store](#), [ScreenText](#), [ScreenWord](#)

Screen Load

Syntax

Screen Load *filename*, [*x1*], [*y1*], [*x2*], [*y2*], [*b1*], [*b2*]

Synonyms

SN LD

Description

The Screen Load command displays a screen saved with the Screen Save command. Use the x1, y1, etc. parameters only if you want to override the screen position that you designated when you stored the screen.

Parameters

The following table describes the parameters of the Screen Load command:

Parameter	Description
<i>filename</i>	"filename" The name of the PC file that stores the screen. Designate the full path unless the file is in the wIntegrate directory. If you do not specify a file extension, wIntegrate appends the ".wim" extension.
<i>x1</i>	The starting column position to display the saved area.
<i>y1</i>	The starting row position to display the saved area.
<i>x2</i>	The right-most position to display the saved area.
<i>y2</i>	the bottom-most position to display the area.
<i>b1</i>	The back page number of the top left. (0 is the current page.)
<i>b2</i>	The back page number of the top right. (0 is the current page.)

Example

The following example sets screen colors and saves that to the file "reg".

```
* The colors are set to blue text on white, on which the message box
appears. * If the report is displays, it is on the white and blue
screen. If not, the saved screen is loaded.
* Set regular screen colors
Color COL_Green, COL_Black
Display FF
* Store the screen
Screen Save "reg"
Display FF
* Message box for the report
reply = MessageBox("Run 'Student?'", , MB_OKCANCEL | MB_ICONQUESTION)
If reply = "OK" Then
Enter "SORT STUDENT NAME BY NAME"
Else Screen Load "reg"
```

Related Script Commands

[Screen CopyPage](#), [Screen Off](#), [Screen On](#), [Screen Remove](#), [Screen Restore](#), [Screen Save](#), [Screen ScrollRegion](#), [Screen Store](#), [ScreenText](#), [ScreenWord](#)

Screen Off

Syntax

Screen Off

Synonyms

SN OF

Description

The Screen Off command stops host information from updating to the screen. The host information continues to pass through the emulation while the screen is off, so you can still record host data. You may want to use this command while programs are compiling and you want the process to be invisible to the user. use the Screen On command to turn the screen on again.

Parameters

None

Example

The following example turns off the screen display while executing a command, then turns the screen back on to display the report.

```
* Formfeed to clear screen
Display FF
* Turn off the display while executing Report
Screen Off
Capture On stuname, "stu_name.txt", "Keystrokes"
Enter "LIST STUDENT LNAME BY LNAME"
Capture Off stuname
reply = MessageBox("Display Report?", , MB_YESNO | MB_ICONQUESTION)
If reply = "Yes" Then
Screen On ;* Turn screen display back on
Dialog EditPlay
Set Filename = "stu_name.txt"
Set Text = "True"
Invoke
EndIf
```

Related Script Commands

[Screen CopyPage](#), [Screen Load](#), [Screen On](#), [Screen Remove](#), [Screen Restore](#), [Screen Save](#), [Screen ScrollRegion](#), [Screen Store](#), [ScreenText](#), [ScreenWord](#)

Screen On

Syntax

Screen On

Synonyms

SN ON

Description

The Screen On command allows host information to update the screen. Use the Screen Off command to turn the screen updating off.

Parameters

None

Example

The following example turns off the screen display while executing a command, then turns the screen back on to display the report.

```
* Formfeed to clear screen
Display FF
* Turn off the display while executing Report
Screen Off
Capture On stuname, "stu_name.txt", "Keystrokes"
Enter "LIST STUDENT LNAME BY LNAME"
Wait ForText ">"
Capture Off stuname
reply = MessageBox("Display Report?", , MB_YESNO | MB_ICONQUESTION)
If reply = "Yes" Then
Screen On ;* Turn screen display back on
Dialog EditPlay
Set Filename = "stu_name.txt"
Set Text = "True"
Invoke
EndIf
```

Related Script Commands

[Screen CopyPage](#), [Screen Load](#), [Screen Off](#), [Screen Remove](#), [Screen Restore](#), [Screen Save](#), [Screen ScrollRegion](#), [Screen Store](#), [ScreenText](#), [ScreenWord](#)

Screen Remove

Syntax

Screen Remove *name*

Synonyms

SN RM

Description

The Screen Remove command releases the memory used by a Screen Store command.

Parameters

The following table describes the parameters of the Screen Remove command:

Parameter	Description
<i>name</i>	The name you designated for the screen in Screen Store.

Example

The following example sets the screen color, font, etc., then stores it.

```
* Set regular screen colors
Color COL_Green, COL_Black
Display FF
* Set the font and pen color for the text
Draw Font "Times_Roman", 30,30,Font_Heavy|Font_Italic
Draw Pen RGB_Green
Draw Text 75,140, "wIntegrate"
* Store the screen
Screen Store reg
* Message box for the report
reply = MessageBox("Run 'Student?'," , MB_OKCANCEL | MB_ICONQUESTION)
If reply = "OK" Then
Enter "SORT STUDENT NAME BY NAME"
Else Screen Restore reg
Screen Remove reg
```

Related Script Commands

[Screen CopyPage](#), [Screen Load](#), [Screen Off](#), [Screen On](#), [Screen Restore](#), [Screen Save](#),
[Screen ScrollRegion](#), [Screen Store](#), [ScreenText](#), [ScreenWord](#)

Screen Restore

Syntax

Screen Restore *name*, [*x1*], [*y1*], [*x2*], [*y2*], [*b1*], [*b2*]

Synonyms

SN RS

Description

The Screen Restore command restores a screen stored by the Screen Store command. The dimensions of the screen are stored and are used as the default.

Use the x1, y1, etc. parameters only if you want to override the screen position that you designated when you stored the screen.

Parameters

The following table describes the parameters of the Screen Restore command:

Parameter	Description
<i>name</i>	The name of the screen stored by Screen Store.
<i>x1</i>	The starting column position of the stored area.
<i>y1</i>	The starting row position of the stored area.
<i>x2</i>	The right-most position of the stored area.
<i>y2</i>	the bottom-most position of the area.
<i>b1</i>	The back page number of the top left. (0 is the current page.)
<i>b2</i>	The back page number of the top right. (0 is the current page.)

Example

The following examples shows the use of the Screen Restore command:

```
* Set regular screen colors
Color COL_Green, COL_Black
Display FF
* Set the font and pen color for the text
Draw Font "Times_Roman", 30,30,Font_Heavy|Font_Italic
Draw Pen RGB_Green
Draw Text 75,140, "wIntegrate"
* Store the screen
Screen Store reg
* Message box for the report
reply = MessageBox("Run 'Student?'", , MB_OKCANCEL | MB_ICONQUESTION)
If reply = "OK" Then
Enter "SORT STUDENT NAME BY NAME"
Else Screen Restore reg
EndIf
Screen Remove reg
```

Related Script Commands

[Screen CopyPage](#), [Screen Load](#), [Screen Off](#), [Screen On](#), [Screen Remove](#), [Screen Save](#), [Screen ScrollRegion](#), [Screen Store](#), [ScreenText](#), [ScreenWord](#)

Screen Save

Syntax

Screen Save *filename*, [*x1*], [*y1*], [*x2*], [*y2*], [*b1*], [*b2*]

Synonyms

SN SV

Description

The Screen Save command saves an area of the terminal screen to the PC disk. Use this command to create special screen effects or to save an entire screen to use later.

If you want to save the entire screen, do not use the *x1*, *y1*, etc. parameters.

Tip: To show the saved screen later in the same script or in another script, use the Screen Load command.

Parameters

The following table describes the parameters of the Screen Save command:

Parameter	Description
<i>filename</i>	The name of the PC file that will store the screen. Designate the full path unless the file is in the wIntegrate directory. If you do not specify a file extension, wIntegrate appends the ".wim" extension.
<i>x1</i>	The starting column position of the saved area.
<i>y1</i>	The starting row position of the saved area.
<i>x2</i>	The right-most position of the saved area.
<i>y2</i>	the bottom-most position of the area.
<i>b1</i>	The back page number of the top left. (0 is the current page.)
<i>b2</i>	The back page number of the bottom right. (0 is the current page.)

Example

The following example sets screen colors and saves that to the file "reg". Then the colors are set to blue text on white, on which the message box appears. If the report is displays, it is on the white and blue screen. If not, the saved screen is loaded.

```
* Set regular screen colors
Color COL_Green, COL_Black
Display FF
* Store the screen
Screen Save "reg"
Display FF
* Message box for the report
reply = MessageBox("Run 'Student?'," , MB_OKCANCEL | MB_ICONQUESTION)
If reply = "OK" Then
Enter "SORT STUDENT NAME BY NAME"
Else Screen Load "reg"
EndIf
```

Related Script Commands

[Screen CopyPage](#), [Screen Load](#), [Screen Off](#), [Screen On](#), [Screen Remove](#), [Screen Restore](#), [Screen ScrollRegion](#), [Screen Store](#), [ScreenText](#), [ScreenWord](#)

Screen ScrollRegion

Syntax

Screen ScrollRegion [*x1*], [*y1*], [*x2*], [*y2*], *move_outside*

Synonyms

SN SR

Description

The Screen ScrollRegion command restricts terminal output and scrolling to a region of the screen. Use Screen ScrollRegion without parameters to restore scrolling to the entire screen.

Parameters

The following table describes the parameters of the Screen ScrollRegion command:

Parameter	Description
<i>x1</i>	The starting column position of the scroll region.
<i>y1</i>	The starting row position of the scroll region.
<i>x2</i>	The right-most position of the scroll region.
<i>y2</i>	The bottom-most position of the scroll region.
<i>move_outside</i>	Specifies that the cursor can be moved outside the current scroll region, using cursor addressing (e.g. MoveXY emulation) functions. When a scroll region is set, the cursor is moved to the top left of the scroll region or the top left of the screen if <i>move_outside</i> is "true".

Example

The following example shows a host side subroutine (WIN.CEDEMO) that uses a script command.

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

Related Script Commands

[Screen CopyPage](#), [Screen Load](#), [Screen Off](#), [Screen On](#), [Screen Remove](#), [Screen Restore](#), [Screen Save](#), [Screen Store](#), [ScreenText](#), [ScreenWord](#)

Screen Store

Syntax

Screen Store *name*, [*x1*], [*y1*], [*x2*], [*y2*], [*b1*], [*b2*]

Synonyms

SN ST

Description

The Screen Store command stores an area of the terminal screen in memory. If you want to store the entire screen, do not use the *x1*, *y1*, etc. parameters.

To show the screen again in the same script, use the Screen Restore command.

Note: If you want to save the screen to a file to use in other scripts after the session has benn stopped, use the Screen Save command. Saved screens can be used again using the Screen Load command.

Parameters

The following table describes the parameters of the Screen Store command:

Parameter	Description
<i>name</i>	The name you give to the stored screen.
<i>x1</i>	The starting column position of the stored area.
<i>y1</i>	The starting row position of the stored area.
<i>x2</i>	The right-most position of the stored area.
<i>y2</i>	the bottom-most position of the area.
<i>b1</i>	The back page number of the top left. (0 is the current page.)
<i>b2</i>	The back page number of the top right. (0 is the current page.)

Example

Set regular screen colors

```
Color COL_Green, COL_Black
Display FF
* Set the font and pen color for the text
Draw Font "Times_Roman", 30,30,Font_Heavy|Font_Italic
Draw Pen RGB_Green
Draw Text 75,140, "wIntegrate"
* Store the screen
Screen Store reg
* Message box for the report
reply = MessageBox("Run 'Student?'", , MB_OKCANCEL |
MB_ICONQUESTION)
If reply = "OK" Then
Enter "SORT STUDENT NAME BY NAME"
Else Screen Restore reg
EndIf
Screen Remove reg
```

Related Script Commands

[Screen CopyPage](#), [Screen Load](#), [Screen Off](#), [Screen On](#), [Screen Remove](#), [Screen Restore](#), [Screen Save](#), [Screen ScrollRegion](#), [ScreenText](#), [ScreenWord](#)

ScreenOn

Syntax

ScreenOn

Description

This function checks if screen updating is on.

Parameters

None

Return Value

True (1) is screen updating is on otherwise False (0).

Related Script Commands

[Screen On](#), [Screen Off](#), [DisplayOn](#)

Version

5.0 Original

ScreenText

Syntax

ScreenText *x1*, *y1*, *x2*, *y2*, [*format*], [*b1*], [*b2*]

Description

The ScreenText function returns the text from a specified screen area.

Parameters

The following table describes the parameters of the ScreenText command:

Parameter	Description
<i>x1</i>	The starting column position of the text.
<i>y1</i>	The starting row position of the text.
<i>x2</i>	The right-most position of the text.
<i>y2</i>	The bottom-most position of the text.
<i>format</i>	Specifies a format. For more information, see the following Formats table.
<i>b1</i>	The back page number of the top left. (0 is the current page.)
<i>b2</i>	The back page number of the top right. (0 is the current page.)

Values for format

Value	Description
<i>1</i>	Lines - This default format returns full lines of text.
<i>2</i>	Block - This returns only what is in the specified rectangle.
<i>3</i>	Lines Table - Parses out columns from full lines of text and separates them by tabs.

Value	Description
4	Block Table - Parses out columns from a block of text and separates them by tabs.

Return Value

The text from the specified area of the screen.

Example

The following example is taken from the PARSE.WIS demonstration program.

```
text = ScreenText(x1,y1,x2,y2,type,b1,b2)
Convert text, lf, "" ;* Remove linefeeds from returned text
```

Related Script Commands

[Screen CopyPage](#), [Screen Load](#), [Screen Off](#), [Screen On](#), [Screen Remove](#), [Screen Restore](#), [Screen Save](#), [Screen ScrollRegion](#), [Screen Store](#), [ScreenWord](#)

ScreenWord

Syntax

ScreenWord *x*, *y*, [*format*], [*number*], [*delimiter*]

Description

The ScreenWord function returns the word or character from the screen depending on the format.

Parameters

The following table describes the parameters of the ScreenWord command:

Parameter	Description
<i>x</i>	The column to start the search.
<i>y</i>	The line or row position to start the search.
<i>format</i>	The format for the search. 0 - Finds the word to the right of the specified position. 1 - Finds only the highlighted letter (search position.) 2 - Allows you to use the delimiter parameter.
<i>number</i>	The backpage number to start the search.
<i>delimiter</i>	The delimiter you specify for parsing the word. Default is a space. See the second example.

Return Value

The word at the specified screen position

Example

The following example shows that when ScreenWord is set as a mouse key definition, it sends the word the mouse is clicked on to the host.

```
Enter ScreenWord(Mouse(Get_X), Mouse(Get_Y))
```

The next example uses the delimiter parameter as a macro on a mouse button.

```
Enter ScreenWord(Mouse(Get_X),Mouse(Get_Y),2,Mouse(2), " ()"  
* The above program example returns a.b from the screen text "(a.b )"  
":
```

Related Script Commands

[Screen CopyPage](#), [Screen Load](#), [Screen Off](#), [Screen On](#), [Screen Remove](#), [Screen Restore](#), [Screen Save](#), [Screen ScrollRegion](#), [Screen Store](#), [ScreenText](#)

Script

Syntax

Script *scriptfile*, [*commandline*]

Synonyms

SC

Description

The Script command runs a script from the current script. It returns to the current script when the called script ends.

Parameters

The following table describes the parameters of the Script command:

Parameter	Description
<i>scriptfile</i>	The full path of the script file to run.
<i>commandline</i>	The command line to pass to the script. When the new script starts, the global variable CommandLine is set to the value of commandline or to Null if commandline is omitted.

Example

The following example shows a part of the STRING.WIS demonstration program.

```
InitCommand "Script 'example\script\dostring' "  
ControlCommand OK, "Script 'example\script\dostring' "  
ControlCommand CANCEL, "DialogBox End StringTest,FALSE"  
EndCreate
```

Related Script Commands

[Chain](#), [Execute](#), [Library](#)

ScriptError End

Syntax

ScriptError End *script_id*

Synonyms

SE EN

Description

This command will end the specified error script.

Parameters

The following table describes the parameters of the ScriptError End command:

Parameter	Description
<i>script_id</i>	Identifier for the error script to end

Related Script Commands

[ScriptError EndAll](#)

Version

5.0

ScriptError EndAll

Syntax

ScriptError EndAll

Synonyms

SE EA

Description

This command will shut down all the scripts that currently have stopped due to script errors.

Parameters

None

Related Script Commands

[ScriptError End](#)

Version

5.0 Original

ScriptError Exit

Syntax

ScriptError Exit *script_id*, [*parent_id_var*]

Synonyms

SE EX

Description

This command will Exit the current script and return to the parent script (if any). The parent script will then be added to the list of error scripts.

Parameters

The following table describes the parameters of the ScriptError Exit command:

Parameter	Description
<i>script_id</i>	The identifier of the script to exit
<i>parent_id_var</i>	If specified this variable is set to the value of the parent identifier for this script. If the script did not have a parent the variable is set to 0.

Related Script Commands

[ScriptError End](#)

Version

5.0

ScriptError Ignore

Syntax

ScriptError Ignore *script_id*

Synonyms

SE IG

Description

This command will ignore the current error in a script and continue the script from the next statement

Parameters

The following table describes the parameters of the ScriptError Ignore command:

Parameter	Description
<i>script_id</i>	The identifier of the error script.

Related Script Commands

[ScriptError Step](#), [ScriptError Skip](#), [ScriptError RestartScript](#)

Version

5.0 Original

ScriptError ReplaceLine

Syntax

ScriptError ReplaceLine *script_id, new_line*

Synonyms

SE RL

Description

This command allows the line that caused an error in a script file to be replaced to allow the script to continue. The command will only work on non-compiled scripts and scripts where the source code is available. The `ScriptErrorInfo(17)` function returns true if this command will modify the script.

Note: The line is replaced only for the running script. To make a permanent change the original script file must be changed.

Parameters

The following table describes the parameters of the `ScriptError ReplaceLine` command:

Parameter	Description
<i>script_id</i>	The identifier of the error script.
<i>new_line</i>	The script source code to replace the error line

Related Script Commands

[ScriptErrorInfo](#)

Version

5.0

ScriptError RestartScript

Syntax

ScriptError RestartScript *script_id*

Synonyms

SE RS

Description

This command restarts an error script. The error in the script must have been fixed or skipped over otherwise the error that stopped the script will be repeated

Parameters

The following table describes the parameters of the ScriptError RestartScript command:

Parameter	Description
<i>script_id</i>	The script identifier of the error script.

Related Script Commands

[ScriptError Ignore](#), [ScriptError Skip](#), [ScriptError Step](#)

Version

5.0 Original

ScriptError Skip

Syntax

ScriptError Skip *script_id*

Synonyms

SE SK

Description

This command will skip over the error statement in the specified script and show the next line in the script monitor.

Parameters

The following table describes the parameters of the ScriptError Skip command:

Parameter	Description
<i>script_id</i>	The script identifier of the error script

Related Script Commands

[ScriptError Step](#), [ScriptError Ignore](#)

Version

5.0 Original

ScriptError Step

Syntax

ScriptError Step *script_id*

Synonyms

SE ST

Description

This command will run the current statement in an error script and then stop the script and return to the script monitor.

Parameters

The following table describes the parameters of the ScriptError Step command:

Parameter	Description
<i>script_id</i>	The error script identifier

Related Script Commands

[ScriptError Skip](#), [ScriptError Ignore](#)

Version

5.0 Original

ScriptErrorInfo

Syntax

ScriptErrorInfo *option*, [*script_id*]

Description

This function returns information about any script that has generated a script error.

Parameters

The following table describes the parameters of the ScriptErrorInfo command:

Parameter	Description
<i>option</i>	Specify the information to return. See table below.
<i>script_id</i>	The script identifier. This value must be specified for options 2 and greater.

Values for option

The number specifies the information to return. For option 2 and above a valid script id must be provided

Value	Description
<i>0</i>	List of error scripts
<i>1</i>	Number of error scripts
<i>2</i>	Check id valid
<i>3</i>	Next error script in list
<i>4</i>	Previous error script in list
<i>5</i>	Error number and text
<i>6</i>	Script name
<i>7</i>	Run type
<i>8</i>	Run From
<i>9</i>	File name

Value	Description
<i>10</i>	Current subroutine
<i>11</i>	Compiled script
<i>12</i>	Error line number
<i>13</i>	Error line text
<i>14</i>	Lines around error
<i>15</i>	Context (command running)
<i>16</i>	Location in error list
<i>17</i>	Edit line allowed
<i>18</i>	Edit script allowed
<i>19</i>	Script in error state
<i>20</i>	Local variables
<i>21</i>	Reason script has been stopped

Return Value

The information specified by the option parameter.

Version

5.0 Original

ScriptGlobalInfo

Syntax

ScriptGlobalInfo *topic*, [*object*]

Description

This command returns information about global script objects and data. A global script object is an object that is not tied to any one script and can be seen by all scripts.

Parameters

The following table describes the parameters of the ScriptGlobalInfo command:

Parameter	Description
<i>topic</i>	The name of the area for the script object. If "" a list of all topics is returned.
<i>object</i>	The name of the object to return the data for. If this is "" a list of all objects in the specified topic is returned.

Return Value

A cr separated list of the information returned. This is dependent on the values specified for the topic and object parameters.

Version

5.0 Original

ScriptInfo

Syntax

ScriptInfo *option*

Description

The ScriptInfo function returns a number showing how a script was run. The number returned correlates to the "run type" of the script. See the Run Type table for descriptions.

Parameters

The following table describes the parameters of the ScriptInfo command:

Parameter	Description
<i>option</i>	Specifies the script to return information on: 0 - Returns the run type of the current script. 1 - Returns the run type of the beginning script of the current script chain.

Return Value

The returned run type. See table below

Value	Description
<i>1000</i>	File The script was run from a file. For example from RunScript, FileOpen, or Script command.
<i>1001</i>	New Session This running script is creating a new session.
<i>1002</i>	Start Up This script is run once when wIntegrate starts.
<i>1003</i>	Load Script This script runs after a session is loaded.
<i>2000</i>	Execute The script is run by the Execute command.

Value	Description
3000	Menu Item This script is run from the menu or the Show command.
3001	Menu Invoked This script is run using the Invoke command.
3002	Menu Open This script is run when the menu opens.
3003	Menu Close This script runs when a menu closes.
3100	DialogBox Initialize Script runs when a dialog box is initialized.
3101	DialogBox Button Script runs when a dialog box button is clicked.
3102	DialogBox Scroll Script runs when a scroll bar is moved.
3103	DialogBox Listbox Script runs when a list box is double-clicked.
3104	DialogBox Valid Script runs on a dialog box control validation.
3105	DialogBox Default Script runs when a dialog box control default control is clicked.
3106	DialogBox OnEvent script runs on all events on dialog box controls not specified above.
3107	DialogBox OnDialogEvent script runs on events on the dialog box (3100 is used for Init event).
4000	Host Command Script runs from the host Command escape sequence.
4001	Terminal Command Script runs as a result of an emulation definition.

Value	Description
4002	StartCommand script runs when an emulation file is loaded.
5000	HotSpot Script is activated by a hot spot.
5100	Soft key - mouse Script is run from a soft key or a mouse button script.
5200	Trigger script runs when a screen trigger activates.
5300	Event OnHostText script runs when specified characters are received.
5400	Event OnExit script runs when an attempt is made to exit the session.
5401	Event OnRestore script runs when an attempt is made to restore a minimized session.
5402	Event OnPortClose script runs when the communications port is closed.
5403	Event OnPortOpen script runs when the communications port is opened.
5500	MenuGui Show script runs to show the interface for a menu option.
5501	MenuGui Set script runs to set a variable on a menu option interface.
5502	MenuGuiGet script runs to get a variable on a menu option interface.
5600	CallBack script runs when an internal process requires a scripted GUI interface.
6000	Application Script is run from another Windows application using Dynamic Data Exchange.
7000	ErrorHandler script runs when an error in a script has been detected.

Example

The following example shows the use of the ScriptInfo function:

```
* If run from the host inform the host the script has finished  
If ScriptInfo(1) = 4000 Then Enter "Done"
```

Version

5.0 Script types 5500-5502, 5600 and 7000

ScriptVersion

Syntax

ScriptVersion *FileName*

Description

This function returns the version information placed into a source or compiled script by the *\$Version directive.

Parameters

The following table describes the parameters of the ScriptVersion command:

Parameter	Description
<i>FileName</i>	The script file name to read the version from. If this is omitted the version is returned from the currently running script file.

Return Value

The version text from the *\$Version directive

Example

Check the version of a compiled script

```
script_file = AppDir:"example\script\hexview"
If ScriptVersion(script_file:".wis") #
ScriptVersion(script_file:".wix") Then MessageBox "Script versions
differ from the source and compiled versions of the script"
```

Version

4.2.1 Original

Select Area

Syntax

Select Area *x1, y1, x2, y2, [type], [b1], [b2]*

Synonyms

SL AR

Description

The Select Area command sets the current screen selection.

The screen selection is shown in reverse and is used by the various copy menu options to copy data to the clipboard (or location specified for copy special). The selection can also be set by the user by dragging the mouse with the left or right button held down.

Use other Select commands to increase or change the size of the selected area. Use Select Clear to clear the selected areas of the screen.

Parameters

The following table describes the parameters of the Select Area command:

Parameter	Description
<i>x1</i>	The starting column position of the selected area.
<i>y1</i>	The starting row position of the selected area.
<i>x2</i>	The right-most position of the selected area.
<i>y2</i>	The bottom-most row of the selected area.
<i>type</i>	0 if lines are selected, 1 if a block is selected.
<i>b1</i>	The back page number of the top left. (0 is the current page.)
<i>b2</i>	The back page number of the bottom right. (0 is the current page.)

Example

The following example uses the Select Area command to select the top part of the wIntegrate screen.

```
Select Area 0, 0, 79, 10
```

Related Script Commands

[Select Clear](#), [Select End](#), [Select Extend](#), [SelectInfo](#), [Select Keyboard](#), [Select Start](#)

Select Clear

Syntax

Select Clear

Synonyms

SL CL

Description

The Select Clear command clears the current selection.

The screen selection is shown in reverse and is used by the various copy menu options to copy data to the clipboard (or location specified for copy special). The selection can also be set by the user by dragging the mouse with the left or right button held down.

Parameters

None

Example

The following example selects a part of the screen and then clears the selection.

```
Select Area 0,0,79,10
Select End 40,11
.
.
.
Select Clear
```

Related Script Commands

[Select Area](#), [Select End](#), [Select Extend](#), [SelectInfo](#), [Select Keyboard](#), [Select Start](#)

Select End

Syntax

Select End *x1*, *y1*, [*b1*]

Synonyms

SL EN

Description

The Select End command moves the end position of the current selection.

The screen selection is shown in reverse and is used by the various copy menu options to copy data to the clipboard (or location specified for copy special). The selection can also be set by the user by dragging the mouse with the left or right button held down.

Parameters

The following table describes the parameters of the Select End command:

Parameter	Description
<i>x1</i>	The new ending column position of the selection.
<i>y1</i>	The new ending row position of the selection.
<i>b1</i>	The new back page ending position. The default is 0.

Example

The following example selects the top area of the screen, then selects a new end position that

```
selects part of the next line down.  
Select Area 0,0,79,10  
Select End 40,11
```

Related Script Commands

[Select Area](#), [Select Clear](#), [Select Extend](#), [SelectInfo](#), [Select Keyboard](#), [Select Start](#)

Select Extend

Syntax

Select Extend *x1*, *y1*, [*b1*]

Synonyms

SL EX

Description

The Select Extend command moves the starting or ending position to extend the current selection.

The screen selection is shown in reverse and is used by the various copy menu options to copy data to the clipboard (or location specified for copy special). The selection can also be set by the user by dragging the mouse with the left or right button held down.

Parameters

The following table describes the parameters of the Select Extend command:

Parameter	Description
<i>x1</i>	The column position to extend the selection.
<i>y1</i>	The row position to extend the selection.
<i>b1</i>	The new back page position. The default is 0.

Example

The following example selects the top half of the screen, then uses Select Extend to increase the

```
selected area to include all of the first back page.  
Select Area 0,0,79,10  
.  
.  
.  
Select Extend 0,0,1
```

Related Script Commands

[Select Area](#), [Select Clear](#), [Select End](#), [SelectInfo](#), [Select Keyboard](#), [Select Start](#)

Select Keyboard

Syntax

Select Keyboard *flag*

Synonyms

SL KB

Description

The Select Keyboard command turns on or off the keyboard selection mode.

The keyboard selection mode allows users to use the cursor keys make selection by holding down the SHIFT key to select lines or by holding down the CTRL key to select a block.

The screen selection is shown in reverse and is used by the various copy menu options to copy data to the clipboard (or location specified for copy special). The selection can also be set by the user by dragging the mouse with the left or right button held down.

Parameters

The following table describes the parameters of the Select Keyboard command:

Parameter	Description
<i>flag</i>	Can be one of the following: 0 (Off) - Turn off keyboard selection mode. 1 (On) - Turn on keyboard selection mode. 2 - Toggle keyboard selection mode.

Select Start

Syntax

Select Start *x1*, *y1*, [*b1*]

Synonyms

SL ST

Description

The Select Start command does not start a selection, but moves the start position of the current selection.

The screen selection is shown in reverse and is used by the various copy menu options to copy data to the clipboard (or location specified for copy special). The selection can also be set by the user by dragging the mouse with the left or right button held down.

Parameters

The following table describes the parameters of the Select Start command:

Parameter	Description
<i>x1</i>	The new starting column position of the selection.
<i>y1</i>	The new row position of the selection.
<i>b1</i>	The new back page starting position. The default is 0.

Example

The following example selects the top half of the screen, then uses Select Start to start the selection further down the screen.

```
Select Area 0,0,79,10
.
.
.
Select Start 0, 5
```

Related Script Commands

[Select Area](#), [Select Clear](#), [Select End](#), [Select Extend](#), [SelectInfo](#), [Select Keyboard](#)

SelectInfo

Syntax

SelectInfo

Description

The SelectInfo function returns the coordinates of the current selection.

Parameters

None

Return Value

See the following table

The list of information returned is in the following order:

Value	Description
<i>x1</i>	The starting column position of the selected area.
<i>y1</i>	The starting row position of the selected area.
<i>x2</i>	The right-most position of the selected area.
<i>y2</i>	The bottom-most position of the area.
<i>type</i>	0 if lines are selected, 1 if a block is selected.
<i>b1</i>	The back page number of the top left. (0 is the current page.)
<i>b2</i>	The back page number of the top right. (0 is the current page.)

Example

The following example is taken from the PARSE.WIS demonstration program.

```
* Parse columns from selected area of screen
If SelectInfo() = "" Then
  MessageBox "No section of screen selected"
EndScript
EndIf
```

Related Script Commands

[Select Area](#), [Select Clear](#), [Select End](#), [Select Extend](#), [Select Keyboard](#), [Select Start](#)

SendKeys

Syntax

SendKeys *keystring*, [*wait*]

Synonyms

SK

Description

The SendKeys command sends keys to the Windows keyboard input buffer. The keystrokes are sent to the active window.

Use this with applications that do not support DDE or OLE.

Parameters

The following table describes the parameters of the SendKeys command:

Parameter	Description
<i>keystring</i>	The keystrokes that you want to send to the other application. See the following Special Keys table.
<i>wait</i>	Returns the following: 0 - Return immediately. 1 - Wait until the keystrokes have been sent.

Values for keystring

Special key values used within the keystring parameter:

Value	Description
+	Designates the SHIFT key.
^	Designates the CTRL key.
%	Designates the ALT key.
+^%	Designates the use of all three of the above together.

Value	Description
<i>{char num}</i>	Designates the repeating of char, num times.
<i>{CAPSLOCK}</i>	Caps lock key
<i>{NUMLOCK}</i>	Number lock key
<i>{SCROLLLOCK}</i>	Scroll lock key
<i>{ESC}</i>	Escape key
<i>{ENTER}</i>	Return key
<i>{PRTSC}</i>	Print screen key
<i>{TAB}</i>	Tab key
<i>{BREAK}</i>	Break key
<i>{BACKSPACE}</i>	Back space key
<i>{BS}</i>	Back space key
<i>{BKSP}</i>	Back space key
<i>{DELETE}</i>	Delete key
<i>{DEL}</i>	Delete key
<i>{INSERT}</i>	Insert key
<i>{LEFT}</i>	Left arrow key
<i>{RIGHT}</i>	Right arrow key
<i>{UP}</i>	Up arrow key
<i>{DOWN}</i>	Down Arrow Key
<i>{PGUP}</i>	Page up
<i>{PGDN}</i>	Page Down
<i>{HOME}</i>	Home Key
<i>{END}</i>	End Key
<i>{F1} - {F16}</i>	Function keys 1 to 16

Value	Description
()	By enclosing several keystrokes in parenthesis and placing one of the shift characters before them will apply the shift to all the characters.

Example

The following example is part of the COPYMENU.WIS demonstration program.

```
If IsTask(title_text) Then
  Invoke EditCopy
  Activate title_text
  SendKeys "^V" ;* Paste key for notepad
Else
  MessageBox "This example copies to ":title_text:" only", "Copy To
  Notepad"
EndIf
```

Related Script Commands

[Activate](#)

SendMessage

Syntax

SendMessage *hwnd*, *msg*, [*wparam*], [*lparam*], [*format*]

Description

The SendMessage function sends a Windows message to the specified window handle.

This function maps to the Windows API SendMessage call, and requires detailed knowledge of Microsoft Windows messages. Sending incorrect messages to a window or passing incorrect parameters can cause a General Protection Fault or other system error.

Parameters

The following table describes the parameters of the SendMessage command:

Parameter	Description
<i>hwnd</i>	Window handle
<i>msg</i>	Message to send
<i>wparam</i>	16 bit message argument. Default 0
<i>lparam</i>	32 bit message argument. Default 0
<i>format</i>	Format of lparam (and post flag). See the Format table below. Default is no formatting. Specifies how to convert the lparam argument. If there are no formatting parameters, lparam is sent as a 32 bit integer. Only integers can be sent to other applications windows.

Values for format

Value	Description
<i>P</i>	Post the message instead of sending it.

Value	Description
<i>S</i>	Treat the lparam as a string.
<i>Vn</i>	lparam contains the name of a variable in which to place a return result. n is the maximum size of the result.
<i>L</i>	Result of the SendMessage specifies length of text data returned from the message call.
<i>R</i>	Convert from a dynamic array to a binary format for the call and the result from a binary format to a dynamic array. The format argument is specified between brackets { } as repeat type where repeat is the number of items of the following type and type is W for Word (16 bit integer) or D for Double Word (32 bit integer).
<i>RI</i>	Specify Conversion for input to SendMessage. repeat can be 0 to convert all further entries in the dynamic array. No conversion is done on output unless RO is also specified.
<i>RO</i>	Specify Conversion for output from SendMessage. No conversion is done on input unless RI is also specified.

Return Value

The integer result from the send message call

Example

The following example finds the number of lines in a multi-line edit control:

```
* Find the number of lines in a multi-line edit control
hwnd = Hwnd(CustMaint, Notes)
EM_GETLINECOUNT = 1034
If hwnd Then no_lines = SendMessage(hwnd, EM_GETLINECOUNT)
```

In the next example, the program segment replaces the current selection in an edit control with "The company":

```
* Replace current selection in edit control with "The company"
EM_REPLACESEL = 1042
hwnd = Hwnd(CustMaint, Notes)
If hwnd Then result = SendMessage(hwnd, EM_REPLACESEL, 0, "The
company", "S")
```

In the next example, the program segment gets the width of the formatting rectangle for a multi-line edit control:

```
* Get the width of the formatting rectangle for a multi-line edit
control
EM_GETRECT = 1026
hwnd = Hwnd(CustMaint, Notes)
If hwnd Then result = SendMessage(hwnd, EM_GETRECT, 0, "rect",
"V32RO{4W}")
width = Extract(rect,3) - Extract(rect,1)
```

In the next example, the program segment adds a border and resets the rectangle:

```
* Now add a border and reset the rectangle
EM_SETRECT = 1027
Rep rect, 1, 12
Rep rect, 3, 12 + width - 24
If hwnd Then result = SendMessage(hwnd, EM_SETRECT, 0, rect,
"SRI{4W}")
```

ServerErrMsg

Syntax

ServerErrMsg *error_no*

Description

The ServerErrMsg function returns a string description of the error number.

The ServerErrMsg command is a Server Library command. For more information, see "Appendix A - Scripting Libraries and Modules".

Parameters

The following table describes the parameters of the ServerErrMsg command:

Parameter	Description
<i>error_no</i>	An error message number returned by the server.

Return Value

The text for the error message

Example

The following example is a part of the DDE_SCR.WIS demonstration program.

```
Sub GetAndSend(HostFile, HostItem, HostAtt, $LinkName, LinkItem)
HReadv Value, HostFile, HostItem, HostAtt
If Server_Error Then
  MsgBox ServerErrMsg(Server_Error) : " " : HostFile : ", " :
  HostItem
Else
  DDE Poke $LinkName, LinkItem, Value
EndIf
EndSub
```

Session Enable

Syntax

Session Enable *flag*, [*session_name*]

Synonyms

SS EN

Description

The Session Enable command enables and disables a wIntegrate session window. When there is more than one session open, you can designate the session name to enable or disable.

Parameters

The following table describes the parameters of the Session Enable command:

Parameter	Description
<i>flag</i>	Can be one of the following: True - Enables the session window. False - Disables the session window.
<i>session_name</i>	The name of the session. The default is the current session.

Example

The following example disables the current wIntegrate window.

```
Session Enable False
```

Session Execute

Syntax

Session Execute *command*, [*session_name*]

Synonyms

SS EX

Description

The Session Execute command executes a script command on another session.

Parameters

The following table describes the parameters of the Session Execute command:

Parameter	Description
<i>command</i>	The script command to run.
<i>session_name</i>	The name of the session; the default is the current session.

Example

The following example runs a script using the Script command from Session Execute.

```
Session Execute "Script 'C:\pctest\time.wis'", wintegl
```

Related Script Commands

[Execute](#), [Session Script](#)

Session LockSize

Syntax

Session LockSize *locksize*

Synonyms

SS LS

Description

This command locks the session size to stop it from being resized by the user or through other script commands.

Note: This will also stop actions which can also cause a resize. e.g. changing the font size or autosizing when scroll bars are added/removed. The size grip will disappear on the status bar. The border around the whole session window will reduce in size when the lock is on and increase again when the lock is off.

Parameters

The following table describes the parameters of the Session LockSize command:

Parameter	Description
<i>locksize</i>	True to lock the size, false to unlock it

Version

4.0.3 Original

Session Move

Syntax

Session Move *x, y, [session_name]*

Synonyms

SS MV

Description

The Session Move command moves a wIntegrate session to a new position on the desktop. This command is like Set Window, but Session Move uses the outside area of the window, where the Set Window command uses the internal area of the window (where the terminal data displays). When there is more than one session, Session Move can move any of them, but does not change the window focus.

Parameters

The following table describes the parameters of the Session Move command:

Parameter	Description
<i>x</i>	The column position for the left side of the window.
<i>y</i>	The row position for the right side of the window.
<i>session_name</i>	The name of the session to move. The default is the current session.

Example

The following example moves the session to the top left of the screen.

```
Session Move 0,0, winteg2
```

Related Script Commands

[Set](#)

Session Script

Syntax

Session Script *scriptfile*, *command_line*, [*session_name*]

Synonyms

SS SC

Description

The Session Script command runs a script file on the named session.

Parameters

The following table describes the parameters of the Session Script command:

Parameter	Description
<i>scriptfile</i>	The full path name of the script file.
<i>command_line</i>	The optional command or data to pass to the script.
<i>session_name</i>	The name of the session where the script will run.

Example

The following example runs the DISPBOX.WIS demonstration script on a session named "Mail".

The script line was run from the session named "winteg1".
Session Script "example\script\dispbox.wis", , Mail

Related Script Commands

[Execute](#), [Script](#), [Session Execute](#)

Session Show

Syntax

Session Show *option*, [*session_name*]

Synonyms

SS SH

Description

The Session Show command changes the current session display. Use this command to hide, minimize, or maximize sessions, or to change the focus from one session to another.

Parameters

The following table describes the parameters of the Session Show command:

Parameter	Description
<i>option</i>	The options for how to display a session are: 0 - Normal 1 - Minimized 2 - Maximized 3 - Hidden
<i>session_name</i>	The name of the session. The default is the current session.

Example

The following example displays the "Mail" session as a maximized window.

```
Session Show 2, Mail
```

Session Size

Syntax

Session Size *width, depth, [session_name]*

Synonyms

SS SZ

Description

The Session Size command resizes the session window. wIntegrate automatically scales the font. Session Size works on the external area of the session window.

Parameters

The following table describes the parameters of the Session Size command:

Parameter	Description
<i>width</i>	The new width for the session window.
<i>depth</i>	The new depth for the session window.
<i>session_name</i>	The name of the session to resize. The default is the current session.

Example

The following example shows the use of the Session Size command:

```
Session Size 600, 400
```

Related Script Commands

[Set](#)

SessionInfo

Syntax

SessionInfo *option*, [*session_name*]

Description

The SessionInfo function returns information on a session, depending on the option.

Parameters

The following table describes the parameters of the SessionInfo command:

Parameter	Description
<i>option</i>	0 Returns the session window status, 1 checks if the session is busy, 2 Returns the sessions title with the special characters expanded.
<i>session_name</i>	The name of the session. The default is the current session.

Return Value

The information to return. If option is 0 see the table below. If option is 1 the value is true if the session is busy otherwise false

The session status value (when the option is 0) is:

Value	Description
0	Session is normal.
1	Session is minimized
2	Session is maximized.
3	Session is hidden.

Example

The following example checks the session named "Mail". If it is minimized, then it is shown normal size.

```
If SessionInfo(0, Mail) = 1 Then Session Show 0, Mail
```

Related Script Commands

[Sessions](#)

Version

4.1.0

SessionName

Syntax

SessionName

Description

Get the name of the current session.

Each session has a unique name that is defined in the Setup Preferences dialog box. This name is used to communicate between the sessions using the session command and is the topic name when remote applications talk to the session using DDE.

This is identical to using Get(Name).

Parameters

None

Return Value

The name of the current session

Sessions

Syntax

Sessions *option*, [*param*]

Description

The Sessions function returns information about running sessions.

Parameters

The following table describes the parameters of the Sessions command:

Parameter	Description
<i>option</i>	The options for what sessions to include. 0 - Returns the number of sessions that are running. 1 - Returns a list of the sessions defined by param.
<i>param</i>	Defines the sessions to list; use with option 1. See below.

Values for param

Value	Description
<i>1</i>	Exclude current session.
<i>2</i>	Include the current session only.
<i>+4</i>	Add internal ID number (assigned when loaded).
<i>+8</i>	Add the session name.
<i>+16</i>	Add the session title.

Return Value

A cr separated list of the information specified by the option and param parameters.

Example

The following example checks how many sessions are running and sends a message if there is more than one session:

```
If Sessions(0) >1 Then  
  MessageBox "Close one of your sessions"
```

The next example uses option 1, which returns a list. The 9 represents the param 1 + 8, which means that the current session (which is running the script), is excluded, and the session name is listed.

```
MessageBox "Other wIntegrate sessions" : Sessions((1),9)
```

Set

Syntax

Set *variable* [=] *value*

Description

The Set command sets a variable within the current session.

See the "Reference - Script Menu Options" in this manual for details of the menu options and variables available.

Parameters

The following table describes the parameters of the Set command:

Parameter	Description
<i>variable</i>	A current session variable name.
=	An assignment operator (e.g. =, :=, +=). The default is =.
<i>value</i>	A single value or list of values separated by commas.

Example

The following example is part of the AP.WIS demonstration program.

```
Set LeftArrow = Char(Asc("J")-64)
Set RightArrow = Char(Asc("K")-64)
Set UpArrow = Char(Asc("B")-64)
Set DownArrow = Char(Asc("N")-64)
Set Control_LeftArrow = Char(Asc("Y")-64)
Set Control_RightArrow = Char(Asc("U")-64)
Set Control_UpArrow = Char(Asc("D")-64)
Set Control_DownArrow = Char(Asc("F")-64)
Set Insert = Char(Asc("W")-64)
Set Delete = Char(Asc("L")-64)
Set Control_Delete = Char(Asc("O")-64)
Set Shift_Delete = Char(Asc("E")-64)
Set ControlShift_Insert = Char(Asc("R")-64)
Set Control_Home = Char(Asc("T")-64)
```



```
Set End = Char(Asc("G")-64)
```

The next examples show other variables that you can set.

```
Set Font = "'IBSFont',3,7"  
Set EditRecord.Format = "ControlCodes"
```

The next example uses the assignment operator ":= " to append a value to a variable.

```
Set Title = "wIntegrate"  
Set Title := " - demo of scripts"
```

Related Script Commands

[Store](#)

SetFocus

Syntax

SetFocus *where*

Synonyms

SF

Description

The SetFocus command sets the input focus to a session or dialog control. You will get an error message if the object of the SetFocus command is not available.

Parameters

The following table describes the parameters of the SetFocus command:

Parameter	Description
<i>where</i>	This must either be the name of a session, the name of a dialog box or the name of a dialog box control (i.e. dialog name.control name).

Example

The following example uses the PRODM2.WIS demonstration program.

```
* Enable fields/disable id entry
Sub EnableFields(flag)
DialogBox EnableAll ProdM, 1, flag
DialogBox Enable ProdM, ProdRef, Not(flag)
DialogBox Enable ProdM, Cancel, True
If Flag Then
SetFocus ProdM.Name
Else
SetFocus ProdM.ProdRef
EndIf
EndSub
```

Related Script Commands

[Activate](#)

SetIconColumn

Syntax

SetIconColumn *position*

Description

The SetIconColumn command moves the position for the next icon to be added to a toolbar. It is normally used with the GetIconColumn function to line up icons on multiple row toolbars.

The SetIconColumn command is a Icon Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

The following table describes the parameters of the SetIconColumn command:

Parameter	Description
<i>position</i>	The position for following icons (normally a value returned from GetIconColumn)

Version

4.0 Original

Show

Syntax

Show [*name*]

Synonyms

SH

Description

The Show command displays a dialog attached to a wIntegrate menu command.

The Invoke command is similar, but it runs the command without showing the associated dialog box. If there is no dialog box associated with the menu command, Show is identical to Invoke.

For a full list of menu options available see the "Reference - Script Menu Options" section of this manual.

Parameters

The following table describes the parameters of the Show command:

Parameter	Description
<i>name</i>	The name of the menu command to show. If no name is specified, Show displays the most recent menu command that was used in the Dialog command.

Example

The following example brings up the RunProgram dialog box with the requested information.

```
Dialog RunProgram
Set Filename "notepad"
Set Arguments "c:\pctest\pcnew.txt"
Show
```

Related Script Commands

[Configure](#), [Get](#), [Invoke](#), [Set](#), [Store](#)

ShowURL

Syntax

ShowURL *url*, [*where*]

Synonyms

SU

Description

This command will show the given URL in a web browser. For the Java Client a second parameter can be used to specify the frame to place the URL in.

No error is returned if the URL is invalid or can not be displayed. As the URL is launched asynchronously it can not be assumed that the web page has been displayed before following script lines are executed.

Parameters

The following table describes the parameters of the ShowURL command:

Parameter	Description
<i>url</i>	The url to display. If the url starts with "www." then the protocol is assumed to be http.
<i>where</i>	Where to show the URL. This option is only used in the Java Client version. See the table below. The default is "_blank"

Values for where

Use one of the following strings.

Value	Description
<i>_blank</i>	Show in a new window
<i>_self</i>	Show in the page containing the Java Client. Warning: this will close the Java Client.

Value	Description
<i>_parent</i>	Show in the parent frame. If there is no parent this acts as "_self". Warning: This may close the Java Client.
<i>_top</i>	Show in the top level frame. If the Java Client is in the top-level frame this acts as "_self". Warning: This may close the Java Client.
<i>name</i>	Show in the named frame. If the named frame doesn't exist a new window will be created with this name.

Example

Show the IBM home page.

```
ShowURL "www.ibm.com"
```

Version

6.0 Original version

Sound Loop

Syntax

Sound Loop *file_name*, [*ok_var*]

Synonyms

SD LP

Description

The Sound Loop command continuously plays a sound, restarting it when it reaches the end of the sound. The command returns immediately.

To stop the sound use the Sound Stop command or Sound Play command to play another sound.

The sound file specified must fit into available physical memory and be playable by an installed waveform-audio device driver. If the sound cannot be found the default system sound is played instead. If the default entry can't be found no sound is made and *ok_var* is set to false.

Parameters

The following table describes the parameters of the Sound Loop command:

Parameter	Description
<i>file_name</i>	The file name of the sound to be played
<i>ok_var</i>	The name of a variable to be set to true if the sound plays correctly.

Example

Play a sound repetively for 2 minutes

```
* MySoundDir is a variable set to the folder containing the sounds
* Play "CantAllow.wav" sound repetively for 2 minutes
Sound Loop MySoundDir:"CantAllow.wav"
Wait Delay "2m"
Sound Stop
```

Related Script Commands

[Sound Play](#), [Sound PlaySystem](#), [Sound Stop](#)

Version

4.1.1 Original

Sound Play

Syntax

Sound Play *file_name*, [*async*], [*ok_var*]

Synonyms

SD PL

Description

The Sound Play command plays a sound.

The sound file specified must fit into available physical memory and be playable by an installed waveform-audio device driver. If the sound cannot be found the default system sound is played instead. If the default entry can't be found no sound is made and *ok_var* is set to false.

Parameters

The following table describes the parameters of the Sound Play command:

Parameter	Description
<i>file_name</i>	The file name of the sound to be played
<i>async</i>	Set to true if the command should return immediately without waiting for the sound to finish. This is true by default.
<i>ok_var</i>	The name of a variable to be set to true if the sound plays correctly.

Example

Play "CantAllow.wav" sound and wait until it finished

```
* MySoundDir is a variable set to the folder containing the sounds
Sound Play MySoundDir:"CantAllow.wav", False, ok
If ok Then
* Play "OkThen.wav" without waiting for it finish
Sound Play MySoundDir:"OkThen.wav"
EndIf
```

Related Script Commands

[Sound Loop](#), [Sound PlaySystem](#), [Sound Stop](#)

Version

4.1.1 Original

Sound PlaySystem

Syntax

Sound PlaySystem *sound_name*, [*async*], [*ok_var*]

Synonyms

SD PS

Description

The Sound PlaySystem command plays a system sound from the registry.

The system sound specified must fit into available physical memory and be playable by an installed waveform-audio device driver. If the sound cannot be found the default system sound is played instead. If the default entry can't be found no sound is made and *ok_var* is set to false.

Parameters

The following table describes the parameters of the Sound PlaySystem command:

Parameter	Description
<i>sound_name</i>	The name of the system sound to be played
<i>async</i>	Set to true if the command should return immediately without waiting for the sound to finish. This is true by default.
<i>ok_var</i>	The name of a variable to be set to true if the sound plays correctly.

Example

Play the sound associated with the Exclamation icon on a message box

```
Sound PlaySystem "SystemExclamation"
```

Related Script Commands

[Sound Play](#), [Sound Loop](#), [Sound Stop](#)

Version

4.1.1

Sound Stop

Syntax

Sound Stop

Synonyms

SD ST

Description

The Sound Stop command stops a sound that is playing asynchronously with the Sound Play command or looping from the Sound Loop command.

Parameters

None

Example

Play "CantAllow.wav" sound repetively for 2 minutes

```
* MySoundDir is a variable set to the folder containing the sounds
Sound Loop MySoundDir:"CantAllow.wav"
Wait Delay "2m"
Sound Stop
```

Related Script Commands

[Sound Loop](#), [Sound Play](#), [Sound PlaySystem](#)

Version

4.1.1 Original

Stamp

Syntax

Stamp

Description

The Stamp function returns the current time and date in the format shown in the example below.

Parameters

None

Return Value

A textual representation of the current time and date

Example

The following example prints "The time and date is 12:00:45 18 December 1995" in a message box.

```
MessageBox "The time and date is " : Stamp()
```

Related Script Commands

[Date](#), [Iconv](#), [Oconv](#), [Time](#)

StopList

Syntax

StopList

Description

The StopList command stops a listing started by the Files, Fields, or Items commands.

The StopList command is a Server Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

None

Example

The following example stops a file list and then restarts it:

```
* The following subroutine stops a file list in mid flow and restarts
it.
Sub ReLoadFiles()
Library 'wintsys\lib\server'
StopList ;* Stop the current files command
Files Util.FileList
EndSub
```

Store

Syntax

Store *variable, assignment, value*

Synonyms

ST

Description

The Store command stores a wIntegrate session variable that the Update command invokes. You can use this to reduce screen updates when you are changing session parameters.

See the "Reference - Script Menu Options" in this manual for details of the menu options and variables available.

Parameters

The following table describes the parameters of the Store command:

Parameter	Description
<i>variable</i>	The session variable name.
<i>assignment</i>	The assignment operator. The default is =.
<i>value</i>	The value or list of values for the variable.

Example

The following example is from the LOGON.WIS script.

```
DialogBox Show LearnInfo
If ReturnValue = "OK" Then
CreateLearn
If Not(Get(PortOpen)) Then
Store PortOpen = True;Update
Wait Delay "1s"
Wait RxEmpty
EndIf
```

Related Script Commands

[Configure](#), [Set](#), [Update](#)

StoreAccess

Syntax

StoreAccess

Description

The StoreAccess command puts the server in a mode where it will store all the ECLcommands sent to it using the Enter command. This is kept for backward compatability, use the MultExec library command for new scripts.

The StoreAccess command is a Server Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

None

Related Script Commands

[ExecAccess](#), [MultiExec](#)

Version

4.0 Original

String

Syntax

String *string*, *repeat_count*

Description

The String function returns a copy of a string.

Parameters

The following table describes the parameters of the String command:

Parameter	Description
<i>string</i>	The string to repeat.
<i>repeat_count</i>	The number of times to repeat the string.

Return Value

repeat_count copies of the string concatenated together

Example

The following example prints "wIntegrate * wIntegrate * wIntegrate *" on the status bar.

```
Print "" : (String("wIntegrate *", 3))
```

Strip

Syntax

Strip *string*

Description

The Strip function returns a copy of string which contains only ASCII characters between 32 and 126.

Parameters

The following table describes the parameters of the Strip command:

Parameter	Description
<i>string</i>	The string or text to return as ASCII characters.

Return Value

The string stripped of characters with ascii values less than 32 or greater than 126.

Example

The following example is from the LOGON.WIS script.

```
While n > 0 And Strip(Extract(text,n)) = ""  
n -= 1  
Repeat  
If n = 0 Then  
Learn.prompt = ""  
Else Learn.prompt = Strip(Extract(text,n))
```

Sub

Syntax

Sub *sub_name*([*arg1*], [*arg2*], ... [*argN*])

Description

The Sub command defines a user-defined subroutine. You must use it after the EndScript command. Finish the definition of a subroutine with the EndSub command. The arguments are variables whose values are passed to the subroutine.

Use the Local command to create a variable for use in this subroutine only.

Parameters

The following table describes the parameters of the Sub command:

Parameter	Description
<i>sub_name</i>	The name of the subroutine.
<i>arg1, arg2 ... argN</i>	Optional arguments in the subroutine. If the argument begins with \$, a variable is set with a name. This kind of argument can be passed by reference so that the original variable can change.

Example

The following example is part of the WC.WIS demonstration program.

```
Sub RunCmnd()  
pos = Locate(DialogList(wc,Cmnd), wc.Cmnd, "EU")  
If pos = 0 Then DialogBox AddToList wc,cmnd,wc.cmnd,1  
Execute wc.Cmnd  
If IsShown(wc) Then DialogBox SetText wc,Cmnd, ''  
EndSub
```

Related Script Commands

[Define](#), [EndScript](#), [EndSub](#), [Local](#), [Return](#)

SystemColor

Syntax

SystemColor *option*

Description

This function returns the color used for various parts of the Windows Desktop.

Parameters

The following table describes the parameters of the SystemColor command:

Parameter	Description
<i>option</i>	Specifies the color to return. See table below

Values for option

Value	Description
<i>0</i>	Scroll Bar
<i>1</i>	Desktop
<i>2</i>	Active Caption
<i>3</i>	Inactive Caption
<i>4</i>	Menu background
<i>5</i>	Window background
<i>6</i>	Window Frame
<i>7</i>	Menu text
<i>8</i>	Window text
<i>9</i>	Caption text
<i>10</i>	Active window border
<i>11</i>	Inactive window border
<i>12</i>	Background for MDI workspaces

Value	Description
13	Selected item in control
14	Selected item text in a control
15	Face color for 3D elements
16	Shadow color for 3D elements
17	Disabled/Gray text
18	Text on push buttons
19	Inactive caption text
20	Highlight color for 3D elements
21	Dark shadow for 3D elements
22	Light color for 3D elements
23	Text color for tooltip controls
24	Background color for tooltip controls
26	Hot tracked item (Windows 98/2000)
27	Active caption gradient right (Windows 98/2000)
28	Inactive caption gradient right (Windows 98/2000)

Return Value

The return is the color number for the specified system element.

Example

Set the text in a control the same colors as tool tips

```
Dlg.Ctrl.ForeColor = SystemColor(23)
Dlg.Ctrl.BackColor = SystemColor(24)
```

Also see the example script DeskInfo.wis

Version

4.1.1 Original

SystemFolder

Syntax

SystemFolder

Description

This function returns the specified folder used by the system.

Parameters

None

Return Value

A number specifying the folder to retrieve. Must be 5 to retrieve the personal folder for the user (Usually the folder used to store "My Documents")

Related Script Commands

[AppFolder](#)

Version

5.0 Original

SystemFont

Syntax

SystemFont *option*

Description

This function returns the fonts used by the Windows Desktop. The value returned can be used directly to set the font for a dialog box control

Parameters

The following table describes the parameters of the SystemFont command:

Parameter	Description
<i>option</i>	Specifies the font to return. See table below

Values for option

Value	Description
0	Default GUI
1	System
2	Oem fixed pitch
3	Ansi fixed pitch
4	Ansi variable pitch
5	Caption
6	Small caption
7	Menu
8	Status bar/tooltips
9	Message box

Return Value

The return is a comma separated value containing the font name, the point size and the style.

Example

Set the text in a control the same as used in message boxes

```
Dlg.Ctrl.Font = SystemFont(9)
```

Also see the example script DeskInfo.wis

Version

4.1.1 Original

SystemMetrics

Syntax

SystemMetrics *option*

Description

The SystemMetrics function returns the size of various components of a window depending on the option.

Parameters

The following table describes the parameters of the SystemMetrics command:

Parameter	Description
<i>option</i>	See the following table

Values for option

Value	Description
0	Width of desktop.
1	Depth of desktop.
2	Width of Vert. scroll bar.
3	Height of Horizontal. scroll bar.
4	Height of caption with non-sizing border.
5	Width of non-sizing border.
6	Height of non-sizing border.
7	Width of dialog box border. (With WS_DLGFRAME style.)
8	Height of dialog box border. (With WS_DLGFRAME style.)

Return Value

The measurement requested by the option parameter.

Example

The following example centers terminal window horizontally on desktop.

```
width = Field(WindowPos(0),",",3)
x = (SystemMetrics(0) - width)/2
```

Related Script Commands

[WindowPos](#)

T

Syntax

T *text*, [*table*]

Description

The T function returns text that has been translated using a specified translation table. If the text cannot be translated, it returns the inputted text.

Parameters

The following table describes the parameters of the T command:

Parameter	Description
<i>text</i>	Text to translate.
<i>table</i>	Table handle to use for the translation. Default is table specified by the Translation Use command.

Return Value

The translated text

Example

The following example shows the use of the T function:

```
MessageBox T("Invalid Entry"), T("Customer maintenance")
```

Related Script Commands

[Translation Load](#), [Translation Unload](#), [Translation Use](#)

Time

Syntax

Time

Description

The Time function returns the internal time, which is based on the number of seconds since midnight.

Parameters

None

Return Value

The time as the number of seconds since midnight

Example

The following example gives the internal time in a message box.

```
MessageBox "It is now " : Time()
```

Related Script Commands

[Date](#), [Iconv](#), [Oconv](#), [Stamp](#)

ToolSpace

Syntax

ToolSpace

Description

The ToolSpace command leaves a blank space between icons or groups of icons in a tool bar.

The ToolSpace command is a Tool Library command. For more information, see "Appendix A -Scripting Libraries and Modules".

Parameters

None

Example

The following example is a tool bar script which was created using the same icons as wIntegrate's Large Icon Bar.

```
* Big Tool bar
If IsShown(BigTool_bar) Then EndScript
If Not(IsDialog(BigTool_bar)) Then
Library 'wintsys\lib\toollib'
CreateToolBar BigTool_bar,1,6,"L"
AddTool 'icon\excel.ico','Dialog RunProgram;Set Filename =
"excel.exe";Set
Arguments = "";Invoke'
AddTool 'icon\word.ico','Dialog RunProgram;Set Filename =
"winword.exe";Set
Arguments = "";Invoke'
AddTool 'icon\control.ico','Invoke FileControlPanel'
AddTool 'icon\write.ico','Dialog RunProgram;Set Filename =
"write.exe";Set
Arguments = "";Invoke'
AddTool 'icon\pbrush.ico','Dialog RunProgram;Set Filename =
"pbrush.exe";Set
Arguments = "";Invoke'
AddTool 'icon\help.ico','Invoke HelpIndex'
ToolSpace
EndCreate
EndIf
DialogBox Window BigTool_bar
```

Related Script Commands

[AddTool](#), [CreateToolBar](#), [DialogBox Window](#)

Translation Load

Syntax

Translation Load *name, filename*

Synonyms

TN LD

Description

The Translation Load command loads in a specified translation file. The filename is an optional path + the 4 hex digit translation code.

A translation files must have the following format:

Line 1 - Description of file.

Line 2 - Build number (this is used to ensure files are up to date).

Line 3 - n - Translated Words/Phrases.

Each translation has a key file (filename0900.txt) and one or more language files (filename lang_id.txt).

Line 2 in the key and translation files must contain the same number. Each subsequent line in the translated file matches the same line in the key file.

If the translation file is missing or the build number does not match, the translations file is not loaded so the T function will return the inputted text.

Parameters

The following table describes the parameters of the Translation Load command:

Parameter	Description
<i>name</i>	Handle for this translation table.
<i>filename</i>	Prefix for translation files.

Example

* If language is french loads key table quer0900.txt and translation quer0c01.txt

```
Translation Load Query, ScriptDir:"quer"
```

Related Script Commands

[T](#), [Translation Unload](#), [Translation Use](#)

Translation Unload

Syntax

Translation Unload *name*

Synonyms

TN UL

Description

The Translation Unload command unloads a translation table from memory. The table is only unloaded when the total number of Translation Unload commands matches the total number of Translation Load commands.

Parameters

The following table describes the parameters of the Translation Unload command:

Parameter	Description
<i>name</i>	Handle to previously loaded translation table.

Example

Translation Unload Query

Related Script Commands

[T](#), [Translation Load](#), [Translation Use](#)

Translation Use

Syntax

Translation Use *name*

Synonyms

TN US

Description

The Translation Use command sets the default translation table for the current script to use.

Parameters

The following table describes the parameters of the Translation Use command:

Parameter	Description
<i>name</i>	The translation table to set as the default.

Example

The following example specifies to use the table "Query" to translate the word "SORT":

```
Translation Use Query
Verb = T("SORT")
```

Related Script Commands

[T](#), [Translation Load](#), [Translation Unload](#)

Trigger Add

Syntax

Trigger Add *name, type, left, top, right, bottom, cmdnd, [match]*

Synonyms

TR AD

Description

The Trigger Add command creates a trigger that runs a script command when the specified area of the screen changes.

Parameters

The following table describes the parameters of the Trigger Add command:

Parameter	Description
<i>name</i>	The name of the trigger.
<i>type</i>	The type of trigger. See the following Trigger Types table.
<i>left</i>	Left column of the area to be watched.
<i>top</i>	Top row of the area to be watched.
<i>right</i>	Right column of the area to be watched.
<i>bottom</i>	Bottom row of the area to be watched.
<i>cmdnd</i>	The script command to run when the screen area changes.

Parameter	Description
<i>match</i>	The matching pattern. The format of the pattern string is the following: num1 type1 Number (num) stands for the number of repetitions of the type of character in the type field. If num is not set, it defaults to 1. If it is set to 0, it searches for zero or more occurrences of the type. Type is specified by "A" for alphabetic characters, "N" for numeric, and "X" for any character. You can also use a literal text string inside of quotes. For example, "0X'Error'0X" searches for any number of characters followed by the text "Error" followed by any number of characters.

Values for type

Value	Description
<i>0</i>	Run trigger when screen area changes.
<i>1</i>	Run trigger when screen text equals the text specified in match.
<i>2</i>	Run trigger when screen text matches the pattern specified in match.
<i>3</i>	Run trigger when one of the lines in screen text equals the text specified in match.
<i>4</i>	Run trigger when one of the lines in screen text matches the pattern specified in match.

Example

The following example creates a trigger that runs a script whenever the top left of the screen changes:

```
Trigger Add prog_change, 0, 0,0,20,0, "Script 'ProgChng' "
```

In the next example, the program segment displays a message box whenever the word "Error" is displayed anywhere on line 23 of the screen:

```
Trigger Add errbox, 2, 0,23,79,23, "MessageBox Trim(Screen-Text(
0,23,79,23,1))", "0X'Error'0X"
```

In the next example, the program segment prints "At ECL" whenever any line of the screen starts with ": " and the cursor is in column 1:

```
Trigger Add ECL, 3, 0,0,1,23, "If Cursor(Get_X) = 1 Then;Print 'At
ECL;Else Print '": " "
```

Related Script Commands

[IsTrigger](#), [Trigger Delete](#), [Trigger DeleteAll](#)

Trigger Delete

Syntax

Trigger Delete *name*

Synonyms

TR DL

Description

The Trigger Delete command deletes a trigger by name.

Parameters

The following table describes the parameters of the Trigger Delete command:

Parameter	Description
<i>name</i>	The name of the trigger to delete.

Example

The following example deletes the trigger named "errbox":

```
Trigger Delete errbox
```

Related Script Commands

[IsTrigger](#), [Trigger Add](#), [Trigger DeleteAll](#)

Trigger DeleteAll

Syntax

Trigger DeleteAll

Synonyms

TR DA

Description

The Trigger DeleteAll command deletes all currently defined triggers.

Parameters

None

Related Script Commands

[IsTrigger](#), [Trigger Add](#), [Trigger Delete](#)

Trim

Syntax

Trim *string*, [*option*]

Description

The Trim function removes the spaces from a string.

Parameters

The following table describes the parameters of the Trim command:

Parameter	Description
<i>string</i>	The string to trim spaces from.
<i>option</i>	The option to use. Options are cumulative and the default is 7. See below

Values for option

Add together the following values

Value	Description
<i>1</i>	Remove leading spaces
<i>2</i>	Remove trailing spaces
<i>4</i>	Replace two or more spaces with a single space

Return Value

A copy of the string with the specified spaces removed

Example

The following example trims all the extra spaces and prints "Run wIntegrate".

```
Print (Trim("  Run      wIntegrate  "))
```

Type

Syntax

Type *string*, [*translate*]

Synonyms

TY

Description

The Type command sends text to the host.

Parameters

The following table describes the parameters of the Type command:

Parameter	Description
<i>string</i>	Data to send to the host.
<i>translate</i>	Translates the string from backslash format if set to "True". The default is "True".

Example

The following example is taken from the AP.WIS demonstration program.

```
Sub ListClick()  
If AP_Keys.ShowKeys = "Normal" Then  
    If AP_Keys.KeyList # "" Then Type Char(Asc(AP_Keys.KeyList)-64)  
Else If AP_Keys.ShowKeys = "File" Then  
    If AP_Keys.KeyList # "" Then Type Char(Asc("X")-64) : Left(  
AP_Keys.KeyList, 1)  
Else If AP_Keys.ShowKeys = "Cut" Then  
    If AP_Keys.KeyList # "" Then Type Char(Asc("C")-64) : Left(  
AP_Keys.KeyList, 1)
```

Related Script Commands

[Enter](#)

Until

Syntax

Until *expression*

Description

The Until command continues the current loop specified by the Loop and Repeat commands until the expression is true.

When the expression becomes true execution of the script will continue from the next statement following the Repeat statement that marks the end of the loop.

Parameters

The following table describes the parameters of the Until command:

Parameter	Description
<i>expression</i>	The expression to evaluate. When this evaluates to true the loop will be exited.

Related Script Commands

[Loop](#), [Repeat](#), [While](#)

Update

Syntax

Update [*err*]

Synonyms

UP

Description

The Update command updates the session variables stored with the Store command or assigned after a Configure command.

If the script is in configure mode, Update will also exit the configuration mode.

See the "Reference - Script Menu Options" in this manual for details of the menu options and variables available.

Parameters

The following table describes the parameters of the Update command:

Parameter	Description
<i>err</i>	This parameter specifies an optional variable that will receive an error code if the update fails or 0 if the update succeeds. If this variable isn't specified the script error dialog is shown on an error. The text for the error can be retrieved with the LastErrorText function.

Example

The example below is part of the MACIMP.WIS demonstration program.

```
Dialog RunImportFile
Configure
Account=""
File="WIN.PROGS"
Items="MACHINE.TYPES"
Fields="*"
DosFile="C:\WINTeG\MACTYPES.TXT"
```

```
Format="ASC"  
Inform="0"  
Timeout="5"  
Retries="3"  
AutoExit="1"  
Translate="Ascii"  
Translation="\255,\r\n\f\r\n,\254,\r\n"  
Overwrite="Yes"  
SuppressId="0"  
Mode="Normal"  
FieldDescriptions="0"  
Update  
Invoke
```

Related Script Commands

[Configure](#), [Set](#), [Store](#), [LastErrorText](#)

Version

5.2 Added error variable

UserGlobal

Syntax

UserGlobal *var_name*, [= *value*], ...

Synonyms

UG

Description

The UserGlobal command creates up to five global script variables that will be saved when the session is saved and restored when the session is next loaded. While the session is running these variables are identical to the variables created using the Global script command. Multiple UseGlobal statements can be used to create additional variables.

Use the Destroy command to remove the global variables from memory.

Parameters

The following table describes the parameters of the UserGlobal command:

Parameter	Description
<i>var_name</i>	The name of the variable to create
= <i>value</i>	The initial value for the variable. If this is omitted the variables value is null.
...	Up to a total of five variable can be created.

Example

Set up a variable to be saved with the users name

```
UserGlobal UserName = "David Robertshaw"
```

Set open/saved title processing for Query Builder

```
* After the following has been saved to a session file the
* Query Builder will show the name of the last opened/saved
* query in its title bar
```

UserGlobal Query_FileTitle=true

Version

4.2.1 Original

VarType

Syntax

VarType *type, value*

Description

The VarType function returns a string which specifies the type of an Automation variable.

Parameters

The following table describes the parameters of the VarType command:

Parameter	Description
<i>type</i>	See list of types below
<i>value</i>	The value to be set to the specific type

Values for type

Use the quoted name or the number in parenthesis to specify the type

Value	Description
"NULL" (1)	SQL style Null. Does not require a value
"I2" (2)	2 byte signed integer
"I4" (3)	4 byte signed integer
"R4" (4)	4 byte real number
"R8" (5)	8 byte real number
"CY" (6)	Currency
"DATE" (7)	Date
"BSTR" (8)	String
"DISPATCH" (9)	Dispatch pointer. The value is the name of a previously defined object
"ERROR" (10)	Error code

Value	Description
" <i>BOOL</i> " (11)	Boolean value (True = -1, False = 0)
" <i>OPT</i> " (none)	Sets up variable as an optional parameter. Value is ignored

Return Value

A string that is prefixed with a code used in the automation routines

Example

Set up a variable which can be used in the Microsoft Word `execute` method for the optional parameters

```
optParam = VarType("OPT")
Find.Execute optParam,optParam,optParam,optParam,optParam,optParam,
True, wdFindContinue, optParam, optParam, wdReplaceAll
```

Create a new Excel chart from a range of cells

```
Object Set range = worksheet.Range("A1:D2")
Object Set chart = workbook.Charts.Add()

* We have to pass the ChartWizard method the range objects
* dispatch pointer. Use VarType to retrieve it.
chart.ChartWizard VarType("DISPATCH", "range"), xl3DPie,7, xlRows, 1,
0,2, "Sales Percentages"
```

Version

4.0.1 Original version

VirtualScreen Background

Syntax

VirtualScreen Background *name, back_name, [error_var]*

Synonyms

VS BK

Description

This command sets the background for a virtual screen. This is the area that is behind the window in the virtual screen.

It is used to cover the areas of the virtual screen when it is moved or resized and to paint any area showed by panning that is in the window but not on the virtual screen. To create a background use the screen store command to save an area of the main screen.

Parameters

The following table describes the parameters of the VirtualScreen Background command:

Parameter	Description
<i>name</i>	The name of the virtual screen.
<i>back_name</i>	The name of a previously stored area of the main screen
<i>error_var</i>	The name of a variable to receive any errors that may occur. If this field is omitted any errors will show the standard script error dialog box. See VirtualScreen New command for a list of possible values.

Related Script Commands

[VirtualScreen New](#)

Version

4.1.1 Original

VirtualScreen Delete

Syntax

VirtualScreen Delete *name*

Synonyms

VS DL

Description

This screen deletes a virtual screen. If a virtual screen with the given name does not exist no error will occur.

Parameters

The following table describes the parameters of the VirtualScreen Delete command:

Parameter	Description
<i>name</i>	The name of the virtual screen.

Version

4.1.1 Original

VirtualScreen DeleteAll

Syntax

VirtualScreen DeleteAll

Synonyms

VS DA

Description

This command deletes all the virtual screens currently defined.

Parameters

None

Version

4.1.1 Original

VirtualScreen MoveWindow

Syntax

VirtualScreen MoveWindow *name, by_x, by_y, border, [error_var]*

Synonyms

VS MW

Description

This command moves the window on the main screen through which the virtuals screen is seen.

Parameters

The following table describes the parameters of the VirtualScreen MoveWindow command:

Parameter	Description
<i>name</i>	The name of the virtual screen.
<i>by_x</i>	The number of columns to move the window by. Negative numbers move left, positive move right
<i>by_y</i>	The number of rows to move the window by. Negative numbers move up, positive move down
<i>border</i>	Paint the screen with the virtual spaces background from a position border rows/columns on the opposite side(s) to the direction it moved to its new position. Set to 0 if no border painting is required.
<i>error_var</i>	The name of a variable to receive any errors that may occur. If this field is omitted any errors will show the standard script error dialog box. See VirtualScreen New command for a list of possible values.

Related Script Commands

[VirtualScreen New](#)

Version

4.1.1 Original

VirtualScreen New

Syntax

VirtualScreen New *name, width, depth, [error_var]*

Synonyms

VS NW

Description

This command create a new virtual screen.

A virtual screen is a separate area of screen memory from the main screen that can be updated independently. A window into the virtual screen can be provided on the main screen that will show a section of the virtual screen.

Parameters

The following table describes the parameters of the VirtualScreen New command:

Parameter	Description
<i>name</i>	The name for the new virtual screen.
<i>width</i>	The new width of the virtual screen
<i>depth</i>	The new depths of the virtual screen
<i>error_var</i>	The name of a variable to receive any errors that may occur. If this field is omitted any errors will show the standard script error dialog box. See below for a list of possible values.

Values for error_var

Value	Description
<i>0</i>	No error
<i>1</i>	A virtual screen with this name already exists

Value	Description
2	A virtual screen with this name does not exist
3	Unable to allocate enough memory to create the virtual screen
4	The virtual screen size specified is too large
5	Invalid co-ordinates passed to function

Version

4.1.1 Original

VirtualScreen Pan

Syntax

VirtualScreen Pan *name, by_x, by_y, [error_var]*

Synonyms

VS PN

Description

This command changes which part of the virtual screen can be seen through the window on the main screen.

Parameters

The following table describes the parameters of the VirtualScreen Pan command:

Parameter	Description
<i>name</i>	The name of the virtual screen.
<i>by_x</i>	The amount to move the left side of the virtual screen
<i>by_y</i>	The amount to move the top side of the virtual screen
<i>error_var</i>	The name of a variable to receive any errors that may occur. If this field is omitted any errors will show the standard script error dialog box. See VirtualScreen New command for a list of possible values.

Related Script Commands

[VirtualScreen New](#)

Version

4.1.1 Original

VirtualScreen PrintOff

Syntax

VirtualScreen PrintOff *name, restore_back, [error_var]*

Synonyms

VS PO

Description

This command redirects output back to the screen after a VirtualScreen PrintOn command.

Parameters

The following table describes the parameters of the VirtualScreen PrintOff command:

Parameter	Description
<i>name</i>	The name of the virtual screen.
<i>restore_back</i>	If true the window is repainted with the background for the virtual screen.
<i>error_var</i>	The name of a variable to receive any errors that may occur. If this field is omitted any errors will show the standard script error dialog box. See VirtualScreen New command for a list of possible values.

Related Script Commands

[VirtualScreen New](#), [VirtualScreen PrintOn](#)

Version

4.1.1 Original

VirtualScreen PrintOn

Syntax

VirtualScreen PrintOn *name, show, [error_var]*

Synonyms

VS PR

Description

This command redirects screen output to a virtual screen.

While output is directed to the virtual screen all emulation commands, the color script command and the Display command all effect the virtual screen instead of the main screen.

Note: The screen commands still effect the main screen.

Parameters

The following table describes the parameters of the VirtualScreen PrintOn command:

Parameter	Description
<i>name</i>	The name of the virtual screen.
<i>show</i>	Set to true to update the main screen when any updates to the virtual screen effect an area within its window
<i>error_var</i>	The name of a variable to receive any errors that may occur. If this field is omitted any errors will show the standard script error dialog box. See VirtualScreen New command for a list of possible values.

Related Script Commands

[VirtualScreen New](#), [VirtualScreen PrintOff](#)

Version

4.1.1 Original

VirtualScreen Resize

Syntax

VirtualScreen Resize *name, new_width, new_depth, [error_var]*

Synonyms

VS RS

Description

This command set a new width and depth for a virtual screen.

Parameters

The following table describes the parameters of the VirtualScreen Resize command:

Parameter	Description
<i>name</i>	The name of the virtual screen.
<i>new_width</i>	The new width for the virtual screen
<i>new_depth</i>	The new depth for the virtual screen
<i>error_var</i>	The name of a variable to receive any errors that may occur. If this field is omitted any errors will show the standard script error dialog box. See VirtualScreen New command for a list of possible values.

Related Script Commands

[VirtualScreen New](#)

Version

4.1.1 Original

VirtualScreen ResizeWindow

Syntax

VirtualScreen ResizeWindow *name, by_x, by_y, border, [error_var]*

Synonyms

VS RW

Description

This command resizes the window on the main screen through which the virtuals screen is seen.

Parameters

The following table describes the parameters of the VirtualScreen ResizeWindow command:

Parameter	Description
<i>name</i>	The name of the virtual screen.
<i>by_x</i>	The amount to increase the width of the window. Negative numbers reduce the width
<i>by_y</i>	The amount to increase the depth of the window. Negative numbers reduce the depth
<i>border</i>	Paint the screen with the virtual spaces background from a position border rows/columns on the opposite side(s) to the direction it moved to its new position. Set to 0 if no border painting is required.
<i>error_var</i>	The name of a variable to receive any errors that may occur. If this field is omitted any errors will show the standard script error dialog box. See VirtualScreen New command for a list of possible values.

Related Script Commands

[VirtualScreen New](#)

Version

4.1.1 Original

VirtualScreen Restore

Syntax

VirtualScreen Restore *name*, *store_name*, [*left*], [*top*], [*right*], [*bottom*], [*error_var*]

Synonyms

VS RT

Description

This command restores an area of the virtual screen from a memory variable.

These memory areas can be created with the VirtualScreen Store or Screen Store commands. If the left,top,right or bottom parameters are omitted the values from the stored screen will be used.

Parameters

The following table describes the parameters of the VirtualScreen Restore command:

Parameter	Description
<i>name</i>	The name of the virtual screen.
<i>store_name</i>	The name for the stored area
<i>left</i>	The left side of the area to be stored
<i>top</i>	The top of the area to be stored
<i>right</i>	The right side of the area to be stored
<i>bottom</i>	The bottom of the area to be stored
<i>error_var</i>	The name of a variable to receive any errors that may occur. If this field is omitted any errors will show the standard script error dialog box. See VirtualScreen New command for a list of possible values.

Related Script Commands

[VirtualScreen New](#), [VirtualScreen Store](#), [Screen Store](#)

Version

4.1.1 Original

VirtualScreen Scroll

Syntax

VirtualScreen Scroll *name, left, top, right, bottom, by_x, by_y, [error_var]*

Synonyms

VS SC

Description

This command scrolls an area of the virtual screen.

Parameters

The following table describes the parameters of the VirtualScreen Scroll command:

Parameter	Description
<i>name</i>	The name of the virtual screen.
<i>left</i>	The left side of the area to be scrolled
<i>top</i>	The top of the area to be scrolled
<i>right</i>	The right side of the area to be scrolled
<i>bottom</i>	The bottom of the area to be scrolled
<i>by_x</i>	The number of columns to scroll the area. Positive scrolls right, negative scrolls left
<i>by_y</i>	The number of rows to scroll the area. Positive scrolls down, negative scrolls up
<i>error_var</i>	The name of a variable to receive any errors that may occur. If this field is omitted any errors will show the standard script error dialog box. See VirtualScreen New command for a list of possible values.

Related Script Commands

[VirtualScreen New](#)

Version

4.1.1 Original

VirtualScreen Store

Syntax

VirtualScreen Store *name, store_name, left, top, right, bottom, [error_var]*

Synonyms

VS ST

Description

This command store an area of the virtual screen to a memory variable.

These area are the same as those used by the Screen store command.

Parameters

The following table describes the parameters of the VirtualScreen Store command:

Parameter	Description
<i>name</i>	The name of the virtual screen.
<i>store_name</i>	The name for the stored area
<i>left</i>	The left side of the area to be stored
<i>top</i>	The top of the area to be stored
<i>right</i>	The right side of the area to be stored
<i>bottom</i>	The bottom of the area to be stored
<i>error_var</i>	The name of a variable to receive any errors that may occur. If this field is omitted any errors will show the standard script error dialog box. See VirtualScreen New command for a list of possible values.

Related Script Commands

[VirtualScreen New](#), [VirtualScreen Restore](#), [Screen Store](#)

Version

4.1.1 Original

VirtualScreen Window

Syntax

VirtualScreen Window *name*, *left*, *top*, *right*, *bottom*, *offset_x*, *offset_y*, *show*, [*error_var*]

Synonyms

VS WN

Description

This command sets a window on the screen through which the data from a virtual screen can be viewed.

Parameters

The following table describes the parameters of the VirtualScreen Window command:

Parameter	Description
<i>name</i>	The name of the virtual screen.
<i>left</i>	The left side of the window on the screen
<i>top</i>	The top of the window on the screen
<i>right</i>	The right side of the window on the screen
<i>bottom</i>	The bottom of the window on the screen
<i>offset_x</i>	The position of the left side of the virtual screen in relation to the left side of the window on the screen
<i>offset_y</i>	The position of the top of the virtual screen in relation to the top of the window on the screen
<i>show</i>	True to show the data from the virtual screen
<i>error_var</i>	The name of a variable to receive any errors that may occur. If this field is omitted any errors will show the standard script error dialog box. See VirtualScreen New command for a list of possible values.

Related Script Commands

[VirtualScreen New](#)

Version

4.1.1 Original

VSAttribute

Syntax

VSAttribute *name*, *Options*, [*x*], [*y*], [*page*]

Description

This function returns the current screen attribute or the attribute at the specified location from a virtual screen.

Parameters

The following table describes the parameters of the VSAttribute command:

Parameter	Description
<i>name</i>	The name of the virtual screen
<i>Options</i>	See below
<i>x</i>	x co-ordinate to read attribute from. Default current position of cursor
<i>y</i>	y co-ordinate to read attribute from. Default current position of cursor
<i>page</i>	Back page number to check. Default 0

Values for Options

Value	Description
<i>1</i>	Effect (see table below)
<i>2</i>	Font bank (0-3)
<i>4</i>	Foreground color (0-15)
<i>8</i>	Background color (0-15)
<i>12</i>	Foreground color + 16 * background colour

Return Value

The return value is dependent on the option and is in the range specified in the option description above. See below for the meaning of the bits returned for an effect.

The number returned for the effect is made up of the following bits.

Value	Description
0	Normal
1	Dim
2	Reverse
4	Flash
8	Underline
16	Bold
32	Secret

Example

Display the foreground color to be used for the next character to be written to the screen

```
MessageBox VSAttribute(ReportVS,4)
```

Check if the start of the bottom line of the screen is displayed in red

```
If VSAttribute(ReportVS,4,0,24) = COL_Red Then ...
```

Check if the cursor is over a reversed field

```
If VSAttribute(ReportVS, 1,,,0) = 2 Then ...
```

Version

4.1.1 Original

VSCursor

Syntax

VSCursor *name, option*

Description

This function returns the cursor position on the specified virtual screen.

Parameters

The following table describes the parameters of the VSCursor command:

Parameter	Description
<i>name</i>	The name of the virtual screen
<i>option</i>	The value to return. See below

Values for option

Value	Description
<i>0</i>	The x-coordinate of the cursor
<i>1</i>	The y-coordinate of the cursor

Return Value

The co-ordinate as specified by the option parameter

Version

4.1.1 Original

VSInfo

Syntax

VSInfo *name, option, [sub_option]*

Description

This function returns information about the virtual screen.

Parameters

The following table describes the parameters of the VSInfo command:

Parameter	Description
<i>name</i>	The name of the virtual screen
<i>option</i>	The value to return. See below
<i>sub_option</i>	Depends on the option. See option table below. If omitted it defaults to 0

Values for option

The following table shows the value returned for option, sub_option. If sub_option is omitted it is not required

Value	Description
<i>1</i>	Returns true if the screen output has been redirected to this virtual space
<i>2,0</i>	Size of virtual screen as width,depth
<i>2,1</i>	width of virtual screen
<i>2,2</i>	depth of virtual screen
<i>3,0</i>	offset of window on virtual screen as x,y
<i>3,1</i>	x offset of window on virtual screen
<i>3,2</i>	y offset of window on virtual screen
<i>4</i>	Returns true if a window has been defined for this virtual screen

Value	Description
5,0	Returns window co-ordinates as left,top,right,bottom
5,1	left side of the window
5,2	top of the window
5,3	right side of the window
5,4	bottom of the window
6	background name of the virtual screen

Return Value

The requested information about the virtual screen

Version

4.1.1 Original

VSText

Syntax

VSText *name, x1, y1, x2, y2, type*

Description

This function returns text from the specified virtual screen. It only returns character which were are in the main ansi font, other characters are returned as spaces. If multiple lines are copied they are separated by a cr/lf pair.

Parameters

The following table describes the parameters of the VSText command:

Parameter	Description
<i>name</i>	The name of the virtual screen
<i>x1</i>	Start column
<i>y1</i>	Start row
<i>x2</i>	End column. If omitted defaults to the start column
<i>y2</i>	End row. If omitted defaults to the start row
<i>type</i>	How to do the copy:0 - copy all the lines from start point to end point1 - copy the block of text.

Return Value

The text from the virtual screen

Version

4.1.1 Original

Wait Delay

Syntax

Wait Delay *timeout*

Synonyms

WT DL

Description

The Wait Delay command pauses the script for a specified period.

Parameters

The following table describes the parameters of the Wait Delay command:

Parameter	Description
<i>timeout</i>	The time to wait. Specify as a number of centi-seconds to delay the script, or the number of seconds followed by an "s" or the number of minutes followed by an "m".

Example

The following example displays an info box and waits for ten seconds before clearing it from the screen.

```
* Show the install info box, then close it
InfoBox "Program is installing", B
Wait Delay "10s"
InfoBox ""
```

Related Script Commands

[Wait DuringTask](#), [Wait ForText](#), [Wait RxEmpty](#), [Wait TxEmpty](#)

Wait DuringTask

Syntax

Wait DuringTask *task_name*

Synonyms

WT DT

Description

The Wait During Task command waits while the specified task is running.

Parameters

The following table describes the parameters of the Wait DuringTask command:

Parameter	Description
<i>task_name</i>	The name of the task to wait for.

Related Script Commands

[Wait Delay](#), [Wait ForText](#), [Wait RxEmpty](#), [Wait TxEmpty](#)

Wait ForText

Syntax

Wait ForText *text*, [*timeout*], [*found_var*], [*delimiter*]

Synonyms

WT FT

Description

The Wait ForText command waits for the specified text to be received by wIntegrate from the host.

Parameters

The following table describes the parameters of the Wait ForText command:

Parameter	Description
<i>text</i>	The text that is being sent to wIntegrate by the host. You can specify more than one string by separating the strings by delimiter.
<i>timeout</i>	The maximum time to wait for the text. The default is 10 seconds.
<i>found_var</i>	A variable containing the index of the string that was matched in text or 0 if unnecessarily. If not found, this is set to 0.
<i>delimiter</i>	The delimiter to specify more than one text string. The delimiter character must have a code value in the range 0 to 255.

Example

The following example is taken from the LOGON.WIS script found in "wintsys\script"

```
Sub AddSendData(send_data)
If Learn.prompt # "" Then
Learn_Script := cr:lf:"text="':Learn.prompt:''
Learn_Script := cr:lf:"Wait ForText text,,found"
```



```
Learn_Script := cr:lf:"If Not(found) Then LogonError text"  
EndIf  
Learn_Script := cr:lf:"Type ":send_data  
EndSub
```

in the next example, the Wait ForText command searches for two different strings to be returned from the host ("Connect OK" or "Connect Failed") which are separated by the "|" in the statement:

```
Wait ForText "Connect OK|Connect Failed",,found, "|" "  
If found = 2 Then  
  MessageBox "Unable to connect"  
EndIf
```

Related Script Commands

[Wait Delay](#), [Wait DuringTask](#), [Wait RxEmpty](#), [Wait TxEmpty](#)

Wait RxEmpty

Syntax

Wait RxEmpty [*timeout*], [*empty_var*]

Synonyms

WT RE

Description

The Wait RxEmpty command waits until the receive from host buffer is empty.

Parameters

The following table describes the parameters of the Wait RxEmpty command:

Parameter	Description
<i>timeout</i>	The maximum time to wait for the buffer.
<i>empty_var</i>	This parameter returns true if the buffer is empty.

Example

The following example comes from the logon script found in "wintsys\script\LOGON.WIS".

```
* Interactively create logon script
* Version 1.1 (14 July 1994)
* Set up variable to include current script in dialog boxes
DLib = 'Library "':ScriptFile:'";'
CreateLearnInfo ; * Create the learn info dialog box
DialogBox Show LearnInfo
If ReturnValue = "OK" Then
CreateLearn
If Not(Get(PortOpen)) Then
Store PortOpen = True;Update
Wait Delay "1s"
Wait RxEmpty
EndIf
```

Related Script Commands

[Wait Delay](#), [Wait DuringTask](#), [Wait ForText](#), [Wait TxEmpty](#)

Wait TxEmpty

Syntax

Wait TxEmpty [*timeout*], [*empty_var*]

Synonyms

WT TE

Description

The Wait TxEmpty command waits until the transmit to host buffer is empty.

Parameters

The following table describes the parameters of the Wait TxEmpty command:

Parameter	Description
<i>timeout</i>	The maximum time to wait for the buffer.
<i>empty_var</i>	This parameter returns true if the buffer is empty.

Related Script Commands

[Wait Delay](#), [Wait DuringTask](#), [Wait ForText](#), [Wait RxEmpty](#)

WebStyle ControlType

Syntax

WebStyle ControlType *control_type*, *name*

Synonyms

WS CT

Description

This command sets the default web style used for all controls of a specified type.

The web styles are defined in a cascading style sheet and are used by the Java Client only. All other versions ignore this command. The web style allows the fonts and colours for a dialog box or control to be changed.

See the Java Client documentation on how to specify the style sheet used.

Parameters

The following table describes the parameters of the WebStyle ControlType command:

Parameter	Description
<i>control_type</i>	The type of the control to set the web style for. See below.
<i>name</i>	Name of the style from the cascading style sheet

Values for control_type

Value	Description
<i>CheckBox</i>	Check box control
<i>ColorButton</i>	Color button
<i>Combobox</i>	Combo box control
<i>DateTime</i>	Date/Time control

Value	Description
<i>Draw</i>	Drawing control
<i>EditText</i>	Editable field control
<i>Graphic</i>	Graphic control
<i>GraphicButton</i>	Graphic button control
<i>GroupBox</i>	Group box control
<i>Listbox</i>	List box control
<i>Listview</i>	List view control
<i>Progressbar</i>	Progress bar
<i>PushButton</i>	Push button
<i>RadioButton</i>	radio button
<i>Rect</i>	Rectangle control
<i>Scrollbar</i>	Scroll bar control
<i>Tabcontrol</i>	Tab control
<i>Text</i>	Text label control
<i>Treeview</i>	Tree view control
<i>Updownbutton</i>	Up down button

Example

Set all push buttons to the ".greentext" style

```
WebStyle ControlType Button, ".greentext"
```

Related Script Commands

[WebStyle Global](#), [WebStyle Menu](#), [DialogBox](#) [WebStyleName](#)

Version

6.0 Original version

WebStyle Global

Syntax

WebStyle Global *name*

Synonyms

WS GL

Description

This command sets the default web-style for all dialog boxes.

The web styles are defined in a cascading style sheet and are used by the Java Client only. All other versions ignore this command. The web style allows the default fonts and colours for a dialog box to be changed.

See the Java Client documentation on how to specify the style sheet used.

Parameters

The following table describes the parameters of the WebStyle Global command:

Parameter	Description
<i>name</i>	The name of the web style to use. This matches the style name in the cascading style sheet.

Example

Set the default web style to ".MyBlueStyle"

```
WebStyle Global ".MyBlueStyle"
```

Related Script Commands

[WebStyle ControlType](#), [WebStyle Menu](#), [DialogBox WebStyleName](#)

Version

6.0 Original version

WebStyle Menu

Syntax

WebStyle Menu *name*

Synonyms

WS MN

Description

This command sets the web style for all menus in the application.

The web styles are defined in a cascading style sheet and are used by the Java Client only. All other versions ignore this command. The web style allows the fonts and colours for all menus to be changed.

See the Java Client documentation on how to specify the style sheet used.

Parameters

The following table describes the parameters of the WebStyle Menu command:

Parameter	Description
<i>name</i>	The name of the web style to use for all menus

Example

Set the menus to use the ".blue" style

```
WebStyle Menu ".blue"
```

Related Script Commands

[WebStyle Global](#), [WebStyle ControlType](#), [DialogBox WebStyleName](#)

Version

6.0.0 Original version

While

Syntax

While *expression*

Description

The While command continues the current loop specified by the Loop and Repeat commands while the expression is true.

When the expression becomes false execution of the script will continue from the next statement following the Repeat statement that marks the end of the loop.

Parameters

The following table describes the parameters of the While command:

Parameter	Description
<i>expression</i>	The expression to evaluate. If it evaluates to true, the loop continues, otherwise it exits.

Related Script Commands

[Loop](#), [Repeat](#), [Until](#)

WindowPos

Syntax

WindowPos *option*

Description

The WindowPos function returns the size of the current wIntegrate session in pixels depending on the option.

The Terminal Screen is the area in which the host text is displayed. The Session Window is the whole of the wIntegrate window.

The result is returned as a string in the format "top_left_x, top_left_y, bottom_right_x, bottom_right_y".

For options 0 and 2 the top left is always at position 0,0.

Option 1 is equivalent to Get(Window) except that it gives the actual size of the window even when it is maximized.

Parameters

The following table describes the parameters of the WindowPos command:

Parameter	Description
<i>option</i>	See below.

Values for option

Value	Description
<i>0</i>	Returns the position of the inside of the terminal screen.
<i>1</i>	Returns the position of the outside of the terminal screen.
<i>2</i>	Returns the position of the inside of the session window.

Value	Description
3	Returns the position of the outside of the session window.

Return Value

Window rectangle for the specified option in pixels

Example

The following example aligns the KeySmall dialog box outside the bottom left of the current session.

```
window.left_x = Field(WindowPos(3),"",1)
bottom_y = Field(WindowPos(3),"",4)
DialogBox Move KeySmall left_x, bottom_y
```

Related Script Commands

[DialogUnit](#), [SystemMetrics](#)

With

Syntax

With *object_name*

Synonyms

WH

Description

After this command has been executed properties and methods of the object specified can be referenced by a period followed by their name.

A With statement must have a terminating EndWith after the statements where this object is used.

With/EndWith blocks can be nested, EndWith reset the *object_name* to its value before the previous With command.

Parameters

The following table describes the parameters of the With command:

Parameter	Description
<i>object_name</i>	The name of the object. (i.e. Automation object, dialog box or dialog box control)

Example

Set properties on a grid and edit control

```
* MyVariable is a variable and MyGrid and MyEdit are
* Controls in MyDialog.
With MyDialog
.MyVariable = 123
With .MyGrid
.ResizeColToFit 0,6
EndWith
.MyEdit = .MyVariable + 1
EndWith
```

Set sel to selection or whole grid if there is no selection

```
With MyDialog.MyGrid
sel = .SelectInfo()
If sel = "" Then
sel = 1:tab:1:tab:.Rows:tab:.Columns
EndIf
EndWith
```

Related Script Commands

[EndWith](#)

Version

4.0.2 Original

WorkArea

Syntax

WorkArea [*option*]

Description

In the local version and Windows thin client version this function returns the area of the main monitor that is not covered by toolbars.

On the Java client it gives the size of the applet.

Parameters

The following table describes the parameters of the WorkArea command:

Parameter	Description
<i>option</i>	Specifies the information to return. See Table below.

Values for option

Value	Description
<i>0</i>	A comma seperated list of the area in the order left, top, right, bottom. (This is the default value)
<i>1</i>	The size of the area seperated by a comma in the order width, height.

Return Value

The location or size of the desktop work area as a comma seperated list depending on the option specified.

Example

Check if any of the work area height is less than the screen height

```
If Field(WorkArea(1),"",2) < SystemMetrics(1) Then
    MsgBox "The work area height is smaller than the display"
EndIf
```

Related Script Commands

[SystemMetrics](#)

Version

6.0 Original version

Yield

Syntax

Yield

Synonyms

YL

Description

The Yield command yields control of the PC so other wIntegrate processes and Windows applications can process events.

Parameters

None

Example

The following example allow other events while counting to 100 on status bar.

```
j = 0
Loop
j += 1
Print j
Yield
Until j = 100
Repeat
```

Related Script Commands

[Wait Delay](#)

|

Syntax

expr1 | *expr2*

Description

The | operator returns the "binary or" value of the two numbers.

It compares each binary bit of the number and sets the corresponding bit in the result if either bit is set.

This operator is frequently used to combine flag values in other commands.

Note: The script language only understands integer numbers from -2147483648 to 2147483647.

Parameters

The following table describes the parameters of the | command:

Parameter	Description
<i>expr1</i>	An expression that evaluates to a number
<i>expr2</i>	An expression that evaluate to a number

Example

Use the | operator to combine the styles for the FontDemo dialog box

```
DialogBox Create FontDemo 23,16,251,152
Caption "Demo different fonts in dialogs"
Style WS_CAPTION|WS_VISIBLE|WS_SYSMENU|DS_MODALFRAME
```

|=

Syntax

variable |= *value*

Description

The |= assignment used the "binary or" operator to modify the specified variable.

It is equivalent to: `variable = variable | number`

Note: The script language only understands integer numbers from -2147483648 to 2147483647.

Parameters

The following table describes the parameters of the |= command:

Parameter	Description
<i>variable</i>	The name of the variable to modify
<i>number</i>	The number to combine with the value in the variable

Reference - Script Controls

This section details the controls that can be created for use on scripted dialog boxes.

The controls are added to the dialog box using the DialogBox script commands.

There are four types of control:

- Standard - The basic controls introduced with Windows 3.0
- Common - The new controls introduced with Windows 95 and later
- Application - Controls provided by this application
- Active X - Active X controls that are provided with this application.

Note: Third party Active X controls can also be used.

Animate

Description

The Animate control plays back an AVI (Audio Video Interleave - Microsoft's Video for Windows standard) file in the control. The style for this control defaults to `WS_VISIBLE`.

The AVI file must meet the following requirements:

There must be exactly one video stream and it must have at least one frame.

There can be at most two streams in the file (typically the other stream, if present, is an audio stream, although the animation control ignores audio information).

The clip must either be uncompressed or compressed with RLE8 compression.

No palette changes are allowed in the video stream.

This is a Common control.

Styles

The following table describes the styles for the Animate control:

Style	Description
<i>ACS_AUTOPLAY</i>	Automatically start playing the control.
<i>ACS_CENTER</i>	Center animation in control (otherwise control is expanded to fit the animation).
<i>ACS_TRANSPARENT</i>	Background of animation is transparent.
<i>WS_DISABLED</i>	The control is initially disabled

Style	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

Properties

The following table describes the properties for the Animate control:

Property	Description
<i>BackColor</i>	Background color of the control
<i>Enabled</i>	Set or get the enabled state of the control
<i>File</i>	Specifies the filename to use for the animation
<i>Height</i>	Get/Set the Height of the control.
<i>Left</i>	Get/Set left hand edge of the control
<i>Top</i>	Get/Set top edge of the control.
<i>Visible</i>	Set or get the visible state for the control
<i>Width</i>	Get/Set the width of the control.

Methods

The following table describes the methods for the Animate control:

Method	Description
<i>Close</i>	The Close method closes an AVI file.
<i>Open</i>	The Open method opens an AVI file.
<i>Play</i>	The Play method plays an AVI file.

Method	Description
<i>Seek</i>	The Seek method moves to the specified frame.
<i>Stop</i>	The Stop method stops playing the AVI file.

Events

The following table describes the events for the Animate control:

Event	Description
<i>DragEnter</i>	Dragged data has entered the control
<i>DragLeave</i>	Dragged data is leaving a control
<i>DragOver</i>	Data is being dragged over the control
<i>Drop</i>	Data has been dropped on this control
<i>Start</i>	AVI file has started playing
<i>Stop</i>	AVI file has stopped playing.

Version

4.1 Added BackColor Property

Calculator

Description

This control displays a simple calculator on the dialog box.

Define the control using the DialogBox AxControl command with a prog id of "wIntegrate.Calculator.1".

Note: This control traps all keystrokes when it has the focus.

See the example script "example\script\calc.wis".

This is a Active X control.

Styles

The following table describes the styles for the Calculator control:

Style	Description
<i>WS_VISIBLE</i>	The control is visible

Properties

The following table describes the properties for the Calculator control:

Property	Description
<i>DecimalPlaces</i>	The number of decimal places to use with the control
<i>Height</i>	Get/Set the Height of the control.
<i>Left</i>	Get/Set left hand edge of the control
<i>Top</i>	Get/Set top edge of the control.
<i>Value</i>	The current value of the control
<i>Width</i>	Get/Set the width of the control.

Events

The following table describes the events for the Calculator control:

Event	Description
<i>EqualsPressed</i>	The equals key or enter button has been pressed
<i>KeyUp</i>	A key has been released

Version

4.2.1 Original

CheckBox

Description

The CheckBox control is a square check box that 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

This is a Standard control.

Styles

The following table describes the styles for the CheckBox control:

Style	Description
<i>BS_AUTOCHECKBOX</i>	Specifies a check box
<i>BS_BITMAP</i>	Button displays a bitmap instead of text. Set the Bitmap with the Image property.
<i>BS_BOTTOM</i>	Align the caption to the bottom of the button
<i>BS_CENTER</i>	Horizontally center the caption in the button
<i>BS_FLAT</i>	Display the button without 3D effects.
<i>BS_LEFT</i>	Align the caption to the left of the button
<i>BS_LEFTEXT</i>	The caption is place on the left of the check box
<i>BS_MULTILINE</i>	Word wrap the caption if it is longer then the buttons size.
<i>BS_NOTIFY</i>	Enable extra events. Normally a button will only react to the click event. Enabling this check box enable the DblClick, SetFocus and KillFocus events as well.

Style	Description
<i>BS_PUSHLIKE</i>	Give the button the same appearance as a push button. But unlike a push button is stays down when selected.
<i>BS_RIGHT</i>	Align the caption to the right of the button
<i>BS_TOP</i>	Align the caption to the top of the button
<i>BS_VCENTER</i>	Vertically center the caption in the button
<i>WS_DISABLED</i>	The control is initially disabled
<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 <i>WS_TABSTOP</i> style.

Properties

The following table describes the properties for the CheckBox control:

Property	Description
<i>BackColor</i>	Background color of the control
<i>BackStyle</i>	Set or get the background style
<i>Caption</i>	Set or get the current value of the caption of the control.
<i>Check</i>	Set or get the current state of the check box
<i>Enabled</i>	Set or get the enabled state of the control
<i>Font</i>	Set or get the font used with the control.
<i>ForeColor</i>	Set or get the color used for the caption or text of the control

Property	Description
<i>Height</i>	Get/Set the Height of the control.
<i>Image</i>	bitmap file name for the control
<i>Left</i>	Get/Set left hand edge of the control
<i>Top</i>	Get/Set top edge of the control.
<i>Visible</i>	Set or get the visible state for the control
<i>WebStyle</i>	Set/get the web style for this control
<i>Width</i>	Get/Set the width of the control.

Events

The following table describes the events for the CheckBox control:

Event	Description
<i>Click</i>	Left mouse clicked on the control
<i>DblClick</i>	Left mouse double clicked on the control
<i>DragEnter</i>	Dragged data has entered the control
<i>DragLeave</i>	Dragged data is leaving a control
<i>DragOver</i>	Data is being dragged over the control
<i>Drop</i>	Data has been dropped on this control
<i>KillFocus</i>	Control has lost the focus
<i>SetFocus</i>	Control has received the input focus

Version

4.1 Modified to add new styles, events and properties

6.0.0 Added WebStyle property

CheckBox

Description

The CheckBox control displays a list of items with check marks next to them.

The check mark is toggled by clicking on it with the mouse or pressing space when the item has the focus (or is selected).

You can also highlight single or multiple options with the mouse or keyboard.

This is a Application control.

Styles

The following table describes the styles for the CheckBox control:

Style	Description
<i>LBS_DISABLENOSCROLL</i>	Shows the vertical scroll bar even if it is not necessary.
<i>LBS_MULTICOLUMN</i>	Designates multiple columns.
<i>LBS_MULTIPLESEL</i>	Allows multiple items to be selected in the list box.
<i>LBS_NOINTEGRALHEIGHT</i>	If this is not specified, the last line of the list may not completely display.
<i>LBS_NOSEL</i>	Creates a list box where no items can be selected
<i>LBS_NOTIFY</i>	Notifies wIntegrate of clicks.
<i>LBS_SORT</i>	Sorts the contents of the list box alphabetically.
<i>LBS_STANDARD</i>	This is a combination of these styles: WS_BORDER, LBS_SORT, LBS_NOTIFY, WS_SCROLL.
<i>WS_BORDER</i>	Adds a border to the edit box.

Style	Description
<i>WS_DISABLED</i>	The control is initially disabled
<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_HSCROLL</i>	Adds a horizontal scroll bar.
<i>WS_TABSTOP</i>	Tab or back tab moves to the previous or next control with <i>WS_TABSTOP</i> style.
<i>WS_VSCROLL</i>	Adds a vertical scroll bar.

Properties

The following table describes the properties for the CheckBox control:

Property	Description
<i>AnchorIndex</i>	Index of first line when selecting multiple lines
<i>BackColor</i>	Background color of the control
<i>CaretIndex</i>	Index of line with focus
<i>CheckIndex</i>	Set or get the currently checked items by line number.
<i>CheckList</i>	Set or get a list of all the items in the list with their checks.
<i>CheckText</i>	A list of the currently checked items
<i>ColumnWidth</i>	Width of the columns in the list box
<i>Enabled</i>	Set or get the enabled state of the control
<i>Font</i>	Set or get the font used with the control.
<i>ForeColor</i>	Set or get the color used for the caption or text of the control
<i>Height</i>	Get/Set the Height of the control.
<i>HorizontalExtent</i>	Set or get the width in pixels of the list

Property	Description
<i>Left</i>	Get/Set left hand edge of the control
<i>List</i>	all the items in a listbox list or the list part of the combox
<i>SelIndex</i>	The index of the current selection
<i>SelText</i>	The selected text
<i>Top</i>	Get/Set top edge of the control.
<i>TopIndex</i>	Set or get the line number of the top line visible in the list part of
<i>UpdateMode</i>	Specifies the information to store when the dialog box is closed
<i>Visible</i>	Set or get the visible state for the control
<i>Width</i>	Get/Set the width of the control.

Methods

The following table describes the methods for the CheckListBox control:

Method	Description
<i>AddLine</i>	Add a new line to the list
<i>CheckAll</i>	Check or uncheck all lines in a check list box
<i>ClearSel</i>	Clear the selection state
<i>Count</i>	Returns the number of lines in the list
<i>DeleteLine</i>	Delete a line from the list
<i>Empty</i>	Clears all the lines from the list
<i>Find</i>	Find line starting with specified text
<i>FindExact</i>	Searches for the line in the list with the specified text.
<i>GetCheck</i>	Get the checked status of a line from a check list box

Method	Description
<i>GetSel</i>	Get the selection state for a line
<i>InitStorage</i>	Increases the amount of storage used for the list
<i>InsPosFromPoint</i>	Return the line number where a new item would be inserted
<i>InsertLine</i>	Insert a line into the list
<i>ItemFromPoint</i>	Return the index of the line nearest to the specified point
<i>Line</i>	Returns the text of the specified line
<i>SelCount</i>	Return the number of lines selected
<i>SelLine</i>	Selects the line of text which start with the specified text
<i>SelRange</i>	Select a range of lines
<i>SetCheck</i>	Set or clear the check mark against a line
<i>SetSel</i>	Set the selection state for a line

Events

The following table describes the events for the CheckListBox control:

Event	Description
<i>DblClick</i>	Left mouse double clicked on the control
<i>DragEnter</i>	Dragged data has entered the control
<i>DragLeave</i>	Dragged data is leaving a control
<i>DragOver</i>	Data is being dragged over the control
<i>Drop</i>	Data has been dropped on this control
<i>KillFocus</i>	Control has lost the focus
<i>SelCancel</i>	The selection from the list box has been cancelled
<i>SelChange</i>	The selection is about to change

Event	Description
<i>SetFocus</i>	Control has received the input focus

Version

5.1 Original version

ColorButton

Description

This ColorButton control is a push button control that allows you to set the foreground and background colors.

This is a Application control.

Styles

The following table describes the styles for the ColorButton control:

Style	Description
<i>BS_NOTIFY</i>	Enable extra events. Normally a button will only react to the click event. Enabling this check box enable the DblClick, SetFocus and KillFocus events as well.
<i>WS_DISABLED</i>	The control is initially disabled
<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

Properties

The following table describes the properties for the ColorButton control:

Property	Description
<i>BackColor</i>	Background color of the control

Property	Description
<i>Caption</i>	Set or get the current value of the caption of the control.
<i>Enabled</i>	Set or get the enabled state of the control
<i>Font</i>	Set or get the font used with the control.
<i>ForeColor</i>	Set or get the color used for the caption or text of the control
<i>Height</i>	Get/Set the Height of the control.
<i>Left</i>	Get/Set left hand edge of the control
<i>Top</i>	Get/Set top edge of the control.
<i>Visible</i>	Set or get the visible state for the control
<i>WebStyle</i>	Set/get the web style for this control
<i>Width</i>	Get/Set the width of the control.

Events

The following table describes the events for the ColorButton control:

Event	Description
<i>Click</i>	Left mouse clicked on the control
<i>SetFocus</i>	Control has received the input focus
<i>KillFocus</i>	Control has lost the focus
<i>Drop</i>	Data has been dropped on this control
<i>DragOver</i>	Data is being dragged over the control
<i>DragLeave</i>	Dragged data is leaving a control
<i>DragEnter</i>	Dragged data has entered the control
<i>DblClick</i>	Left mouse double clicked on the control

Version

5.1 First version

6.0.0 WebStyle added

ComboBox

Description

The ComboBox control is an edit control with an attached list of options.

There are three styles of ComboBox:

Simple - The edit control and attached list are always displayed.

Drop Down - The list box is shown when the arrow on the right of the edit control is pressed.

Drop Down List - This is the same as the drop down except you can not edit the text in the edit control.

This is a Standard control.

Styles

The following table describes the styles for the ComboBox control:

Style	Description
<i>CBS_AUTOHSCROLL</i>	Automatic horizontal scroll when the cursor reaches the right hand edge of the edit control.
<i>CBS_DISABLENOSCROLL</i>	Shows the vertical scroll bar even if it is not necessary.
<i>CBS_DROPDOWN</i>	List box displays only when the arrow is clicked.
<i>CBS_DROPDOWNLIST</i>	List box displays when arrow is clicked and one option is selected
<i>CBS_NOINTEGRALHEIGHT</i>	If this is not set, the last line in the list part may not display.
<i>CBS_OEMCONVERT</i>	Converts OEM text.

Style	Description
<i>CBS_SIMPLE</i>	List box always displays.
<i>CBS_SORT</i>	Sorts the contents of the list box.
<i>WS_DISABLED</i>	The control is initially disabled
<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 <i>WS_TABSTOP</i> style.
<i>WS_VSCROLL</i>	Adds a vertical scroll bar.

Properties

The following table describes the properties for the ComboBox control:

Property	Description
<i>Dropped</i>	dropped status of the drop down list
<i>Enabled</i>	Set or get the enabled state of the control
<i>Font</i>	Set or get the font used with the control.
<i>Height</i>	Get/Set the Height of the control.
<i>HorizontalExtent</i>	Set or get the width in pixels of the list
<i>Left</i>	Get/Set left hand edge of the control
<i>LimitText</i>	Maximum number of characters allowed
<i>List</i>	all the items in a listbox list or the list part of the combobox
<i>SelIndex</i>	Currently selected item in a list
<i>Text</i>	The text from the control
<i>Top</i>	Get/Set top edge of the control.
<i>TopIndex</i>	Set or get the line number of the top line visible in the list part of

Property	Description
<i>UpdateMode</i>	Specify what to update when a dialog box is closed
<i>Visible</i>	Set or get the visible state for the control
<i>WebStyle</i>	Set/get the web style for this control
<i>Width</i>	Get/Set the width of the control.

Methods

The following table describes the methods for the ComboBox control:

Method	Description
<i>AddLine</i>	Adds a line to the list part of the combo box
<i>Clear</i>	Clears the current selection from the control
<i>Copy</i>	Copy the current selection to the clipboard
<i>Count</i>	Returns the number of lines in the list part of the combo box
<i>Cut</i>	Cut the current selection to the clipboard
<i>DeleteLine</i>	Deletes the specified line
<i>EditSelect</i>	Sets the current selection in the edit part of the combo box.
<i>EditSelectInfo</i>	Get the current selection from the edit text
<i>Empty</i>	Clears all the lines from the list part of the combo box.
<i>Find</i>	Find line starting with specified text
<i>FindExact</i>	Searches for the line in the list with the specified text.
<i>GetItemData</i>	Returns the number associated with the line
<i>InitStorage</i>	Increases the amount of storage used for the list

Method	Description
<i>InsertLine</i>	Inserts a line
<i>Line</i>	Returns the text of the specified line
<i>Paste</i>	Pastes text from the clipboard to the control.
<i>SelLine</i>	Selects the line of text which start with the specified text
<i>SetItemData</i>	Set an number against the specified line of a list box.

Events

The following table describes the events for the ComboBox control:

Event	Description
<i>CloseUp</i>	The dropdown list of the combobox has been closed
<i>DblClick</i>	Left mouse double clicked on the control
<i>DragEnter</i>	Dragged data has entered the control
<i>DragLeave</i>	Dragged data is leaving a control
<i>DragOver</i>	Data is being dragged over the control
<i>Drop</i>	Data has been dropped on this control
<i>DropDown</i>	The dropdown list of the combobox has been dropped
<i>EditChange</i>	The edit section of the combo box has been changed
<i>EditUpdate</i>	The edit section of the combo box has been changed,
<i>ErrSpace</i>	The combo box ran out of space when adding an item
<i>KillFocus</i>	Control has lost the focus

Event	Description
<i>SelChange</i>	The selection has changed
<i>SelEndCancel</i>	The selection from the list box has been cancelled
<i>SelEndOK</i>	A selection from the list box has been chosen
<i>SetFocus</i>	Control has received the input focus

Version

4.1 Modified to add new properties, methods and events

6.0.0 WebStyle added

DateTime

Description

The DateTime control is a control which allows you to enter a date and/or a time.

It can optionally provide a drop down calendar.

This is a Application control.

Styles

The following table describes the styles for the DateTime control:

Style	Description
<i>DTS_CALENDAR</i>	Provide a drop down calendar control
<i>DTS_UPDOWN</i>	Provide a spin control
<i>WS_DISABLED</i>	The control is initially disabled
<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

Properties

The following table describes the properties for the DateTime control:

Property	Description
<i>BackColor</i>	Background color of the control
<i>CustomFormat</i>	Custom format for displaying the date and/or time.
<i>DateMax</i>	Set or get the maximum date and/or time

Property	Description
<i>DateMin</i>	Set or get the minimum date and/or time
<i>Dropped</i>	Set or get dropped status of the calendar
<i>Enabled</i>	Set or get the enabled state of the control
<i>FastEntry</i>	Set or get the fast entry mode
<i>ForeColor</i>	Set or get the color used for the caption or text of the control
<i>FormatType</i>	Set or get the type of the date control
<i>Height</i>	Get/Set the Height of the control.
<i>Left</i>	Get/Set left hand edge of the control
<i>NoEdit</i>	Set or get which fields can not be edited
<i>Optional</i>	Set or get if the date is optional
<i>Top</i>	Get/Set top edge of the control.
<i>ValidMode</i>	Set or get the validation
<i>Value</i>	current value of the date time control.
<i>Visible</i>	Set or get the visible state for the control
<i>WebStyle</i>	Set/get the web style for this control
<i>Width</i>	Get/Set the width of the control.

Events

The following table describes the events for the DateTime control:

Event	Description
<i>Changed</i>	The value of the date time control has changed
<i>DragEnter</i>	Dragged data has entered the control
<i>DragLeave</i>	Dragged data is leaving a control
<i>DragOver</i>	Data is being dragged over the control
<i>Drop</i>	Data has been dropped on this control

Event	Description
<i>KillFocus</i>	Control has lost the focus
<i>SetFocus</i>	Control has received the input focus

Version

- 4.0.1 Original
- 4.1 ForeColor/BackColor property added
- 5.2.1 Added WebStyle property
- 6.0.0

Divider

Description

This control is used to create a vertical or horizontal divider that sends events specifying where it would like to be dragged to within the dialog box.

Define the control using the DialogBox AxControl command with a prog id of "wIntegrate.Divider.1".

To use it press and hold down the left mouse button over the divider and then move the mouse to the new position and then release it.

This is a Active X control.

Styles

The following table describes the styles for the Divider control:

Style	Description
<i>WS_VISIBLE</i>	The control is visible

Properties

The following table describes the properties for the Divider control:

Property	Description
<i>BorderStyle</i>	Set the style of the divider
<i>Height</i>	Get/Set the Height of the control.
<i>Horizontal</i>	Set orientation of the divider
<i>Left</i>	Get/Set left hand edge of the control
<i>MaximumPosition</i>	Maximum position in pixels that the control can be dragged.

Property	Description
<i>MinimumPosition</i>	Minimum position in pixels that the control can be dragged.
<i>Top</i>	Get/Set top edge of the control.
<i>Width</i>	Get/Set the width of the control.

Events

The following table describes the events for the Divider control:

Event	Description
<i>Drag</i>	A drag is in progress.
<i>DragStarted</i>	A drag has started
<i>DragEnded</i>	A drag has finished

Draw

Description

The Draw control displays a drawing created using the Draw script commands.

To create a draw control use the DialogBox Control command with the class name set to "winteg.draw".

To send the Draw control output to the control use the Draw To command.

This is a Application control.

Styles

The following table describes the styles for the Draw control:

Style	Description
<i>WS_VISIBLE</i>	The control is visible

Events

The following table describes the events for the Draw control:

Event	Description
<i>Click</i>	Left mouse clicked on the control

EditText

Description

The edit text control is used to get text input from the user

This is a Standard control.

Styles

The following table describes the styles for the EditText control:

Style	Description
<i>ES_AUTOHSCROLL</i>	Automatically scrolls horizontally when the cursor reaches the right edge of the edit box.
<i>ES_AUTOVSCROLL</i>	Scrolls vertically automatically when entering text.
<i>ES_CENTER</i>	Centers lines, used with <i>ES_MULTILINE</i> .
<i>ES_LEFT</i>	A single, left-justified line
<i>ES_LOWERCASE</i>	Converts data to lower case.
<i>ES_MULTILINE</i>	Multiple, left-justified lines
<i>ES_NOHIDESEL</i>	Keeps any selected characters highlighted when the focus leaves this control.
<i>ES_OEMCONVERT</i>	Converts OEM text.
<i>ES_PASSWORD</i>	Displays "*" instead of the letters entered.
<i>ES_READONLY</i>	Prevents editing the edit box.
<i>ES_RIGHT</i>	Right-justified lines, used with <i>ES_MULTILINE</i> .
<i>ES_UPPERCASE</i>	Converts data to upper case,

Style	Description
<i>ES_WANTRETURN</i>	Adds a carriage return within the control to start a new line in a multiline edit control.
<i>WS_BORDER</i>	Adds a border to the edit box.
<i>WS_DISABLED</i>	The control is initially disabled
<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_HSCROLL</i>	Adds a horizontal scroll bar.
<i>WS_TABSTOP</i>	Tab or back tab moves to the previous or next control with <i>WS_TABSTOP</i> style.
<i>WS_VSCROLL</i>	Adds a vertical scroll bar.

Properties

The following table describes the properties for the EditText control:

Property	Description
<i>BackColor</i>	Background color of the control
<i>DragData</i>	Set the data to be dragged.
<i>DragMode</i>	Set the mode for dragging data from this control
<i>DragTarget</i>	Set the valid targets for data dragged from this control
<i>Enabled</i>	Set or get the enabled state of the control
<i>Font</i>	Set or get the font used with the control.
<i>ForeColor</i>	Set or get the color used for the caption or text of the control
<i>Height</i>	Get/Set the Height of the control.
<i>Left</i>	Get/Set left hand edge of the control

Property	Description
<i>LimitText</i>	Maximum number of characters allowed
<i>Modified</i>	Get or Set the modified flag for the control.
<i>PasswordChar</i>	Set or get the password character
<i>ReadOnly</i>	Set or get the read only state of the control.
<i>TabStops</i>	Tab stops for the control
<i>Text</i>	The text from the control
<i>Top</i>	Get/Set top edge of the control.
<i>Visible</i>	Set or get the visible state for the control
<i>WebStyle</i>	Set/get the web style for this control
<i>Width</i>	Get/Set the width of the control.

Methods

The following table describes the methods for the EditText control:

Method	Description
<i>CharFromPos</i>	Get character index from position
<i>Clear</i>	Clears the current selection from the control
<i>Copy</i>	Copy the current selection to the clipboard
<i>Cut</i>	Cut the current selection to the clipboard
<i>FirstVisibleLine</i>	Return first visible line in the control
<i>Line</i>	Return a single line from the control
<i>LineCount</i>	Returns the number of lines in the edit control.
<i>LineFromChar</i>	Returns the line number for the specified position.
<i>LineIndex</i>	Return character index of first character on a line

Method	Description
<i>LineLength</i>	Return the length of the line
<i>LineScroll</i>	Scrolls the edit control by the specified columns and lines.
<i>Paste</i>	Pastes text from the clipboard to the control.
<i>ReplaceSel</i>	Replaces the current selection with the specified text
<i>Scroll</i>	Scrolls the edit control a number of lines
<i>ScrollCaret</i>	Scrolls the edit control so the caret is visible.
<i>Select</i>	Selects the specified section of the edit control.
<i>SelectInfo</i>	Return start and end position of the selection
<i>Undo</i>	Undoes the last edit of the control.

Events

The following table describes the events for the EditText control:

Event	Description
<i>Change</i>	The text has been changed
<i>DragEnd</i>	The dragging of data from this control has finished
<i>DragEnter</i>	Dragged data has entered the control
<i>DragLeave</i>	Dragged data is leaving a control
<i>DragOver</i>	Data is being dragged over the control
<i>DragStart</i>	Data is being dragged from this control
<i>Drop</i>	Data has been dropped on this control
<i>ErrSpace</i>	Edit control failed to add some text
<i>HScroll</i>	The edit control has been scrolled horizontally

Event	Description
<i>KillFocus</i>	Control has lost the focus
<i>MaxText</i>	The maximum amount of text exceeded
<i>SetFocus</i>	Control has received the input focus
<i>Update</i>	The text has been changed
<i>VScroll</i>	The edit control has been scrolled vertically

Version

4.1.0 Properties, Methods and Events added

4.2 Drag and Drop properties/events added

6.0 Added WebStyle property

Graphic

Description

The Graphic control displays an image in a dialog box.

This is a Application control.

Styles

The following table describes the styles for the Graphic control:

Style	Description
<i>WS_DISABLED</i>	The control is initially disabled
<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 <i>WS_TABSTOP</i> style.

Properties

The following table describes the properties for the Graphic control:

Property	Description
<i>Display</i>	Set or get how to display the image in the graphic control
<i>Enabled</i>	Set or get the enabled state of the control
<i>Height</i>	Get/Set the Height of the control.
<i>Image</i>	The filename of the image for the control
<i>ImageHeight</i>	Read the height in pixels of the image
<i>ImageWidth</i>	Read the width in pixels of the image

Property	Description
<i>Left</i>	Get/Set left hand edge of the control
<i>Top</i>	Get/Set top edge of the control.
<i>Visible</i>	Set or get the visible state for the control
<i>WebStyle</i>	Set/get the web style for this control
<i>Width</i>	Get/Set the width of the control.

Methods

The following table describes the methods for the Graphic control:

Method	Description
<i>ScrollTo</i>	Scrolls the image to the specified position.

Events

The following table describes the events for the Graphic control:

Event	Description
<i>Click</i>	Left mouse clicked on the control
<i>DragEnter</i>	Dragged data has entered the control
<i>DragLeave</i>	Dragged data is leaving a control
<i>DragOver</i>	Data is being dragged over the control
<i>Drop</i>	Data has been dropped on this control

Version

4.1 add new properties and method. Support for scroll styles

6.0 Added WebStyle property

GraphicButton

Description

The GraphicButton control is a push button control with a graphic image.

This is a Application control.

Styles

The following table describes the styles for the GraphicButton control:

Style	Description
<i>GBS_FLAT</i>	Displays the graphic as a flat button which is raised as you move the mouse over it.
<i>GBS_NOMAPCOLORS</i>	This style causes the display of the images for the button to be done without mapping colors to the current windows color scheme.
<i>GBS_RAISED</i>	Displays the graphic as a 3D, raised button.
<i>WS_DISABLED</i>	The control is initially disabled
<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.

Properties

The following table describes the properties for the GraphicButton control:

Property	Description
<i>Enabled</i>	Set or get the enabled state of the control
<i>Height</i>	Get/Set the Height of the control.
<i>Image</i>	The filename of the image for the control
<i>Left</i>	Get/Set left hand edge of the control
<i>Top</i>	Get/Set top edge of the control.
<i>Visible</i>	Set or get the visible state for the control
<i>WebStyle</i>	Set/get the web style for this control
<i>Width</i>	Get/Set the width of the control.

Events

The following table describes the events for the GraphicButton control:

Event	Description
<i>Click</i>	Left mouse clicked on the control
<i>DragEnter</i>	Dragged data has entered the control
<i>DragLeave</i>	Dragged data is leaving a control
<i>DragOver</i>	Data is being dragged over the control
<i>Drop</i>	Data has been dropped on this control

Version

4.0.1 New style added

4.1 New property added

6.0 Added WebStyle property

Grid

Description

The DialogBox 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.

This is a Active X control.

Styles

The following table describes the styles for the Grid control:

Style	Description
<i>WS_DISABLED</i>	The control is initially disabled
<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.

Properties

The following table describes the properties for the Grid control:

Property	Description
<i>CellValue</i>	Value of the current cell.
<i>ColHeaders</i>	Display column headers. Default is true.
<i>ColText</i>	Text for column headings
<i>ColWidths</i>	Width of columns.
<i>Column</i>	Current cell column.

Property	Description
<i>Columns</i>	Number of columns in grid.
<i>Data</i>	Dynamic array of data in grid
<i>Enabled</i>	Set or get the enabled state of the control
<i>EnterAction</i>	Set action when RETURN is pressed.
<i>FloatingCells</i>	Allow floating cells.
<i>Height</i>	Get/Set the Height of the control.
<i>Left</i>	Get/Set left hand edge of the control
<i>ReadOnly</i>	Set or get the read only state of the control.
<i>Redraw</i>	Redraw mode on/off.
<i>Row</i>	Current cell row.
<i>RowHeaders</i>	Display row header (default is true).
<i>RowHeights</i>	Height of rows (comma-separated) in dialog units.
<i>RowText</i>	Text for row headings (comma or carriage-return separated).
<i>Rows</i>	Number of rows in grid.
<i>SelectMode</i>	Cell selection mode
<i>SortColOnDbClick</i>	Automatically sort by column if user double clicks on header
<i>SortRowOnDbClick</i>	Automatically sort by row if user double clicks on header
<i>Top</i>	Get/Set top edge of the control.
<i>TrackHeight</i>	Enable user to modify row heights (default is true).
<i>TrackWidth</i>	Enable user to modify column widths (default true)
<i>Visible</i>	Set or get the visible state for the control
<i>Width</i>	Get/Set the width of the control.

Methods

The following table describes the methods for the Grid control:

Method	Description
<i>ClearRange</i>	The ClearRange method clears cells in a grid
<i>Copy</i>	Copy the current selection to the clipboard
<i>Cut</i>	Cut the current selection to the clipboard
<i>DeleteCols</i>	The DeleteCols method deletes columns from a grid
<i>DeleteRows</i>	The DeleteRows method deletes rows from a grid
<i>FillRange</i>	The FillRange method inserts a value into a specified range of columns and rows
<i>Find</i>	Find text anywhere in the grid
<i>GetCellAttribute</i>	The GetCellAttribute method returns the current value of an attribute for a cell
<i>GetCellAutoSize</i>	The GetCellAutoSize method returns whether the cell height will grow to fit the text
<i>GetCellBackColor</i>	The GetCellBackColor method returns the background color of a cell
<i>GetCellBorder</i>	The GetCellBorder method returns the color, style, and width of a cell border
<i>GetCellControl</i>	The GetCellControl method returns the cell control type (for example: push button, radio button,
<i>GetCellEnabled</i>	The GetCellEnabled method returns whether or not a cell is enabled
<i>GetCellFloat</i>	The GetCellFloat method returns whether or not a cell is allowed to overlay cells to the right of it

Method	Description
<i>GetCellFont</i>	The GetCellFont method returns the font type, size and style of a cell
<i>GetCellForeColor</i>	The GetCellForeColor method returns the foreground color of a cell
<i>GetCellJustify</i>	The GetCellJustify method returns the cell justification
<i>GetCellList</i>	list of options for a combo box control
<i>GetCellReadOnly</i>	The GetCellReadOnly returns whether or not a cell is read-only
<i>GetCellValue</i>	The GetCellValue method returns the text from a specified cell
<i>GetColWidth</i>	The GetColWidth method returns the width of a specified column
<i>GetCurrentCell</i>	The GetCurrentCell method returns the row and column numbers of the current cell
<i>GetCurrentText</i>	The GetCurrentText method returns the text from the current cell before it has been entered into
<i>GetRangeAttribute</i>	Returns attributes used in a range
<i>GetRangeAutoSize</i>	Returns cell auto size flag for a range
<i>GetRangeBackColor</i>	Returns the background colors for a range of cells
<i>GetRangeBorder</i>	returns border information for a range
<i>GetRangeControl</i>	Returns the cell control type for a range of cells.
<i>GetRangeEnabled</i>	Returns the enabled flags for a range of cells
<i>GetRangeFloat</i>	Returns the floating flags for a range of cells
<i>GetRangeForeColor</i>	Returns the foreground colors for a range of cells

Method	Description
<i>GetRangeJustify</i>	Returns the justifications for a range of cells
<i>GetRangeList</i>	Returns the lists from a range of cells
<i>GetRangeReadOnly</i>	Returns the read-only flags for a range of cells
<i>GetRangeValue</i>	Returns the values from a range of cells
<i>GetRowHeight</i>	The <code>GetRowHeight</code> method returns the height of a row
<i>HideCols</i>	The <code>HideCols</code> method hides or displays the columns you specify
<i>HideRows</i>	The <code>HideRows</code> method hides or shows the rows you specify
<i>InsertCols</i>	The <code>InsertCols</code> method inserts into a grid the number of columns you specify
<i>InsertRows</i>	The <code>InsertRows</code> method inserts the number of rows you specify into a grid
<i>Paste</i>	Pastes text from the clipboard to the control.
<i>ResizeColToFit</i>	The <code>ResizeColToFit</code> method resizes to fit the data the columns you specify
<i>ResizeRowToFit</i>	The <code>ResizeRowToFit</code> method resizes the rows you specify to fit the data
<i>ScrollToCell</i>	Scroll grid to display a cell you specify
<i>SelectAll</i>	Select/Unselect all the cells in the grid
<i>SelectCell</i>	Select/Unselect the specified cell
<i>SelectInfo</i>	The <code>SelectInfo</code> method returns a dynamic array containing a list of currently selected cells
<i>SelectRange</i>	Select/Unselect the specified range of cells

Method	Description
<i>SetCellAttribute</i>	The SetCellAttribute method sets the attributes of a cell
<i>SetCellAutoSize</i>	The SetCellAutoSize method specifies whether a specified cell height will grow to fit the text
<i>SetCellBackColor</i>	The SetCellBackColor method sets the background color for a cell
<i>SetCellBorder</i>	The SetCellBorder method sets the color, style, and width of a cell border
<i>SetCellControl</i>	The SetCellControl method sets the control type (for example: push button, radio button, check
<i>SetCellEnabled</i>	The SetCellEnabled method sets whether or not a cell is enabled
<i>SetCellFloat</i>	The SetCellFloat method specifies whether a cell is allowed to overlay cells to the right of it
<i>SetCellFont</i>	The SetCellFont method specifies the font name, size, and style for a cell
<i>SetCellForeColor</i>	The SetCellForeColor method sets the foreground color of a cell
<i>SetCellJustify</i>	The SetCellJustify method sets the text justification for a cell
<i>SetCellList</i>	list or caption for a cell control
<i>SetCellReadOnly</i>	The SetCellReadOnly method sets a cell to be read-only
<i>SetCellValue</i>	The SetCellValue method sets the text for a specified cell
<i>SetColWidth</i>	The SetColWidth method sets the width for a specified column

Method	Description
<i>SetCurrentCell</i>	The SetCurrentCell method sets the current cell in the grid and returns true if the current cell
<i>SetCurrentText</i>	The SetCurrentText method sets the text for the current cell within the ValidateCell method
<i>SetRangeAttribute</i>	The SetRangeAttribute method sets an attribute for cells within the specified range
<i>SetRangeAutoSize</i>	The SetRangeAutoSize method allows a specified range of cells to grow in height to fit the text
<i>SetRangeBackColor</i>	The SetRangeBackColor method sets the background color for cells within the specified range
<i>SetRangeBorder</i>	Sets the border appearance for a range of cells
<i>SetRangeControl</i>	Sets the control type for a range of cells.
<i>SetRangeEnabled</i>	The SetRangeEnabled method sets whether or not cells within the specified range is enabled
<i>SetRangeFloat</i>	Set floating status for a range of cells
<i>SetRangeFont</i>	The SetRangeFont method specifies the font name, size, and style for cells within the specified
<i>SetRangeForeColor</i>	The SetRangeForeColor method sets the foreground color for cells within the specified range
<i>SetRangeJustify</i>	The SetRangeJustify method sets the text justification for cells within the specified range
<i>SetRangeList</i>	Sets the list/button caption for a range of cells

Method	Description
<i>SetRangeReadOnly</i>	Sets cells to be read-only for a range of cells
<i>SetRangeValue</i>	The SetRangeValue method sets the text for cells within the specified range
<i>SetRowHeight</i>	The SetRowHeight method sets the height of a row
<i>SortCols</i>	The SortCols method sorts the grid by the specified value in the row
<i>SortRows</i>	The SortRows method sorts the grid by the specified columns

Events

The following table describes the events for the Grid control:

Event	Description
<i>Activate</i>	Called when the grid is activated
<i>ButtonClick</i>	Called when a user selects any of the button controls
<i>CellClick</i>	Called when the left mouse button is clicked on a grid.
<i>CellRightClick</i>	Called when the right mouse button is clicked on a grid.
<i>CurrentCell</i>	Called when the cursor moves to a new cell in the grid
<i>Deactivate</i>	Called when the grid is deactivated.
<i>EndEditing</i>	Called when a user has finished editing a cell
<i>ModifyCell</i>	Called when a user has modified a cell
<i>StartEditing</i>	Called when a user starts editing a cell.
<i>ValidateCell</i>	Called when a cells contents needs validation.

Version

4.0.2 New methods added: SelectCell, SelectRange, SelectAll, Cut, Copy, Paste, Find

4.1 New events EndEditing and ModifyCell

GroupBox

Description

The GroupBox control is a rectangle control with a caption used to contain a group of controls that logically belong together.

When used to group radio buttons the name of the selected radio button can be obtained/set through the GroupBox.

This is a Standard control.

Styles

The following table describes the styles for the GroupBox control:

Style	Description
<i>BS_GROUPBOX</i>	Creates a rectangular box with a single-line border and is the default group box style.
<i>WS_DISABLED</i>	The control is initially disabled
<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.

Properties

The following table describes the properties for the GroupBox control:

Property	Description
<i>Caption</i>	Set or get the current value of the caption of the control.

Property	Description
<i>Enabled</i>	Set or get the enabled state of the control
<i>Font</i>	Set or get the font used with the control.
<i>ForeColor</i>	Set or get the color used for the caption or text of the control
<i>Height</i>	Get/Set the Height of the control.
<i>Left</i>	Get/Set left hand edge of the control
<i>Radio</i>	The name of the radio control in the group that is selected.
<i>Top</i>	Get/Set top edge of the control.
<i>Visible</i>	Set or get the visible state for the control
<i>WebStyle</i>	Set/get the web style for this control
<i>Width</i>	Get/Set the width of the control.

Version

4.1 New properties added

6.0 Added WebStyle property

Header

Description

The Header control is used as a heading for another control in a dialog box.

This is a Common control.

Styles

The following table describes the styles for the Header control:

Style	Description
<i>HDS_BUTTONS</i>	Header items behave like buttons.
<i>WS_DISABLED</i>	The control is initially disabled
<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 <i>WS_TABSTOP</i> style.
<i>WS_VISIBLE</i>	The control is visible

Properties

The following table describes the properties for the Header control:

Property	Description
<i>BackColor</i>	Background color of the control
<i>Enabled</i>	Set or get the enabled state of the control
<i>Font</i>	Set or get the font used with the control.
<i>ForeColor</i>	Set or get the color used for the caption or text of the control

Property	Description
<i>Height</i>	Get/Set the Height of the control.
<i>LabelWidths</i>	A comma separated list of widths
<i>Labels</i>	List of text for headings.
<i>Left</i>	Get/Set left hand edge of the control
<i>Top</i>	Get/Set top edge of the control.
<i>Visible</i>	Set or get the visible state for the control
<i>Width</i>	Get/Set the width of the control.

Methods

The following table describes the methods for the Header control:

Method	Description
<i>Count</i>	The Count method returns the number of items in a header
<i>DeleteItem</i>	The DeleteItem method deletes an item from a header and returns a nonzero value if successful
<i>GetItem</i>	The GetItem method returns the details of a header item and returns a nonzero value if successful
<i>GetItemText</i>	The GetItem Text method retrieves the text from a header item
<i>GetItemWidth</i>	The GetItemWidth method retrieves the width of a header item or returns zero if an error occurs
<i>InsertItem</i>	The InsertItem method inserts a new item into the header and returns the index value of the insertion
<i>InsertItemText</i>	The InsertItemText method inserts a new header item with the specified text and width and

Method	Description
<i>SetItem</i>	The SetItem method sets the properties of a header item from the details specified in the Header
<i>SetItemText</i>	The SetItem method sets the text for a header item and returns a nonzero value if successful
<i>SetItemWidth</i>	Set the width of a header item

Events

The following table describes the events for the Header control:

Event	Description
<i>BeginTrack</i>	Start of column resize
<i>DividerDbClick</i>	Double click on the divider between headings
<i>DragEnter</i>	Dragged data has entered the control
<i>DragLeave</i>	Dragged data is leaving a control
<i>DragOver</i>	Data is being dragged over the control
<i>Drop</i>	Data has been dropped on this control
<i>EndTrack</i>	Tracking has finished
<i>ItemClick</i>	Header item has been clicked.
<i>ItemDbClick</i>	Double click on a header item
<i>Track</i>	Header size is being tracked

Version

4.1

Icon

Description

The Icon control creates a control which holds an icon from the wIntegrate executable (winteg.exe).

The only icon available is the "winteg" icon.

Use the Graphic Control to show other icons/images on the dialog box.

This is a Standard control.

Styles

The following table describes the styles for the Icon control:

Style	Description
<i>WS_VISIBLE</i>	The control is visible

Properties

The following table describes the properties for the Icon control:

Property	Description
<i>Height</i>	Get/Set the Height of the control.
<i>Left</i>	Get/Set left hand edge of the control
<i>Top</i>	Get/Set top edge of the control.
<i>Visible</i>	Set or get the visible state for the control
<i>Width</i>	Get/Set the width of the control.

ListBox

Description

The ListBox control displays a list of items with various options.

You can highlight single or multiple options with the mouse or keyboard.

This is a Standard control.

Styles

The following table describes the styles for the ListBox control:

Style	Description
<i>LBS_DISABLENOSCROLL</i>	Shows the vertical scroll bar even if it is not necessary.
<i>LBS_MULTICOLUMN</i>	Designates multiple columns.
<i>LBS_MULTIPLESEL</i>	Allows multiple items to be selected in the list box.
<i>LBS_NOINTEGRALHEIGHT</i>	If this is not specified, the last line of the list may not completely display.
<i>LBS_NOSEL</i>	Creates a list box where no items can be selected
<i>LBS_NOTIFY</i>	Notifies wIntegrate of clicks.
<i>LBS_SORT</i>	Sorts the contents of the list box alphabetically.
<i>LBS_STANDARD</i>	This is a combination of these styles: WS_BORDER, LBS_SORT, LBS_NOTIFY, WS_SCROLL.
<i>LBS_USETABSTOPS</i>	Uses tab stops in conjunction with the TabStops property
<i>WS_BORDER</i>	Adds a border to the edit box.

Style	Description
<i>WS_DISABLED</i>	The control is initially disabled
<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_HSCROLL</i>	Adds a horizontal scroll bar.
<i>WS_TABSTOP</i>	Tab or back tab moves to the previous or next control with <i>WS_TABSTOP</i> style.
<i>WS_VSCROLL</i>	Adds a vertical scroll bar.

Properties

The following table describes the properties for the ListBox control:

Property	Description
<i>AnchorIndex</i>	Index of first line when selecting multiple lines
<i>BackColor</i>	Background color of the control
<i>CaretIndex</i>	Index of line with focus
<i>ColumnWidth</i>	Width of the columns in the list box
<i>DragData</i>	Set the data to be dragged.
<i>DragMode</i>	Set the mode for dragging data from this control
<i>DragTarget</i>	Set the valid targets for data dragged from this control
<i>Enabled</i>	Set or get the enabled state of the control
<i>Font</i>	Set or get the font used with the control.
<i>ForeColor</i>	Set or get the color used for the caption or text of the control
<i>Height</i>	Get/Set the Height of the control.
<i>HorizontalExtent</i>	Set or get the width in pixels of the list
<i>Left</i>	Get/Set left hand edge of the control

Property	Description
<i>List</i>	all the items in a listbox list or the list part of the combobox
<i>SelIndex</i>	The index of the current selection
<i>SelText</i>	The selected text
<i>TabStops</i>	Position of the tab stops in the listbox
<i>Top</i>	Get/Set top edge of the control.
<i>TopIndex</i>	Set or get the line number of the top line visible in the list part of
<i>UpdateMode</i>	Specifies the information to store when the dialog box is closed
<i>Visible</i>	Set or get the visible state for the control
<i>WebStyle</i>	Set/get the web style for this control
<i>Width</i>	Get/Set the width of the control.

Methods

The following table describes the methods for the ListBox control:

Method	Description
<i>AddLine</i>	Add a new line to the list
<i>ClearSel</i>	Clear the selection state
<i>Count</i>	Returns the number of lines in the list
<i>DeleteLine</i>	Delete a line from the list
<i>Empty</i>	Clears all the lines from the list
<i>Find</i>	Find line starting with specified text
<i>FindExact</i>	Searches for the line in the list with the specified text.
<i>GetItemData</i>	Returns the number associated with the line
<i>GetSel</i>	Get the selection state for a line

Method	Description
<i>InitStorage</i>	Increases the amount of storage used for the list
<i>InsPosFromPoint</i>	Return the line number where a new item would be inserted
<i>InsertLine</i>	Insert a line into the list
<i>ItemFromPoint</i>	Return the index of the line nearest to the specified point
<i>Line</i>	Returns the text of the specified line
<i>SelCount</i>	Return the number of lines selected
<i>SelLine</i>	Selects the line of text which start with the specified text
<i>SelRange</i>	Select a range of lines
<i>SetItemData</i>	Set an number against the specified line of a list box.
<i>SetSel</i>	Set the selection state for a line

Events

The following table describes the events for the ListBox control:

Event	Description
<i>DblClick</i>	Left mouse double clicked on the control
<i>DragEnd</i>	The dragging of data from this control has finished
<i>DragEnter</i>	Dragged data has entered the control
<i>DragLeave</i>	Dragged data is leaving a control
<i>DragOver</i>	Data is being dragged over the control
<i>DragStart</i>	Data is being dragged from this control
<i>Drop</i>	Data has been dropped on this control
<i>KillFocus</i>	Control has lost the focus

Event	Description
<i>SelCancel</i>	The selection from the list box has been cancelled
<i>SelChange</i>	The selection is about to change
<i>SetFocus</i>	Control has received the input focus

Version

- 4.1 Added new properties, methods and events
 - 4.1.1 InsPosFromPoint method added
- 4.2 Drag and drop properties/events
- 5.0 "E" tab stops option
- 5.1 LBS_NOSEL style
- 6.0 Added WebStyle property

ListView

Description

The ListView control displays a list of items with optional images and additional information

This is a Common control.

Styles

The following table describes the styles for the ListView control:

Style	Description
<i>LVS_ALIGNLEFT</i>	Specifies that items are left-aligned in icon and small icon view
<i>LVS_ALIGNTOP</i>	Specifies that icons are automatically kept arranged in icon and small icon view.
<i>LVS_BUTTON</i>	Specifies that item icons look like buttons in icon view.
<i>LVS_EDITLABELS</i>	Allows item text to be edited in place. The parent window must process the ENDLABELEDIT event.
<i>LVS_ICON</i>	Specifies icon view.
<i>LVS_LIST</i>	Specifies list view.
<i>LVS_NOCOLUMNHEADER</i>	Specifies that a column header is not displayed in report view. By default, columns have headers in report view.
<i>LVS_NOLABELWRAP</i>	Displays item text on a single line in icon view. By default, item text may wrap in icon view.
<i>LVS_NOScroll</i>	Disables scrolling. All items must be within the client area.

Style	Description
<i>LVS_NOSORTHEADER</i>	Specifies that column headers do not work like buttons. This style is useful if clicking a column header in report view does not carry out an action, such as sorting.
<i>LVS_REPORT</i>	Specifies report view. When using the LVS_REPORT style with a List View control, the first column is always left-aligned. You can not use LVC_FMT_RIGHT to change this alignment.
<i>LVS_SHOWSELALWAYS</i>	Always show the selection, if any, even if the control does not have the focus.
<i>LVS_SINGLESEL</i>	Allows only one item at a time to be selected. By default, multiple items may be selected.
<i>LVS_SMALLICON</i>	Specifies small icon view.
<i>LVS_SORTASCENDING</i>	Sorts items in ascending order based on item text.
<i>LVS_SORTDESCENDING</i>	Sorts items in descending order based on item text.
<i>WS_BORDER</i>	Adds a border to the edit box.
<i>WS_VISIBLE</i>	The control is visible

Properties

The following table describes the properties for the List View control:

Property	Description
<i>BackColor</i>	Background color of the control
<i>ColumnInfo</i>	Set information for columns.

Property	Description
<i>DragData</i>	Set the data to be dragged.
<i>DragMode</i>	Set the mode for dragging data from this control
<i>DragTarget</i>	Set the valid targets for data dragged from this control
<i>Enabled</i>	Set or get the enabled state of the control
<i>ForeColor</i>	Set or get the color used for the caption or text of the control
<i>Height</i>	Get/Set the Height of the control.
<i>Left</i>	Get/Set left hand edge of the control
<i>TextBackColor</i>	Set/Get background color for text.
<i>Top</i>	Get/Set top edge of the control.
<i>View</i>	Set view style.
<i>Visible</i>	Set or get the visible state for the control
<i>WebStyle</i>	Set/get the web style for this control
<i>Width</i>	Get/Set the width of the control.

Methods

The following table describes the methods for the ListView control:

Method	Description
<i>Arrange</i>	The Arrange method arranges items in listview and returns True if successful
<i>Count</i>	The Count method returns the number of items in the listview
<i>DeleteAllItems</i>	The DeleteAllItems method deletes all the Items in a listview and returns True if successful

Method	Description
<i>DeleteColumn</i>	The DeleteColumn method deletes a column from a listview and returns True if successful
<i>DeleteItem</i>	The DeleteItem method deletes an item from a listview and returns True if successful
<i>EditLabel</i>	The EditLabel method edits an item in a listview and returns a nonzero value if successful
<i>EnsureVisible</i>	The EnsureVisible method scrolls the listview to show the specified item and returns TRUE if the
<i>FindItem</i>	The FindItem method finds an item with the specified text or data
<i>GetColumn</i>	The GetColumn method gets information about a column and returns True if successful
<i>GetColumnWidth</i>	The GetColumnWidth method returns the width of a column
<i>GetCountPerPage</i>	The GetCountPerPage method returns the number of fully visible items when the control is in list
<i>GetItem</i>	Get details on a list view item
<i>GetItemPosition</i>	The GetItemPosition method gets the top left position of an item and returns True if successful
<i>GetItemState</i>	The GetItemState method returns the state of an item
<i>GetItemText</i>	The GetItemText method returns the text for item

Method	Description
<i>GetNextItem</i>	The GetNextItem method returns the next item from the listview and returns the index of the item
<i>GetOrigin</i>	The GetOrigin method gets the co-ordinates of the view origin and returns True if successful or
<i>GetSelectedCount</i>	The GetSelectedCount method returns the number of selected items
<i>GetTopIndex</i>	The GetTopIndex method returns the top index of an item when in list or report mode
<i>InsertColumn</i>	The InsertColumn method insert a new column into the listview and returns the index of the new
<i>InsertItem</i>	The InsertItem method sets the item number field of lvitem to specify the location to insert
<i>SetColumn</i>	The SetColumn method sets information about a column and returns True if successful
<i>SetColumnWidth</i>	The SetColumnWidth method sets the width of a specified column
<i>SetImageList</i>	Set image list to use with a list view
<i>SetItem</i>	The SetItem method sets the details of an item or subitem and returns True if successful
<i>SetItemCount</i>	The SetItemCount method to specify the eventual number of items it will have when adding a
<i>SetItemPosition</i>	The SetItemPosition method moves a specified item to the required position

Method	Description
<i>SetItemState</i>	The SetItemState method sets the state of an item and returns True if the mask modified
<i>SetItemText</i>	The SetItemText method sets the text for an item and returns True if successful
<i>Update</i>	The Update method updates a specified item

Events

The following table describes the events for the ListView control:

Event	Description
<i>BeginLabelEdit</i>	Item is about to be edited.
<i>ItemChanging</i>	Item is being modified
<i>ItemChanged</i>	Item has been modified
<i>Click</i>	Left mouse click.
<i>ColumnClick</i>	Click on a column header.
<i>DbClick</i>	Left mouse double click.
<i>DragEnd</i>	The dragging of data from this control has finished
<i>DragEnter</i>	Dragged data has entered the control
<i>DragLeave</i>	Dragged data is leaving a control
<i>DragOver</i>	Data is being dragged over the control
<i>DragStart</i>	Data is being dragged from this control
<i>Drop</i>	Data has been dropped on this control
<i>EndLabelEdit</i>	Item has been edited
<i>KeyDown</i>	key was pressed
<i>KillFocus</i>	Control has lost the focus
<i>Return</i>	Return key was pressed

Event	Description
<i>RightClick</i>	Right mouse click.
<i>RightDbClick</i>	Right mouse double click.
<i>SetFocus</i>	Control has received the input focus

Version

- 4.1 Standard properties added
- 4.2 Drag and drop properties and events
- 5.0.1 ItemChanged, ItemChanging events added
- 6.0 Added WebStyle property

ProgressBar

Description

The ProgressBar control displays a bar that can be increased in size to indicate the progress of some operation.

This is a Common control.

Styles

The following table describes the styles for the ProgressBar control:

Style	Description
<i>WS_VISIBLE</i>	The control is visible

Properties

The following table describes the properties for the ProgressBar control:

Property	Description
<i>Enabled</i>	Set or get the enabled state of the control
<i>Height</i>	Get/Set the Height of the control.
<i>Left</i>	Get/Set left hand edge of the control
<i>Position</i>	Sets the position.
<i>Range</i>	Sets the minimum and maximum positions of the progress bar.
<i>Step</i>	Sets the step increment.
<i>Top</i>	Get/Set top edge of the control.
<i>Visible</i>	Set or get the visible state for the control
<i>WebStyle</i>	Set/get the web style for this control
<i>Width</i>	Get/Set the width of the control.

Methods

The following table describes the methods for the ProgressBar control:

Method	Description
<i>DeltaPos</i>	The DeltaPos method moves the progress bar by a given st
<i>StepIt</i>	moves the progress bar by a step increment

Events

The following table describes the events for the ProgressBar control:

Event	Description
<i>DragEnter</i>	Dragged data has entered the control
<i>DragLeave</i>	Dragged data is leaving a control
<i>DragOver</i>	Data is being dragged over the control
<i>Drop</i>	Data has been dropped on this control

Version

6.0 Added WebStyle property

PushButton

Description

A PushButton control is a raised button that can be pressed. Use push buttons to initiate some kind of action in the dialog box.

This is a Standard control.

Styles

The following table describes the styles for the PushButton control:

Style	Description
<i>BS_BITMAP</i>	Button displays a bitmap instead of text. Set the Bitmap with the Image property.
<i>BS_BOTTOM</i>	Align the caption to the bottom of the button
<i>BS_CENTER</i>	Horizontally center the caption in the button
<i>BS_DEFPUSHBUTTON</i>	Default push button - only one push button on a dialog can have this style
<i>BS_FLAT</i>	Display the button without 3D effects.
<i>BS_LEFT</i>	Align the caption to the left of the button
<i>BS_MULTILINE</i>	Word wrap the caption if it is longer than the buttons size.
<i>BS_NOTIFY</i>	Enable extra events. Normally a button will only react to the click event. Enabling this check box enable the DblClick, SetFocus and KillFocus events as well.
<i>BS_PUSHBUTTON</i>	Designates a push button control.

Style	Description
<i>BS_RIGHT</i>	Align the caption to the right of the button
<i>BS_TOP</i>	Align the caption to the top of the button
<i>BS_VCENTER</i>	Vertically center the caption in the button
<i>WS_DISABLED</i>	The control is initially disabled
<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.

Properties

The following table describes the properties for the PushButton control:

Property	Description
<i>Caption</i>	Set or get the current value of the caption of the control.
<i>Default</i>	Set/get this button as the default push button
<i>Enabled</i>	Set or get the enabled state of the control
<i>Font</i>	Set or get the font used with the control.
<i>Height</i>	Get/Set the Height of the control.
<i>Image</i>	bitmap file name for the control
<i>Left</i>	Get/Set left hand edge of the control
<i>Top</i>	Get/Set top edge of the control.
<i>Visible</i>	Set or get the visible state for the control
<i>WebStyle</i>	Set/get the web style for this control

Property	Description
<i>Width</i>	Get/Set the width of the control.

Events

The following table describes the events for the PushButton control:

Event	Description
<i>Click</i>	Left mouse clicked on the control
<i>DblClick</i>	Left mouse double clicked on the control
<i>DragEnter</i>	Dragged data has entered the control
<i>DragLeave</i>	Dragged data is leaving a control
<i>DragOver</i>	Data is being dragged over the control
<i>Drop</i>	Data has been dropped on this control
<i>KillFocus</i>	Control has lost the focus
<i>SetFocus</i>	Control has received the input focus

Version

4.1 Modified to add new styles, events and properties

6.0 Added WebStyle property

RadioButton

Description

The RadioButton control adds a selectable control with text to a dialog box.

You can group several radio buttons together in a box by using a GroupBox control. The GroupBox control must be used before the RadioButtons to create a box with radio buttons.

This is a Standard control.

Styles

The following table describes the styles for the RadioButton control:

Style	Description
<i>BS_BITMAP</i>	Button displays a bitmap instead of text. Set the Bitmap with the Image property.
<i>BS_BOTTOM</i>	Align the caption to the bottom of the button
<i>BS_CENTER</i>	Horizontally center the caption in the button
<i>BS_FLAT</i>	Display the button without 3D effects.
<i>BS_LEFT</i>	Align the caption to the left of the button
<i>BS_LEFTEXT</i>	The caption is place on the left of the check box
<i>BS_MULTILINE</i>	Word wrap the caption if it is longer then the buttons size.

Style	Description
<i>BS_NOTIFY</i>	Enable extra events. Normally a button will only react to the click event. Enabling this check box enable the DblClick, SetFocus and KillFocus events as well.
<i>BS_PUSHLIKE</i>	Give the button the same appearance as a push button. But unlike a push button is stays down when selected.
<i>BS_RADIOBUTTON</i>	Creates a radio button with a line of text to the right, and is the default radio button style.
<i>BS_RIGHT</i>	Align the caption to the right of the button
<i>BS_TOP</i>	Align the caption to the top of the button
<i>BS_VCENTER</i>	Vertically center the caption in the button
<i>WS_DISABLED</i>	The control is initially disabled
<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.

Properties

The following table describes the properties for the RadioButton control:

Property	Description
<i>BackColor</i>	Background color of the control
<i>BackStyle</i>	Set or get the background style

Property	Description
<i>Caption</i>	Set or get the current value of the caption of the control.
<i>Check</i>	Set to get the selected state of the radio button.
<i>Enabled</i>	Set or get the enabled state of the control
<i>Font</i>	Set or get the font used with the control.
<i>ForeColor</i>	Set or get the color used for the caption or text of the control
<i>Height</i>	Get/Set the Height of the control.
<i>Image</i>	bitmap file name for the control
<i>Left</i>	Get/Set left hand edge of the control
<i>Top</i>	Get/Set top edge of the control.
<i>Visible</i>	Set or get the visible state for the control
<i>WebStyle</i>	Set/get the web style for this control
<i>Width</i>	Get/Set the width of the control.

Events

The following table describes the events for the RadioButton control:

Event	Description
<i>Click</i>	Left mouse clicked on the control
<i>DbClick</i>	Left mouse double clicked on the control
<i>DragEnter</i>	Dragged data has entered the control
<i>DragLeave</i>	Dragged data is leaving a control
<i>DragOver</i>	Data is being dragged over the control
<i>Drop</i>	Data has been dropped on this control
<i>KillFocus</i>	Control has lost the focus
<i>SetFocus</i>	Control has received the input focus

Version

4.1 Added new styles, events and properties

6.0 Added WebStyle property

Rectangle

Description

The Rectangle control display etched or sunken rectangles or lines.

To create a rectangle control use the DialogBox Control command with the class name of "static" and specify the appropriate style from the styles table below.

This is a Application control.

Styles

The following table describes the styles for the Rectangle control:

Style	Description
<i>SS_BLACKFRAME</i>	Sunken rectangle
<i>SS_ETCHEDFRAME</i>	Etched border of the rectangle
<i>SS_ETCHEDHORZ</i>	Etched horizontal line
<i>SS_ETCHEDVERT</i>	Etched Vertical line
<i>SS_GRAYFRAME</i>	Border is indented into the dialog box
<i>SS_WHITEFRAME</i>	Raised rectangle
<i>WS_VISIBLE</i>	The control is visible

Properties

The following table describes the properties for the Rectangle control:

Property	Description
<i>WebStyle</i>	Set/get the web style for this control

Version

6.0 Added WebStyle property

ScrollBar

Description

The ScrollBar control displays a scroll bar in a dialog box.

This is a Standard control.

Styles

The following table describes the styles for the ScrollBar control:

Style	Description
<i>SBS_HORZ</i>	Designates a horizontal scroll bar.
<i>SBS_VERT</i>	Designates a vertical scroll bar.
<i>WS_DISABLED</i>	The control is initially disabled
<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.

Properties

The following table describes the properties for the ScrollBar control:

Property	Description
<i>BackColor</i>	Background color of the control
<i>Enabled</i>	Set or get the enabled state of the control
<i>Height</i>	Get/Set the Height of the control.
<i>Left</i>	Get/Set left hand edge of the control

Property	Description
<i>Position</i>	The position of the scroll bar. This is the default property
<i>Range</i>	The minimum and maximum values for the scroll bar
<i>Top</i>	Get/Set top edge of the control.
<i>Visible</i>	Set or get the visible state for the control
<i>WebStyle</i>	Set/get the web style for this control
<i>Width</i>	Get/Set the width of the control.

Events

The following table describes the events for the ScrollBar control:

Event	Description
<i>DragEnter</i>	Dragged data has entered the control
<i>DragLeave</i>	Dragged data is leaving a control
<i>DragOver</i>	Data is being dragged over the control
<i>Drop</i>	Data has been dropped on this control

Version

4.1 Added WebStyle property

6.0

TabControl

Description

The TabControl control displays a series of tabs in a dialog box.

This is a Common control.

Styles

The following table describes the styles for the TabControl control:

Style	Description
<i>TCS_BUTTONS</i>	Specifies that tabs appear as buttons, and no border is drawn around the display area.
<i>TCS_FIXEDWIDTH</i>	Specifies that all tabs are the same width. This style cannot be combined with the <i>TCS_RIGHTJUSTIFY</i> style.
<i>TCS_FOCUSNEVER</i>	Specifies that the tab control never receives the input focus.
<i>TCS_FOCUSONBUTTONDOWN</i>	Specifies that tabs receive the input focus when clicked.
<i>TCS_FORCEICONLEFT</i>	Aligns icons with the left edge of each fixed-width tab. This style can only be used with the <i>TCS_FIXEDWIDTH</i> style.
<i>TCS_FORCELABELLEFT</i>	Aligns labels with the left edge of each fixed-width tab; that is, it displays the label immediately to the right of the icon instead of centering it. This style can only be used with the <i>TCS_FIXEDWIDTH</i> style, and it implies the <i>TCS_FORCEICONLEFT</i> style.

Style	Description
<i>TCS_MULTILINE</i>	Displays multiple rows of tabs, if necessary, so all tabs are visible at once.
<i>TCS_RAGGEDRIGHT</i>	Does not stretch each row of tabs to fill the entire width of the control. This style is the default.
<i>TCS_RIGHTJUSTIFY</i>	Increases the width of each tab, if necessary, so that each row of tabs fills the entire width of the tab control. This window style is ignored unless the TCS_MULTILINE style is also specified.
<i>WS_VISIBLE</i>	The control is visible

Properties

The following table describes the properties for the TabControl control:

Property	Description
<i>Enabled</i>	Set or get the enabled state of the control
<i>FocusItem</i>	Index of tab with focus.
<i>Height</i>	Get/Set the Height of the control.
<i>Labels</i>	List of text for headings.
<i>Left</i>	Get/Set left hand edge of the control
<i>SelectedItem</i>	Index of currently selected tab.
<i>Top</i>	Get/Set top edge of the control.
<i>Visible</i>	Set or get the visible state for the control
<i>WebStyle</i>	Set/get the web style for this control
<i>Width</i>	Get/Set the width of the control.

Methods

The following table describes the methods for the TabControl control:

Method	Description
<i>AdjustRect</i>	The AdjustRect method calculates the size of a tab control to fit a specific area
<i>Count</i>	The Count method returns the number of labels in the tab control
<i>DeleteAllItems</i>	The DeleteAllItems method deletes all items and returns True if successful
<i>DeleteItem</i>	The DeleteItem method deletes the specified item and returns True if successful
<i>GetItemText</i>	returns the label text from a tab
<i>InsertItemText</i>	The InsertItemText method inserts a new tab with the specified label text and returns the index of
<i>RowCount</i>	The RowCount method returns the number of rows in the tab control
<i>SetItemSize</i>	The SetItemSize method sets the item size for fixed size tabs
<i>SetItemText</i>	The SetItemText method sets the label for a tab and returns True is successful
<i>SetPadding</i>	The SetPadding method sets the space around the text of a tab

Events

The following table describes the events for the TabControl control:

Event	Description
<i>DragEnter</i>	Dragged data has entered the control
<i>DragLeave</i>	Dragged data is leaving a control
<i>DragOver</i>	Data is being dragged over the control

Event	Description
<i>Drop</i>	Data has been dropped on this control
<i>KeyDown</i>	key was pressed
<i>SelChanged</i>	The selection has changed.
<i>SelChanging</i>	The selection is changing.

Version

4.1 Added WebStyle property

6.0

Text

Description

The text control displays static text on the dialog box.

This is a Standard control.

Styles

The following table describes the styles for the Text control:

Style	Description
<i>SS_CENTER</i>	Centers the text
<i>SS_ENDELLIPSIS</i>	Replace displayed text at end of control with ellipsis if its too long
<i>SS_LEFT</i>	Aligns the text with the left hand side of the control
<i>SS_PATHELLIPSIS</i>	Replace part of the displayed text with ellipsis by missing out parts of a filename path.
<i>SS_RIGHT</i>	Justifies the text to the right.
<i>SS_SUNKEN</i>	Draw a 3D sunken border around the control
<i>WS_DISABLED</i>	The control is initially disabled
<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.

Properties

The following table describes the properties for the Text control:

Property	Description
<i>BackColor</i>	Background color of the control
<i>BackStyle</i>	Set or get the background style
<i>Caption</i>	Set or get the current value of the caption of the control.
<i>Enabled</i>	Set or get the enabled state of the control
<i>Font</i>	Set or get the font used with the control.
<i>ForeColor</i>	Set or get the color used for the caption or text of the control
<i>Height</i>	Get/Set the Height of the control.
<i>Left</i>	Get/Set left hand edge of the control
<i>Top</i>	Get/Set top edge of the control.
<i>Visible</i>	Set or get the visible state for the control
<i>WebStyle</i>	Set/get the web style for this control
<i>Width</i>	Get/Set the width of the control.

Version

4.1 Modified to add new style and property

4.2 Added SS_PATHELLIPSIS, SS_ENDELLIPSIS styles

6.0 Added WebStyle property

TrackBar

Description

The Trackbar control display a bar that can be dragged to choose a number.

This is a Common control.

Styles

The following table describes the styles for the TrackBar control:

Style	Description
<i>TBS_AUTOTICKS</i>	Creates a slider that has a tick mark for each increment in its range of values. These tick marks are added automatically when an application calls the SetRange member function. You cannot use the SetTick method to specify the position of the tick marks if you use this style. Use the ClearTics member function instead.
<i>TBS_BOTH</i>	Displays tick marks on both sides of the slider in any orientation.
<i>TBS_BOTTOM</i>	Displays tick marks on the bottom of a horizontal slider. Can be used with the TBS_TOP style to display tick marks on both sides of the slider control.
<i>TBS_ENABLESELRANGE</i>	Displays a selection range. When a slider control has this style, the tick marks at the starting and ending positions of a selection range are displayed as triangles (instead of vertical dashes) and the selection range is highlighted.

Style	Description
<i>TBS_FIXEDLENGTH</i>	Allows the thumbs length to be changed
<i>TBS_HORZ</i>	Orients the slider horizontally. This is the default orientation.
<i>TBS_LEFT</i>	Displays tick marks on the left of a vertical slider. Can be used with the <i>TBS_RIGHT</i> style to display tick marks on both sides of the slider control.
<i>TBS_NOTICKS</i>	Creates a slider that does not display tick marks.
<i>TBS_RIGHT</i>	Displays tick marks on the right of a vertical slider. Can be used with the <i>TBS_LEFT</i> style to display tick marks on both sides of the slider control.
<i>TBS_TOP</i>	Displays tick marks on the top of a horizontal slider. Can be used with the <i>TBS_BOTTOM</i> style to display tick marks on both sides of the slider control.
<i>TBS_VERT</i>	Orients the slider vertically. If you do not specify an orientation, the slider is oriented horizontally.

Properties

The following table describes the properties for the TrackBar control:

Property	Description
<i>BackColor</i>	Background color of the control
<i>Enabled</i>	Set or get the enabled state of the control
<i>Height</i>	Get/Set the Height of the control.
<i>Left</i>	Get/Set left hand edge of the control

Property	Description
<i>LineSize</i>	Number of units to move when the arrow keys are pressed
<i>PageSize</i>	Number of units to move with the page up/page down keys
<i>Position</i>	Set or get the current position of the slider
<i>RangeMax</i>	Maximum value for the trackbar
<i>RangeMin</i>	Minimum value for the trackbar
<i>SelEnd</i>	End of the selection
<i>SelStart</i>	Start of the selection
<i>ThumbLength</i>	length of the thumb
<i>TickFrequency</i>	Frequency of the ticks
<i>Top</i>	Get/Set top edge of the control.
<i>Visible</i>	Set or get the visible state for the control
<i>Width</i>	Get/Set the width of the control.

Methods

The following table describes the methods for the TrackBar control:

Method	Description
<i>ClearSel</i>	Clear current selection
<i>ClearTicks</i>	Clear all the tick marks
<i>GetTick</i>	return the position of a tick
<i>Select</i>	Select a section of the trackbar
<i>SetTick</i>	Set a tick at the specified position
<i>TickCount</i>	Returns the number of ticks defined on the trackbar.

Events

The following table describes the events for the TrackBar control:

Event	Description
<i>DragEnter</i>	Dragged data has entered the control
<i>DragLeave</i>	Dragged data is leaving a control
<i>DragOver</i>	Data is being dragged over the control
<i>Drop</i>	Data has been dropped on this control
<i>Scroll</i>	The slider has been moved by the user

Version

4.0.1 Original

4.1 BackColor property added

TreeView

Description

TheTreeView control displays a tree structure 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.

This is a Common control.

Styles

The following table describes the styles for the TreeView control:

Style	Description
<i>TVS_EDITLABELS</i>	Enables users to edit the labels.
<i>TVS_HASBUTTONS</i>	Add a button to the left to expand collapsed items.
<i>TVS_HASLINES</i>	Draw lines to link child items to parent.
<i>TVS_LINESATROOT</i>	Draw lines linking root-level objects.
<i>TVS_SHOWSELALWAYS</i>	Show selection when focus is lost.
<i>WS_VISIBLE</i>	The control is visible

Properties

The following table describes the properties for the TreeView control:

Property	Description
<i>BackColor</i>	Background color of the control
<i>DragData</i>	Set the data to be dragged.
<i>DragMode</i>	Set the mode for dragging data from this control
<i>DragTarget</i>	Set the valid targets for data dragged from this control
<i>Enabled</i>	Set or get the enabled state of the control
<i>Font</i>	Set or get the font used with the control.
<i>Height</i>	Get/Set the Height of the control.
<i>Indent</i>	Get or set the indent for sub tree.
<i>Left</i>	Get/Set left hand edge of the control
<i>Redraw</i>	Redraw mode on/off.
<i>Top</i>	Get/Set top edge of the control.
<i>Visible</i>	Set or get the visible state for the control
<i>WebStyle</i>	Set/get the web style for this control
<i>Width</i>	Get/Set the width of the control.

Methods

The following table describes the methods for the TreeView control:

Method	Description
<i>Count</i>	The Count method returns the number of items in treeview
<i>DeleteAllItems</i>	Delete all items in the treeview
<i>DeleteItem</i>	Delete the specified item
<i>EditLabel</i>	The EditLabel method puts the tree into edit mode on the specified item

Method	Description
<i>EnsureVisible</i>	The EnsureVisible method makes an item visible in the tree control by scrolling and/or expanding
<i>Expand</i>	The Expand method expands or collapses a branch of the tree depending on the specified option
<i>GetItem</i>	The GetItem method returns details about an item of the tree view and returns True if successful
<i>GetItemData</i>	The GetItemData method returns a number stored against an item
<i>GetItemImage</i>	The GetItemImage method returns the current images used for an item
<i>GetItemState</i>	The GetItemState method returns the items state
<i>GetItemText</i>	The GetItemText method returns the text for an item
<i>GetNextItem</i>	The GetNextItem method returns an item with the specified relationship to item_id
<i>InsertItem</i>	The InsertItem method inserts a new item into a treeview with its details specified by the text,
<i>InsertLine</i>	The InsertLine method inserts a new item into a treeview and returns the item ID of the inserted
<i>ItemFromPoint</i>	Return the nearest item to the specified point
<i>SelectItem</i>	The SelectItem method select an item in a treeview and returns True if successful
<i>SetImageList</i>	The SetImageList method sets the image lists to use for the icons in the treeview

Method	Description
<i>SetItem</i>	The SetItem method sets the details of an item in the tree view and returns True if successful
<i>SetItemData</i>	The SetItemData method sets a number against an item and returns True if successful
<i>SetItemImage</i>	The SetItemImage method sets the image or images for an item
<i>SetItemState</i>	The SetItemState method sets an items state and returns True if successful
<i>SetItemText</i>	The SetItemText method sets the text for an item in a treeview and returns True if successful
<i>SortChildren</i>	The SortChildren method sorts the children of a branch and returns True if successful
<i>VisibleCount</i>	maximum number of complete items that can display.

Events

The following table describes the events for the TreeView control:

Event	Description
<i>BeginLabelEdit</i>	Label is about to be edited
<i>Click</i>	Left mouse clicked on the control
<i>DbClick</i>	Left mouse double clicked on the control
<i>DragEnd</i>	The dragging of data from this control has finished
<i>DragEnter</i>	Dragged data has entered the control
<i>DragLeave</i>	Dragged data is leaving a control
<i>DragOver</i>	Data is being dragged over the control
<i>DragStart</i>	Data is being dragged from this control

Event	Description
<i>Drop</i>	Data has been dropped on this control
<i>EndLabelEdit</i>	Editing of label is finishing
<i>ItemExpanded</i>	Item has been expanded
<i>ItemExpanding</i>	Item is expanding
<i>KeyDown</i>	key was pressed
<i>KillFocus</i>	Control has lost the focus
<i>OutOfMemory</i>	Out of memory
<i>Return</i>	Return key was pressed
<i>RightClick</i>	Right mouse button clicked upon the control
<i>RightDbClick</i>	Right mouse button was double clicked upon the control
<i>SelChanged</i>	Selected item has changed
<i>SelChanging</i>	Selected item is changing
<i>SetFocus</i>	Control has received the input focus

Version

- 4.0.1 new methods added
- 4.1 Standard properties added
 - 4.1.1 ItemFromPoint method added
- 4.2 Drag and drop properties and events
- 6.0 Added WebStyle property

UpDownButton

Description

The UpDownButton control displays a spin box to a dialog box.

This can be attached to an edit control to automatically increment or decrement its value. The attached control is known as the buddy control.

This is a Common control.

Styles

The following table describes the styles for the UpDownButton control:

Style	Description
<i>UDS_ALIGNLEFT</i>	Align to left of buddy control
<i>UDS_ALIGNRIGHT</i>	Align to the right of the buddy control
<i>UDS_ARROWKEYS</i>	Gives keyboard interface.
<i>UDS_HORZ</i>	Make it horizontal.
<i>UDS_NOTHOUSANDS</i>	Does not include comma (,) in for thousands.
<i>UDS_SETBUDDYINT</i>	Set caption of buddy control.
<i>UDS_WRAP</i>	Wrap when get to range.

Properties

The following table describes the properties for the UpDownButton control:

Property	Description
<i>Base</i>	Set number base decimal or hex.
<i>Buddy</i>	Set or get the name of edit control which is this object buddy.

Property	Description
<i>Enabled</i>	Set or get the enabled state of the control
<i>Height</i>	Get/Set the Height of the control.
<i>Left</i>	Get/Set left hand edge of the control
<i>Position</i>	Set or get current position.
<i>Range</i>	Set or get the range of this control.
<i>Top</i>	Get/Set top edge of the control.
<i>Visible</i>	Set or get the visible state for the control
<i>WebStyle</i>	Set/get the web style for this control
<i>Width</i>	Get/Set the width of the control.

Events

The following table describes the events for the UpDownButton control:

Event	Description
<i>DragEnter</i>	Dragged data has entered the control
<i>DragLeave</i>	Dragged data is leaving a control
<i>DragOver</i>	Data is being dragged over the control
<i>Drop</i>	Data has been dropped on this control

Version

6.0 Added WebStyle property

Script Control Properties

The following section lists alphabetically the properties of the scripted controls.

You can set or get the value of a property in a script as if it was a normal variable. It is specified by the name of the dialog box, the name of the control, and the name of the property separated by periods.

Example

To get the "Text" property for the "ID" edit control on the "Customer" dialog box:

```
value = Customer.ID.Text
```

To set the value of the same property:

```
Customer.ID.Text = value
```

AnchorIndex (ListBox control)

Description

Set or get the line which is the anchor in a multiple selection list box. The selection covers all the lines from the anchor index to the caret index.

Applies to

[CheckListBox](#), [ListBox](#)

BackColor

Description

Set or get the color used for the background of the control. The color number specifies a 24 bit color which is mapped to the nearest color for the specific PC. The color can be specified using the RGB() script function or the RGB_ script constants.

Applies to

[Animate](#), [CheckBox](#), [CheckListBox](#), [ColorButton](#), [CheckBox](#), [DateTime](#), [EditText](#), [Header](#), [ListBox](#), [ListView](#), [RadioButton](#), [ScrollBar](#), [Text](#), [TrackBar](#), [TreeView](#)

BackStyle

Description

Set or get the background style. When set to Transparent controls behind this control can be seen and the BackColor property is ignored.

Values

The back style can be one of the following values:

Value	Description
0	Transparent
1	Opaque

Applies to

[CheckBox](#), [RadioButton](#), [Text](#)

Base (UpDownButton control)

Description

Sets number base decimal (10) or hex (16).

Applies to

[UpDownButton](#)

BorderStyle (Divider control)

Description

Sets the style of the divider control. See table below.

Values

Value	Description
<i>0</i>	No border
<i>1</i>	3D edge on two sides (this it the default)

Applies to

[Divider](#)

Buddy (UpDownButton control)

Description

Set or get name of edit control which is this object buddy.

Applies to

[UpDownButton](#)

Caption

Description

Set or get the current value of the caption/text of the control.

Applies to

[CheckBox](#), [ColorButton](#), [GroupBox](#), [PushButton](#), [RadioButton](#), [Text](#)

CaretIndex (ListBox control)

Description

Set or get the line which has the focus rectangle in a multiple selection list box.

Applies to

[CheckListBox](#), [ListBox](#)

CellValue (Grid control)

Description

Value of the current cell.

Applies to

[Grid](#)

Check (CheckBox control)

Description

Set or get the current state of the check box. 1 if is checked, 0 if unchecked.

Applies to

[CheckBox](#)

Check (RadioButton control)

Description

Set or get the current value of the caption of the radio button.

Applies to

[RadioButton](#)

CheckIndex (CheckListBox control)

Description

This property is used to get or set the line numbers of the checked items. Multiple line numbers are separated by a cr .

Applies to

[CheckListBox](#)

CheckList (CheckListBox control)

Description

This property sets or get the entire contents of the check list box. Each line of the list is composed of a 1 or a 0 for the check mark followed by a tab and then the text of the line. Each line is separated by a cr.

Applies to

[CheckListBox](#)

CheckText (CheckListBox control)

Description

This property sets or gets the items that have a check against them. The list is delimited by the cr character. The is the default property that is read when the dialog box is shown.

Applies to

[CheckListBox](#)

ColHeaders (Grid control)

Description

Display column headers. Default is true.

Applies to

[Grid](#)

ColText (Grid control)

Description

Text for column headings (comma- or tab-separated).

Applies to

[Grid](#)

ColWidths (Grid control)

Description

Width of columns.

This property is write-only.

Values

Specify each column width in pixels separated by a comma plus:

Value	Description
<i>L</i>	for left justified text
<i>R</i>	for right justified text
<i>C</i>	centered text

Applies to

[Grid](#)

Column (Grid control)

Description

Current cell column.

Applies to

[Grid](#)

ColumnInfo

Description

This write-only property sets the information to be used for column headers.

Values

The format is: `column_text,width_in_pixels justification, ...` e.g. `.ColumnInfo = "Name,20,Date,8R"`. The justification for each column can be:

Value	Description
<i>L</i>	Left
<i>C</i>	Center
<i>R</i>	Right

Applies to

[ListView](#)

ColumnWidth (ListBox control)

Description

Set the width in pixels of the columns used in a listbox with the LBS_MULTIPLECOLUMN style.

Applies to

[CheckListBox](#), [ListBox](#)

Columns (Grid control)

Description

Number of columns in grid.

Applies to

[Grid](#)

CustomFormat (DateTime control)

Description

Set or get a custom format for displaying the date and/or time. Setting this property also sets the FormatType property to 3 (custom).

Values

The format is made up of the following parameters.

Value	Description
<i>h</i>	Hours
<i>hh</i>	Hours with leading zero
<i>H</i>	Same as h, but 24 hour format
<i>HH</i>	Same as hh, but 24 hour format
<i>m</i>	Minutes
<i>mm</i>	Minutes with leading zero
<i>s</i>	Seconds
<i>ss</i>	Seconds with leading zero
<i>t</i>	Abbreviated AM/PM designator
<i>tt</i>	Full AM/PM designator
<i>d</i>	Numeric day
<i>dd</i>	Numeric day with leading zero
<i>ddd</i>	Abbreviated day name
<i>dddd</i>	Full day name
<i>M</i>	Month
<i>MM</i>	Month with leading zero
<i>MMM</i>	Abbreviated month
<i>MMMM</i>	Full month name
<i>y</i>	Year without century

Value	Description
<i>yy</i>	Year with leading zero, without century
<i>yyyy</i>	Year including century
<i>text</i>	Quoted text passed straight through.

Applies to

[DateTime](#)

Data (Grid control)

Description

Dynamic array of data in grid

Applies to

[Grid](#)

DateMax (DateTime control)

Description

Set or get the maximum date and/or time for the date control. ValidMode has to be set to an appropriate value for this to take effect

Applies to

[DateTime](#)

DateMin (DateTime control)

Description

Set or get the minimum date and/or time for the date control. ValidMode has to be set to an appropriate value for this to take effect

Applies to

[DateTime](#)

DecimalPlaces (Calculator control)

Description

The number of decimal places to use with the control

Applies to

[Calculator](#)

Default (PushButton control)

Description

Set the push button to be the default push button. There is only one default push button so the current default push button will be changed to a non-default push button. Setting this property to false has no effect. Getting the value will return 1 (true) if this button is the current default push button or 0 (false) if it isn't

Applies to

[PushButton](#)

Display (Graphic control)

Description

Set or get how to display the image in the graphic control

Values

Can be one of the following values:

Value	Description
<i>0</i>	Stretch to fit the control (this is the default)
<i>1</i>	Clip the image in the control
<i>2</i>	Tile the image in the control

Applies to

[Graphic](#)

DragData

Description

Set the data to be dragged.

Applies to

[EditText](#), [ListBox](#), [ListView](#), [TreeView](#)

DragMode

Description

Set the mode for dragging data from this control.

Values

The drag mode is one of the following:

Value	Description
0	No dragging (the default)
1	Automatic dragging. The text of the selection will be dragged from the control and the selection will be deleted for a move operation.
2	Manual. A DragStart and DragEnd event will be generated to setup and complete the drag

Applies to

[EditText](#), [ListBox](#), [ListView](#), [TreeView](#)

DragTarget

Description

Set the valid targets for data dragged from this control

Values

The destination for the drag can be:

Value	Description
""	anywhere
"dialog"	to any control in the specified dialog box
"dialog.control"	to the specified control only.

Applies to

[EditText](#), [ListBox](#), [ListBox](#), [ListView](#), [TreeView](#)

Dropped (ComboBox control)

Description

Set or get the dropped status of the drop down list for the control. Setting this to true will drop down the control, false will close it up.

Applies to

[ComboBox](#)

Dropped (DateTime control)

Description

Set or get the dropped status of the drop down calendar for the control. Setting this to true will drop down the calendar, false will close it up.

Applies to

[DateTime](#)

Enabled

Description

Set or get the enabled state of the control.

Values

Value	Description
<i>True</i>	The control is enabled
<i>False</i>	The control is disabled

Applies to

[Animate](#), [CheckBox](#), [CheckListBox](#), [ColorButton](#), [CheckBox](#), [ComboBox](#), [DateTime](#), [EditText](#), [Graphic](#), [GraphicButton](#), [Grid](#), [GroupBox](#), [Header](#), [ListBox](#), [ListView](#), [ProgressBar](#), [PushButton](#), [RadioButton](#), [ScrollBar](#), [TabControl](#), [Text](#), [TrackBar](#), [TreeView](#), [UpDownButton](#)

EnterAction (Grid control)

Description

Set action when RETURN is pressed.

Values

Move to:

Value	Description
<i>0</i>	nowhere
<i>1</i>	up
<i>2</i>	right
<i>3</i>	down
<i>4</i>	left

Applies to

[Grid](#)

FastEntry (DateTime control)

Description

Set or get the fast entry mode flag. The default is True. When in fast entry mode when a valid entry is made in a field within the control it automatically moves on to the next field without needing to press LEFT ARROW or RIGHT ARROW keys. The TAB key will also move you between the fields when in this mode.

Applies to

[DateTime](#)

File (Animate control)

Description

Specifies the filename to use for the animation

Applies to

[Animate](#)

FloatingCells (Grid control)

Description

Allow floating cells.

Applies to

[Grid](#)

FocusItem (TabControl control)

Description

The index of a tab with focus or -1.

Applies to

[TabControl](#)

Font

Description

Set or get the font used with the control. If this property is "" the control uses the font defined for the dialog box.

Values

The format of the font is "font_name",point_size[,style]. The styles for the font are:

Value	Description
<i>FONT_Light</i>	Light weight font
<i>FONT_Normal</i>	Normal weight font
<i>FONT_Bold</i>	Heavy weight font
<i>FONT_Heavy</i>	Very Heavy weight font
<i>FONT_Italic</i>	Display font in italics
<i>FONT_Underline</i>	Underline the font
<i>FONT_StrikeOut</i>	Draw a line through the font

Applies to

[CheckBox](#), [CheckListBox](#), [ColorButton](#), [ComboBox](#), [EditText](#), [GroupBox](#), [Header](#), [ListBox](#), [PushButton](#), [RadioButton](#), [Text](#), [TreeView](#)

ForeColor

Description

Set or get the color used for the caption or text of the control. The color number specifies a 24 bit color which is mapped to the nearest color for the specific PC. The color can be specified using the RGB() script function or the RGB_ script constants.

Applies to

[CheckBox](#), [CheckListBox](#), [ColorButton](#), [DateTime](#), [EditText](#), [GroupBox](#), [Header](#), [ListBox](#), [ListView](#), [RadioButton](#), [Text](#)

FormatType (DateTime control)

Description

Set or get the type of the date control as in the following table. The default is 1 (local short date).

Values

Value	Description
0	Time only
1	Local short date
2	Local long date
3	Custom. This is set when the CustomFormat property is set

Applies to

[DateTime](#)

Height

Description

Get/Set the Height of the control.

If the dialog box is not shown the value is in dialog units.

If the dialog box is shown the value is in pixels.

Applies to

[Animate](#), [CheckBox](#), [CheckListBox](#), [ColorButton](#), [Calculator](#), [CheckBox](#), [ComboBox](#), [DateTime](#), [Divider](#), [CheckBox](#), [ComboBox](#), [DateTime](#), [EditText](#), [Graphic](#), [GraphicButton](#), [Grid](#), [GroupBox](#), [Header](#), [Icon](#), [ListBox](#), [ListView](#), [ProgressBar](#), [PushButton](#), [RadioButton](#), [ScrollBar](#), [TabControl](#), [Text](#), [TrackBar](#), [TreeView](#), [UpDownButton](#)

Horizontal (Divider control)

Description

Set orientation of the divider control.

True to do a horizontal divider, false for a vertical divider. Default false.

Applies to

[Divider](#)

HorizontalExtent

Description

Set or get the width in pixels of the list box or list part of the combo box control. Use when the list box/combo box has the WS_HSCROLL style to control the display of the horizontal scroll bar.

Applies to

[CheckListBox](#), [ComboBox](#), [ListBox](#)

Image

Description

Set the file name from which a bitmap will be read to be displayed on the control instead of a caption.

Applies to

[CheckBox](#), [PushButton](#), [RadioButton](#)

Image (Graphic control)

Description

Set or get the name of the file to use for the image of the graphic control. The image can also be specified directly as the text for the control when defining it.

Applies to

[Graphic](#)

Image (GraphicButton control)

Description

Set or get the name of the files to use for the images of the graphic button control.

Values

The image text is in the format: "normal_image; [pressed_image]; [focus_image]".

Value	Description
<i>normal_image</i>	The path of the image to display in the button.
<i>pressed_image</i>	The path of the image to display in the button when it is pressed. If pressed_image is not specified, the normal_image graphic is displayed.
<i>focus_image</i>	The path of the image to display when the button has input focus. If the focus_image is not specified, the normal_image graphic is displayed

Applies to

[GraphicButton](#)

ImageHeight (Graphic control)

Description

Read the height in pixels of the image displayed in the graphic control. This property is only available when dialog box containing the control has been shown. If the image height can not be determined it returns 0.

Applies to

[Graphic](#)

ImageWidth (Graphic control)

Description

Read the width in pixels of the image displayed in the graphic control. This property is only available when dialog box containing the control has been shown. If the image width can not be determined it returns 0.

Applies to

[Graphic](#)

Indent (TreeView control)

Description

Get or set the indent for sub tree.

Applies to

[TreeView](#)

LabelWidths (Header control)

Description

A comma separated list of widths and justifications (L - Left, C - Center, R - Right).

Applies to

[Header](#)

Labels (Header control)

Description

A comma or carriage return (when getting the result) separated list of text for headings.

Applies to

[Header](#)

Labels (TabControl control)

Description

A comma/carriage return separated list of text for headings (carriage return separated when getting the result).

Applies to

[TabControl](#)

Left

Description

Get/Set left hand edge of the control.

This value is relative to the client area of the dialog box.

If the dialog box is not shown the value is in dialog units.

If the dialog box is shown the value is in pixels.

Applies to

[Animate](#), [CheckBox](#), [CheckListBox](#), [ColorButton](#), [Calculator](#), [CheckBox](#), [ComboBox](#), [DateTime](#), [Divider](#), [CheckBox](#), [ComboBox](#), [DateTime](#), [EditText](#), [Graphic](#), [GraphicButton](#), [Grid](#), [GroupBox](#), [Header](#), [Icon](#), [ListBox](#), [ListView](#), [ProgressBar](#), [PushButton](#), [RadioButton](#), [ScrollBar](#), [TabControl](#), [Text](#), [TrackBar](#), [TreeView](#), [UpDownButton](#)

LimitText

Description

Set or get the maximum number of character that can be entered into an edit or combo box control. Set to 0 for unlimited characters.

Applies to

[ComboBox](#), [EditText](#)

LineSize (TrackBar control)

Description

Set and get the amount the slider moves in response to keyboard input from the arrow keys, such as the RIGHT ARROW or DOWN ARROW keys. The default is 1.

Applies to

[TrackBar](#)

List

Description

Set or get all the items in a listbox list or the list part of the combo box.

Applies to

[CheckListBox](#), [ComboBox](#), [ListBox](#)

MaximumPosition (Divider control)

Description

Maximum position in pixels that the control can be dragged.

If this is less than or equal to 0 it is the distance from the right of the parent dialog box.

Applies to

[Divider](#)

MinimumPosition (Divider control)

Description

Minimum position in pixels that the control can be dragged.

Applies to

[Divider](#)

Modified

Description

Get or Set the modified flag for the control.

Applies to

[EditText](#)

NoEdit (DateTime control)

Description

Set or get which fields of the date time control can not be edited

Values

This is a combination of 1 or more values from the following table which specify which fields are fixed.

Value	Description
0	No fields have been fixed
1	Year
2	Month
4	Day
8	Hour
16	Minute
32	Second

Applies to

[DateTime](#)

Optional (DateTime control)

Description

Set or get if the date time control can contain a null date and time. When set a date can be set to null by the user by pressing the delete key. number can be one of the following values. An optional date will only return a non null value when all the fields have been set.

Values

Value	Description
0	Field is not optional. A value must be entered
1	Field is optional. Display an underscore for null fields within the control
2	Field is optional. Display a space for null fields within the control

Applies to

[DateTime](#)

PageSize (TrackBar control)

Description

Set and get the amount the slider moves in response to keyboard input of the PAGE UP or PAGE DOWN keys, or mouse input when clicking on the trackbar's channel. The default is 20.

Applies to

[TrackBar](#)

PasswordChar (EditText control)

Description

Set or get the character to show when enter text into an edit control with the ES_PASSWORD style.

Applies to

[EditText](#)

Position (ProgressBar control)

Description

Sets position. This is the default property.

Applies to

[ProgressBar](#)

Position (ScrollBar control)

Description

Set or get the position of the thumb in the scroll bar. This is the default property.

Applies to

[ScrollBar](#)

Position (TrackBar control)

Description

Set and get the position of the trackbar slider. This is the default property.

Applies to

[TrackBar](#)

Position (UpDownButton control)

Description

Set or get current position (Default).

Applies to

[UpDownButton](#)

Radio (GroupBox control)

Description

Set or get the current selected radio button in the group box. This is the default property.

Applies to

[GroupBox](#)

Range (ProgressBar control)

Description

Sets the minimum and maximum positions of the progress bar (default is 0 to 100)

Applies to

[ProgressBar](#)

Range (ScrollBar control)

Description

Set or get the minimum and maximum values for position of the thumb in the scroll bar.

Applies to

[ScrollBar](#)

Range (UpDownButton control)

Description

Set or get the range of this control.

Applies to

[UpDownButton](#)

RangeMax (TrackBar control)

Description

Set and get the maximum value for the slider. If the current position is greater than the new value the position is moved to the new value. The default is 100.

Applies to

[TrackBar](#)

RangeMin (TrackBar control)

Description

Set and get the minimum value for the slider. If the current position is less than the new value the position is moved to the new value. The default is 0.

Applies to

[TrackBar](#)

ReadOnly

Description

Set or get the read only state of the control.

Applies to

[EditText](#), [Grid](#)

Redraw

Description

Redraw mode on/off.

Applies to

[Grid](#), [TreeView](#)

Row (Grid control)

Description

Current cell row.

Applies to

[Grid](#)

RowHeaders (Grid control)

Description

Display row header (default is true).

Applies to

[Grid](#)

RowHeights (Grid control)

Description

Height of rows (comma-separated) in dialog units.

Applies to

[Grid](#)

RowText (Grid control)

Description

Text for row headings (comma or carriage-return separated).

Applies to

[Grid](#)

Rows (Grid control)

Description

Number of rows in grid.

Applies to

[Grid](#)

SelEnd (TrackBar control)

Description

Set and get the ending position of the current selection range in a trackbar.

Applies to

[TrackBar](#)

SellIndex

Description

Get or Set the index of the currently selected line from a list box or the list part of the combo box.

Applies to

[ComboBox](#)

SelIndex (ListBox control)

Description

Get or Set the index of the currently selected line from the list.

Applies to

[CheckListBox](#), [ListBox](#)

SelStart (TrackBar control)

Description

Set and get the starting position of the current selection range in a trackbar.

Applies to

[TrackBar](#)

SelText (ListBox control)

Description

Set or get the current value of the selection. This is the default property when reading when the dialog box is shown. Also see The Update Property.

Applies to

[CheckListBox](#), [ListBox](#)

SelectMode (Grid control)

Description

The SelectMode property specifies the options available to the user to select cells in a grid.

Values

The selection mode is 0 or one or more of the following added together.

Value	Description
0	No selection
1	Row selection
2	Column selection
4	Select whole table
8	Select cells
16	Multiple selections
32	Shift extends selection
64	Allow selection with shift and keyboard
255	Enable all selections

Applies to

[Grid](#)

SelectedItem (TabControl control)

Description

The index of the currently selected tab or -1.

Applies to

[TabControl](#)

SortColOnDbIClick (Grid control)

Description

Automatically sort by column if user double clicks on header (default is false).

Applies to

[Grid](#)

SortRowOnDblClick (Grid control)

Description

Automatically sort by row if user double clicks on header (default is false).

Applies to

[Grid](#)

Step (ProgressBar control)

Description

Sets the step increment (default is 10).

Applies to

[ProgressBar](#)

TabStops (EditText control)

Description

Set the tabstops to use in the edit control. This is a comma separated list of the tab stop positions in dialog units.

Applies to

[EditText](#)

TabStops (ListBox control)

Description

Set the position and justification of the tab stops in a list box control which uses the LBS_USETABSTOPS style.

Values

Each entry is the width followed immediately by a single letter:

Value	Description
<i>L</i>	Left align
<i>C</i>	Center
<i>R</i>	Right align
<i>E</i>	Reduce size of a displayed file name to the column size by showing ellipses

Applies to

[ListBox](#)

Text

Description

Set or get the text from the control.

Applies to

[ComboBox](#), [EditText](#)

TextBackColor (ListView control)

Description

Set or get the background color for text.

Applies to

[ListView](#)

ThumbLength (TrackBar control)

Description

Set and get the length of the trackbars slider.

Applies to

[TrackBar](#)

TickFrequency (TrackBar control)

Description

Set the frequency at which the tick marks appear.

Applies to

[TrackBar](#)

Top

Description

Get/Set top edge of the control.

This value is relative to the client area of the dialog box.

If the dialog box is not shown the value is in dialog units.

If the dialog box is shown the value is in pixels.

Applies to

[Animate](#), [CheckBox](#), [CheckListBox](#), [ColorButton](#), [Calculator](#), [CheckBox](#), [ComboBox](#), [DateTime](#), [Divider](#), [CheckBox](#), [ComboBox](#), [DateTime](#), [EditText](#), [Graphic](#), [GraphicButton](#), [Grid](#), [GroupBox](#), [Header](#), [Icon](#), [ListBox](#), [ListView](#), [ProgressBar](#), [PushButton](#), [RadioButton](#), [ScrollBar](#), [TabControl](#), [Text](#), [TrackBar](#), [TreeView](#), [UpDownButton](#)

TopIndex

Description

Set or get the line number of the top line visible in the list part of the control.

Applies to

[CheckListBox](#), [ComboBox](#), [ListBox](#)

TrackHeight (Grid control)

Description

Enable user to modify row heights (default is true).

Applies to

[Grid](#)

TrackWidth (Grid control)

Description

Enable user to modify column widths (default true)

Applies to

[Grid](#)

UpdateMode (CheckListBox control)

Description

Set or get the update mode which specifies which property is updated when the dialog box containing the list box is closed using DialogBox End <dlgname>, True to update the variables.

Values

The value of mode are taken from the following table and can be either 0, 8 or a combination of the other values added together.

Value	Description
0	The list property is updated with the currently checked items. This has the effect of putting the result in the default value. This is the default.
1	The SelText property is updated with the final text of the combo box
2	The SelIndex property is updated with the line number of the list item selected
4	The list property is updated with the final list in the list box
8	No updates take place
16	The CheckText property is updated with the names of the list items that are checked
32	The CheckIndex property is updated with the indices of the items that are checked
64	The CheckList property is updated with all the list items with a check indicator.

Applies to

[CheckListBox](#)

UpdateMode (ComboBox control)

Description

Set or get the update mode which specifies which property is updated when the dialog box containing the combo box is closed using DialogBox End <dlgname>, True to update the variables.

Values

The value of mode are taken from the following table and can be either 0, 8 or a combination of the other values added together

Value	Description
0	The list property is updated with the current selection. This has the effect of putting the result in the default value. This is the default.
1	The Text property is updated with the final text of the combo box
2	The index property is updated with the line number of the list item selected
4	The list property is updated with the final list in the combo box
8	No updates take place

Applies to

[ComboBox](#)

UpdateMode (ListBox control)

Description

Set or get the update mode which specifies which property is updated when the dialog box containing the list box is closed using DialogBox End <dlgname>,True to update the variables.

Values

The value of mode are taken from the following table and can be either 0, 8 or a combination of the other values added together.

Value	Description
0	The list property is updated with the current selection. This has the effect of putting the result in the default value. This is the default.
1	The SelText property is updated with the final text of the combo box
2	The SelIndex property is updated with the line number of the list item selected
4	The list property is updated with the final list in the list box
8	No updates take place

Applies to

[ListBox](#)

ValidMode (DateTime control)

Description

Set or get the validation of the date and/or time according to the following table. Set the DateMin and DateMax properties to specify the range for the date.

Values

Value	Description
0	None
1	Time only
2	Date only
3	Date and time

Applies to

[DateTime](#)

Value (Calculator control)

Description

The current value of the control

Applies to

[Calculator](#)

Value (DateTime control)

Description

Set or get the current value of the date time control. This is the default property. The `date_time_string` is parsed to read the date and time from it.

Values

The return value depends on the format type as follows:

Value	Description
<i>Time</i>	Returns the time as hh:mm
<i>Short date</i>	Returns the date only in local format.
<i>Long date</i>	Returns the date only in local format.
<i>Custom date</i>	Returns the date in local format followed by a space and then the time.

Applies to

[DateTime](#)

View (ListView control)

Description

This write-only property changes the style of the list view.

Values

The style can be one of the following numbers:

Value	Description
<i>0</i>	Icon view
<i>1</i>	Small icon view
<i>2</i>	List view
<i>3</i>	Report view

Applies to

[ListView](#)

Visible

Description

The visible property sets or gets the visible state for a control.

Values

Value	Description
<i>True</i>	The control is visible
<i>False</i>	The control is invisible

Applies to

[Animate](#), [CheckBox](#), [CheckListBox](#), [ColorButton](#), [CheckBox](#), [ComboBox](#), [DateTime](#), [EditText](#), [Graphic](#), [GraphicButton](#), [Grid](#), [GroupBox](#), [Header](#), [Icon](#), [ListBox](#), [ListView](#), [ProgressBar](#), [PushButton](#), [RadioButton](#), [ScrollBar](#), [TabControl](#), [Text](#), [TrackBar](#), [TreeView](#), [UpDownButton](#)

WebStyle

Description

This property allows the name of the web style to be set or read for this control.

The web styles are defined in a cascading style sheet and are used by the Java Client only. All other versions ignore this command. The web style allows the font and colours for a dialog box or control to be changed.

See the Java Client documentation on how to specify the style sheet used.

Applies to

[CheckBox](#), [DateTime](#), [UpDownButton](#), [TreeView](#), [Text](#), [TabControl](#), [ScrollBar](#), [Rectangle](#), [RadioButton](#), [PushButton](#), [ProgressBar](#), [ListView](#), [ListBox](#), [GroupBox](#), [GraphicButton](#), [Graphic](#), [EditText](#), [DateTime](#), [ComboBox](#), [ColorButton](#)

Width

Description

Get/Set the width of the control.

If the dialog box is not shown the value is in dialog units.

If the dialog box is shown the value is in pixels.

Applies to

[Animate](#), [CheckBox](#), [CheckListBox](#), [ColorButton](#), [Calculator](#), [CheckBox](#), [ComboBox](#), [DateTime](#), [Divider](#), [CheckBox](#), [ComboBox](#), [DateTime](#), [EditText](#), [Graphic](#), [GraphicButton](#), [Grid](#), [GroupBox](#), [Header](#), [Icon](#), [ListBox](#), [ListView](#), [ProgressBar](#), [PushButton](#), [RadioButton](#), [ScrollBar](#), [TabControl](#), [Text](#), [TrackBar](#), [TreeView](#), [UpDownButton](#)

Script Control Methods

Methods are functions or commands that are specific to a particular control.

To invoke a method use the dialog name, the control name and the method name separated by periods.

A method that is a function can also be used as if it were a command if the return value is not required in the script.

Example

To add a new line with the text "dog" to the list box "Pets" in the dialog box "PetShop" use the "AddLine" method:

```
PetShop.Pets.AddLine "dog"
```

To add a new line "cat" to the list and return the line number where it was added use:

```
added_line = PetShop.Pets.AddLine("cat")
```

AddLine (ComboBox)

Syntax

AddLine *text*

Description

Adds a line to the list part of the combo box and returns the line number of the newly added line.

Parameters

The following table describes the parameters of the AddLine method:

Parameters	Description
<i>text</i>	The text of the line to add

Return Value

The index of the newly added line

Applies to

[ComboBox](#)

AddLine (ListBox)

Syntax

AddLine *line_text*

Description

Adds a line to the list part of the combo box and returns the line number of the newly added line.

Parameters

The following table describes the parameters of the AddLine method:

Parameters	Description
<i>line_text</i>	The text to add to the end of the list

Return Value

line number of the newly added line.

Applies to

[CheckListBox](#), [ListBox](#)

AdjustRect (TabControl)

Syntax

AdjustRect *larger, rect_name*

Description

The AdjustRect method calculates the size of a tab control to fit a specific area.

Parameters

The following table describes the parameters of the AdjustRect method:

Parameters	Description
<i>larger</i>	If True, the rectangle is set to the area required for the display area and the area required for the control is returned. If False, the area for the control is specified and the area of the display is returned.
<i>rect_name</i>	Name of variable holding the rectangle to be calculated

Applies to

[TabControl](#)

Arrange (ListView)

Syntax

Arrange code, ok

Description

The Arrange method arranges items in listview and returns True if successful.

Parameters

The following table describes the parameters of the Arrange method:

Parameters	Description
<i>code</i>	The order to arrange items. See below
<i>ok</i>	Returns true if successful

Values for code

Can be one of the following:

Value	Description
<i>0</i>	Default. Aligns items according to the list view controls current alignment styles
<i>1</i>	Left. Aligns items along the left edge of the window.
<i>2</i>	Top. Aligns items along the top edge of the window.
<i>5</i>	Snap to grid. Snaps all icons to the nearest grid position.

Applies to

[ListView](#)

CharFromPos (EditText)

Syntax

CharFromPos *x_pos, y_pos*

Description

Returns the character index of the character at the specified screen position

Parameters

The following table describes the parameters of the CharFromPos method:

Parameters	Description
<i>x_pos</i>	screen pixel x
<i>y_pos</i>	screen pixel y

Return Value

Character index of the screen position

Applies to

[EditText](#)

Version

4.1.0 Original

CheckAll (CheckListBox)

Syntax

CheckAll *set*

Description

This method allows all the lines in a check list box to be checked or unchecked.

Parameters

The following table describes the parameters of the CheckAll method:

Parameters	Description
<i>set</i>	True to check all the items or False to uncheck all the items.

Applies to

[CheckListBox](#)

Example

Select a meeting for every day in our days check list box

```
MeetingDays.days.CheckAll True
```

Clear

Syntax

Clear

Description

Clears the current selection from the control

Parameters

None

Applies to

[ComboBox](#), [EditText](#)

ClearRange (Grid)

Syntax

ClearRange *from_row, from_col, to_row, to_col, clear_style*

Description

The ClearRange method clears cells in a grid.

Parameters

The following table describes the parameters of the ClearRange method:

Parameters	Description
<i>from_row</i>	Start row.
<i>from_col</i>	Start column.
<i>to_row</i>	End row.
<i>to_col</i>	End column.
<i>clear_style</i>	Clear style information as well as text (default is false)

Applies to

[Grid](#)

ClearSel (ListBox)

Syntax

ClearSel

Description

Unselects all the lines that are currently selected.

Parameters

None

Applies to

[CheckListBox](#), [ListBox](#)

ClearSel (TrackBar)

Syntax

ClearSel *redraw*

Description

Clears the current selection

Parameters

The following table describes the parameters of the ClearSel method:

Parameters	Description
<i>redraw</i>	True to redraw the selection immediately

Applies to

[TrackBar](#)

ClearTicks (TrackBar)

Syntax

ClearTicks *redraw*

Description

Clears all the tick marks from the trackbar

Parameters

The following table describes the parameters of the ClearTicks method:

Parameters	Description
<i>redraw</i>	True to redraw the selection immediately

Applies to

[TrackBar](#)

Close (Animate)

Syntax

Close

Description

The Close method closes an AVI file.

Parameters

None

Applies to

[Animate](#)

Copy

Syntax

Copy

Description

Copy the current selection to the clipboard

Parameters

None

Applies to

[ComboBox](#), [EditText](#), [Grid](#)

Version

4.0.2 Added to Grid control

Count (ComboBox)

Syntax

Count

Description

Returns the number of lines in the list part of the combo box.

Parameters

None

Return Value

Number of lines in the list part of the combo box.

Applies to

[ComboBox](#)

Count (Header)

Syntax

Count

Description

The Count method returns the number of items in a header.

Parameters

None

Return Value

The number of items in the header

Applies to

[Header](#)

Count (ListBox)

Syntax

Count

Description

Returns the number of lines in the list

Parameters

None

Return Value

Number of lines in the list

Applies to

[CheckListBox](#), [ListBox](#)

Count (ListView)

Syntax

Count *items*

Description

The Count method returns the number of items in the listview

Parameters

The following table describes the parameters of the Count method:

Parameters	Description
<i>items</i>	Number of items in the list view

Applies to

[ListView](#)

Count (TabControl)

Syntax

Count

Description

The Count method returns the number of labels in the tab control.

Parameters

None

Return Value

number of labels in the tab control.

Applies to

[TabControl](#)

Count (TreeView)

Syntax

Count

Description

The Count method returns the number of items in treeview.

Parameters

None

Return Value

Number of items in the tree view

Applies to

[TreeView](#)

Cut

Syntax

Cut

Description

Cuts the current selection from the control and places it on the clipboard.

Parameters

None

Applies to

[ComboBox](#), [EditText](#), [Grid](#)

Version

4.0.2 Added to Grid control

DeleteAllItems (ListView)

Syntax

DeleteAllItems

Description

The DeleteAllItems method deletes all the Items in a list view and returns True if successful.

Parameters

None

Return Value

True if successful

Applies to

[ListView](#)

DeleteAllItems (TabControl)

Syntax

DeleteAllItems

Description

The DeleteAllItems method deletes all items and returns True if successful.

Parameters

None

Return Value

True if successful.

Applies to

[TabControl](#)

DeleteAllItems (TreeView)

Syntax

DeleteAllItems

Description

Deletes all items from the treeview control and returns true if the deletion was successful.

Parameters

None

Return Value

True if the deletion was successful

Applies to

[TreeView](#)

Version

4.0.1 Original

DeleteCols (Grid)

Syntax

DeleteCols *delete_from*, [*number*]

Description

The DeleteCols method deletes columns from a grid.

Parameters

The following table describes the parameters of the DeleteCols method:

Parameters	Description
<i>delete_from</i>	Column to start delete.
<i>number</i>	Number of rows to delete (default 1).

Applies to

[Grid](#)

DeleteColumn (ListView)

Syntax

DeleteColumn *column*

Description

The DeleteColumn method deletes a column from a listview and returns True is successful.

Parameters

The following table describes the parameters of the DeleteColumn method:

Parameters	Description
<i>column</i>	Column to delete

Return Value

True if successful

Applies to

[ListView](#)

DeleteItem (Header)

Syntax

DeleteItem *index*

Description

The DeleteItem method deletes an item from a header and returns a nonzero value if successful.

Parameters

The following table describes the parameters of the DeleteItem method:

Parameters	Description
<i>index</i>	The index of item to delete

Return Value

True if the item was successfully deleted

Applies to

[Header](#)

DeleteItem (ListView)

Syntax

DeleteItem *item_no*

Description

The DeleteItem method deletes an item from a listview and returns True if successful.

Parameters

The following table describes the parameters of the DeleteItem method:

Parameters	Description
<i>item_no</i>	Index of item to delete.

Return Value

True if successful

Applies to

[ListView](#)

DeleteItem (TabControl)

Syntax

DeleteItem *index*

Description

The DeleteItem method deletes the specified item and returns True if successful.

Parameters

The following table describes the parameters of the DeleteItem method:

Parameters	Description
<i>index</i>	Index of item to delete.

Return Value

True if successful.

Applies to

[TabControl](#)

DeleteItem (TreeView)

Syntax

DeleteItem *item_id*

Description

Deletes the specified item and returns true if the item was successfully deleted.

Parameters

The following table describes the parameters of the DeleteItem method:

Parameters	Description
<i>item_id</i>	Item to delete

Return Value

True if item was successfully deleted.

Applies to

[TreeView](#)

Version

4.0.1

DeleteLine (ComboBox)

Syntax

DeleteLine *line_number*

Description

Deletes the specified line and returns the number of lines left in the list or -1 if an invalid line number is specified.

Parameters

The following table describes the parameters of the DeleteLine method:

Parameters	Description
<i>line_number</i>	The number of the line to delete

Return Value

The number of lines left in the list or -1 if an invalid line number is specified.

Applies to

[ComboBox](#)

DeleteLine (ListBox)

Syntax

DeleteLine *line_number*

Description

Deletes the specified line and returns the number of lines left in the list or -1 if an invalid line number is specified.

Parameters

The following table describes the parameters of the DeleteLine method:

Parameters	Description
<i>line_number</i>	Number of the line to delete

Return Value

Number of lines left in the list or -1 if an invalid line number is specified

Applies to

[CheckListBox](#), [ListBox](#)

DeleteRows (Grid)

Syntax

DeleteRows *delete_from*, [*number*]

Description

The DeleteRows method deletes rows from a grid.

Parameters

The following table describes the parameters of the DeleteRows method:

Parameters	Description
<i>delete_from</i>	Row to start delete.
<i>number</i>	Number of rows to delete (default 1).

Applies to

[Grid](#)

DeltaPos (ProgressBar)

Syntax

DeltaPos *step*

Description

The DeltaPos method moves the progress bar by a given step.

Parameters

The following table describes the parameters of the DeltaPos method:

Parameters	Description
<i>step</i>	The distance to move the progress bar

Applies to

[ProgressBar](#)

EditLabel (ListView)

Syntax

EditLabel *item_no*

Description

The EditLabel method edits an item in a listview and returns a nonzero value if successful.

Parameters

The following table describes the parameters of the EditLabel method:

Parameters	Description
<i>item_no</i>	Index of label to edit. item_no of -1 means cancel current edit

Return Value

Non-zero if successful

Applies to

[ListView](#)

EditLabel (TreeView)

Syntax

EditLabel *item_id*

Description

The EditLabel method puts the tree into edit mode on the specified item.

Parameters

The following table describes the parameters of the EditLabel method:

Parameters	Description
<i>item_id</i>	Item to edit.

Applies to

[TreeView](#)

EditSelect (ComboBox)

Syntax

EditSelect

Description

Sets the current selection in the edit part of the combo box.

Parameters

None

Applies to

[ComboBox](#)

EditSelectInfo (ComboBox)

Syntax

EditSelectInfo *from_char, to_char*

Description

Sets the *from_char_var* variable and the *to_char_var* variable with the values for the start and finish of the current selection in the edit part of the combo box.

If there is no selection it sets the cursor position into both variables. The return value is true if there is a selection.

Parameters

The following table describes the parameters of the EditSelectInfo method:

Parameters	Description
<i>from_char</i>	Variable to receive the beginning of the selection
<i>to_char</i>	Variable to receive the end of the selection

Return Value

True if a selection was found

Applies to

[ComboBox](#)

Empty (ComboBox)

Syntax

Empty

Description

Clears all the lines from the list part of the combo box.

Parameters

None

Applies to

[ComboBox](#)

Empty (ListBox)

Syntax

Empty

Description

Clears all the lines from the list

Parameters

None

Applies to

[CheckListBox](#), [ListBox](#)

EnsureVisible (ListView)

Syntax

EnsureVisible *item_no*, *partial_ok*

Description

The EnsureVisible method scrolls the listview to show the specified item and returns TRUE if the item was made visible.

Parameters

The following table describes the parameters of the EnsureVisible method:

Parameters	Description
<i>item_no</i>	Index of item
<i>partial_ok</i>	True if ok to only show part of the item.

Return Value

True if the item was made visible

Applies to

[ListView](#)

EnsureVisible (TreeView)

Syntax

EnsureVisible *item_id*

Description

The EnsureVisible method makes an item visible in the tree control by scrolling and/or expanding branches and returns True if successful.

Parameters

The following table describes the parameters of the EnsureVisible method:

Parameters	Description
<i>item_id</i>	Item to make visible.

Return Value

True if successful.

Applies to

[TreeView](#)

Expand (TreeView)

Syntax

Expand *option, item_id*

Description

The Expand method expands or collapses a branch of the tree depending on the specified option.

Parameters

The following table describes the parameters of the Expand method:

Parameters	Description
<i>option</i>	Expand/Collapse the branch options. See below.
<i>item_id</i>	ID of the branch of the tree to expand or collapse.

Values for option

Value	Description
<i>1</i>	Collapse the branch.
<i>2</i>	Expand the branch.
<i>3</i>	Collapse the list if it is currently expanded or expand the list if it is currently collapsed.
<i>32769</i>	Collapses the list and removes the child items.

Applies to

[TreeView](#)

FillRange (Grid)

Syntax

FillRange *from_row, from_col, to_row, to_col, value*

Description

The FillRange method inserts a value into a specified range of columns and rows.

Parameters

The following table describes the parameters of the FillRange method:

Parameters	Description
<i>from_row</i>	Starting row
<i>from_col</i>	Starting column
<i>to_row</i>	Ending row.
<i>to_col</i>	Ending column.
<i>value</i>	Value to put in all cells in the range.

Applies to

[Grid](#)

Find (ComboBox)

Syntax

Find *start_line, text*

Description

Searches for the line in the list that starts with the specified text. Searching starts from the given line and wraps round from the beginning of the list. Returns the line number where the text is found or 0. Set *start_line* to 0 to search the whole list.

Parameters

The following table describes the parameters of the Find method:

Parameters	Description
<i>start_line</i>	Line to start search from. 0 for the whole list.
<i>text</i>	The text that the found line must start with

Return Value

The line number where the text is found or 0.

Applies to

[ComboBox](#)

Find (Grid)

Syntax

Find *text, row_var, col_var, options*

Description

The Find methods searches the grid for the specified text.

Parameters

The following table describes the parameters of the Find method:

Parameters	Description
<i>text</i>	The text to search for
<i>row_var</i>	The name of a variable whose value specified the row to start the search. If the text is found the variables value is updated to the row the text was found in. Set the variable to 0 to start the search from the beginning of the grid
<i>col_var</i>	The name of a variable whose value specified the column to start the search. If the text is found the variables value is updated to the column the text was found in
<i>options</i>	The Options for the search. See below. Default 0.

Values for options

The following numbers can be added together to specify the options for

Value	Description
1	Search upward from the specified cell. If rowVar is set to 0 the search starts from the end of the grid.
2	The search text must match the case of the found text in the grid.

Return Value

This method returns true if the text was found and updates the row_var and col_var with the location of the found text. If the text is not found, false is returned and row_var and column_var are left unchanged.

Applies to

[Grid](#)

Version

4.0.2

Find (ListBox)

Syntax

Find *start_line, text*

Description

Searches for the line in the list that starts with the specified text. Searching starts from the given line and wraps round from the beginning of the list. Returns the line number where the text is found or 0. Set *start_line* to 0 to search the whole list.

Parameters

The following table describes the parameters of the Find method:

Parameters	Description
<i>start_line</i>	Line to start search from. 0 for the whole list.
<i>text</i>	The text that the found line must start with

Return Value

The line number where the text is found or 0.

Applies to

[CheckListBox](#), [ListBox](#)

FindExact (ComboBox)

Syntax

FindExact *start_line*, *text*

Description

Searches for the line in the list with the specified text. Searching starts from the given line and wraps round from the beginning of the list. Returns the line number where the text is found or 0. Set *start_line* to 0 to search the whole list.

Parameters

The following table describes the parameters of the FindExact method:

Parameters	Description
<i>start_line</i>	The line number from which to start the search. 0 to search the whole list.
<i>text</i>	The text to search for

Return Value

The line number where the text was found or 0.

Applies to

[ComboBox](#)

FindExact (ListBox)

Syntax

FindExact *start_line, text*

Description

Searches for the line in the list with the specified text. Searching starts from the given line and wraps round from the beginning of the list. Returns the line number where the text is found or 0. Set *start_line* to 0 to search the whole list.

Parameters

The following table describes the parameters of the FindExact method:

Parameters	Description
<i>start_line</i>	The line number from which to start the search. 0 to search the whole list.
<i>text</i>	The text to search for

Return Value

The line number where the text was found or 0.

Applies to

[CheckListBox](#), [ListBox](#)

FindItem (ListView)

Syntax

FindItem *start, text, [begin], [wrap], [data], from_x, from_y, [direction]*

Description

The FindItem method finds an item with the specified text or data. and returns: the index of the item or -1.

Parameters

The following table describes the parameters of the FindItem method:

Parameters	Description
<i>start</i>	Item to start.
<i>text</i>	Text to match.
<i>begin</i>	Match start of text only.
<i>wrap</i>	Restart search from beginning if not found
<i>data</i>	Data value to match.
<i>from_x</i>	Point to start search from x.
<i>from_y</i>	Point to start search from y.
<i>direction</i>	Direction for search from nearest point: 37 - Left from specified point, 38 - Up from specified point, 39 - Right from specified point, 40 - Down from specified point

Return Value

Returns the item number of the found item or -1 if no item was found

Applies to

[ListView](#)

Example

Find rover

```
rover_idx = petshop.petlist.FindItem(-1, "Rover")
```

FirstVisibleLine (EditText)

Syntax

FirstVisibleLine

Description

Returns the character index of the first visible line in the edit control.

Parameters

None

Return Value

The character index of the first visible line in the edit control.

Applies to

[EditText](#)

GetCellAttribute (Grid)

Syntax

GetCellAttribute *row, column, attribute*

Description

The GetCellAttribute method returns the current value of an attribute for a cell. See SetCellAttribute for information on attributes.

Parameters

The following table describes the parameters of the GetCellAttribute method:

Parameters	Description
<i>row</i>	Row number.
<i>column</i>	Column Number
<i>attribute</i>	The name of the attribute

Return Value

The current value of the attribute at the specified cell

Applies to

[Grid](#)

GetCellAutoSize (Grid)

Syntax

GetCellAutoSize *row, column*

Description

The GetCellAutoSize method returns whether the cell height will grow to fit the text.

Parameters

The following table describes the parameters of the GetCellAutoSize method:

Parameters	Description
<i>row</i>	Row number.
<i>column</i>	Column number.

Return Value

True if auto size is set in the specified cell

Applies to

[Grid](#)

GetCellBackColor (Grid)

Syntax

GetCellBackColor *row, column*

Description

The GetCellBackColor method returns the background color of a cell.

Parameters

The following table describes the parameters of the GetCellBackColor method:

Parameters	Description
<i>row</i>	Row number
<i>column</i>	Column number.

Return Value

Current attribute color

Applies to

[Grid](#)

GetCellBorder (Grid)

Syntax

GetCellBorder *row, column, position*

Description

The GetCellBorder method returns the color, style, and width of a cell border.

Parameters

The following table describes the parameters of the GetCellBorder method:

Parameters	Description
<i>row</i>	Row number.
<i>column</i>	Column number.
<i>position</i>	Border to check: 0 - Left, 1 - Right, 2 - Top, 3 - Bottom

Return Value

color, style and width of a cell border

Applies to

[Grid](#)

GetCellControl (Grid)

Syntax

GetCellControl *row, column*

Description

The GetCellControl method returns the cell control type (for example: push button, radio button, check box, combo box, etc.).

Parameters

The following table describes the parameters of the GetCellControl method:

Parameters	Description
<i>row</i>	Row number.
<i>column</i>	Column number.

Return Value

The control type for the cell

Applies to

[Grid](#)

GetCellEnabled (Grid)

Syntax

GetCellEnabled *row, column*

Description

The GetCellEnabled method returns whether or not a cell is enabled.

Parameters

The following table describes the parameters of the GetCellEnabled method:

Parameters	Description
<i>row</i>	Row number.
<i>column</i>	Column number

Return Value

True if the cell is enabled

Applies to

[Grid](#)

GetCellFloat (Grid)

Syntax

GetCellFloat *row, column*

Description

The GetCellFloat method returns whether or not a cell is allowed to overlay cells to the right of it.

Parameters

The following table describes the parameters of the GetCellFloat method:

Parameters	Description
<i>row</i>	Row number
<i>column</i>	Column number

Return Value

True if the cell is allowed to float over following cells

Applies to

[Grid](#)

GetCellFont (Grid)

Syntax

GetCellFont *row, column*

Description

The GetCellFont method returns the font type, size and style of a cell.

Parameters

The following table describes the parameters of the GetCellFont method:

Parameters	Description
<i>row</i>	Row number.
<i>column</i>	Column number.

Return Value

font name, size and style of the cell

Applies to

[Grid](#)

GetCellForeColor (Grid)

Syntax

GetCellForeColor *row, column*

Description

The GetCellForeColor method returns the foreground color of a cell.

Parameters

The following table describes the parameters of the GetCellForeColor method:

Parameters	Description
<i>row</i>	Row number.
<i>column</i>	Column number.

Return Value

The foreground color of the cell

Applies to

[Grid](#)

GetCellJustify (Grid)

Syntax

GetCellJustify *row, column*

Description

The GetCellJustify method returns the cell justification.

Parameters

The following table describes the parameters of the GetCellJustify method:

Parameters	Description
<i>row</i>	Row number.
<i>column</i>	Column number.

Return Value

The current justification of the cell

Applies to

[Grid](#)

GetCellList (Grid)

Syntax

GetCellList *row, column*

Description

The GetCellList method returns a carriage return separated list of options for a combo box or the text for a button control.

Parameters

The following table describes the parameters of the GetCellList method:

Parameters	Description
<i>row</i>	Row number.
<i>column</i>	Column number.

Return Value

The list of items for a combo box cell, or the caption for a button cell.

Applies to

[Grid](#)

GetCellReadOnly (Grid)

Syntax

GetCellReadOnly *row, column*

Description

The GetCellReadOnly returns whether or not a cell is read-only.

Parameters

The following table describes the parameters of the GetCellReadOnly method:

Parameters	Description
<i>row</i>	Row number.
<i>column</i>	Column number.

Return Value

True if the cell is read only

Applies to

[Grid](#)

GetCellValue (Grid)

Syntax

GetCellValue *row, column*

Description

The GetCellValue method returns the text from a specified cell.

Parameters

The following table describes the parameters of the GetCellValue method:

Parameters	Description
<i>row</i>	Row number
<i>column</i>	Column number

Return Value

The value of the cell

Applies to

[Grid](#)

GetCheck (CheckListBox)

Syntax

GetCheck *line*

Description

This method returns the checked status of the given line in a checked list box.

Parameters

The following table describes the parameters of the GetCheck method:

Parameters	Description
<i>line</i>	The line number to check.

Return Value

1 (True) if the specified line was checked, otherwise 0 (false).

Applies to

[CheckListBox](#)

Example

See if Wednesday (line 3) is checked in our days check list box

```
If MeetingDays.days.GetCheck(3) Then MessageBox "Wednesday is  
selected"
```

GetColWidth (Grid)

Syntax

GetColWidth *col*

Description

The GetColWidth method returns the width of a specified column.

Parameters

The following table describes the parameters of the GetColWidth method:

Parameters	Description
<i>col</i>	Column number.

Return Value

width of the specified column

Applies to

[Grid](#)

GetColumn (ListView)

Syntax

GetColumn *index, colinfo_var*

Description

The GetColumn method gets information about a column and returns True if successful. Set the mask field of colinfo_var to specify what data to receive.

Parameters

The following table describes the parameters of the GetColumn method:

Parameters	Description
<i>index</i>	Index of column to get.
<i>colinfo_var</i>	Name of variable to receive ListView column information record.

Values for colinfo_var

The following table describes the column information record:

Value	Description
<i>1</i>	Mask: 1 - format, 2 - width, 4 - text, 8 - sub item
<i>2</i>	Format: 0 - Left, 1 - Right, 2 - Center
<i>3</i>	Width.
<i>4</i>	Text.
<i>5</i>	Max length of text (when getting text).
<i>6</i>	Sub item.

Return Value

Returns true if successful

Applies to

[ListView](#)

GetColumnWidth (ListView)

Syntax

GetColumnWidth *column*

Description

The GetColumnWidth method returns the width of a column.

Parameters

The following table describes the parameters of the GetColumnWidth method:

Parameters	Description
<i>column</i>	Index of column.

Return Value

The width of the column

Applies to

[ListView](#)

GetCountPerPage (ListView)

Syntax

GetCountPerPage

Description

The **GetCountPerPage** method returns the number of fully visible items when the control is in list or report mode. If the current view is icon or small icon view, the return value is the total number of items in the list view control. It returns the number of fully-visible items if successful.

Parameters

None

Return Value

Number of fully visible items in the list view

Applies to

[ListView](#)

GetCurrentCell (Grid)

Syntax

GetCurrentCell *row_var, column_var*

Description

The GetCurrentCell method returns the row and column numbers of the current cell. This method returns True if the current cell can be determined.

Parameters

The following table describes the parameters of the GetCurrentCell method:

Parameters	Description
<i>row_var</i>	The variable to store the row number.
<i>column_var</i>	The variable to store the column number.

Return Value

True if the current cell can be determined

Applies to

[Grid](#)

GetCurrentText (Grid)

Syntax

`GetCurrentText`

Description

The `GetCurrentText` method returns the text from the current cell before it has been entered into the grid.

You can use this in conjunction with the `ValidateCell` event to validate the text that the user entered.

Parameters

None

Return Value

Text from the current cell during editing

Applies to

[Grid](#)

GetItem (Header)

Syntax

GetItem *index*, *var_name*

Description

The GetItem method returns the details of a header item and returns a nonzero value if successful.

Parameters

The following table describes the parameters of the GetItem method:

Parameters	Description
<i>index</i>	Zero based index for item to get
<i>var_name</i>	Variable to set with Header item details record

Values for var_name

The Header Item details record consists of the following fields.

Value	Description
<i>1</i>	The mask field specifies which of the other fields in the record contains valid data. It is a combination of the following values. 1 - width field is valid, 2 - text field is valid, 4 - format field is valid, 8 - user data is valid
<i>2</i>	Width of item.
<i>3</i>	Text of item.
<i>4</i>	Reserved. Must be "" or 0.
<i>5</i>	Text max, number, maximum size the text can be (for retrieving text from an item).

Value	Description
6	Format flags for item. This specifies the format of the data. Justification (check the return value & 3): 0 - Left justified, 1 - Right justified, 2 - Centered. Other flags: 16384 - Item contains text (required when setting/inserting an item)
7	User data. User defined number

Return Value

True if information was returned successfully

Applies to

[Header](#)

GetItem (ListView)

Syntax

GetItem *lvitem_var*, *ok*

Description

The GetItem method sets the item number, mask and state mask fields of *lvitem_var* to specify the item and which details you are interested in. It returns True if successful.

Parameters

The following table describes the parameters of the GetItem method:

Parameters	Description
<i>lvitem_var</i>	Variable to hold the List view item record for the requested item. This must be initialized first to specify what values to use. See below.
<i>ok</i>	Returns true if successful

Values for lvitem_var

The following fields describe the list view item record:

Value	Description
<i>1</i>	mask. Specifies which other fields are valid. A combination of: 1 - Text field is valid, 2 - Image id is valid, 4 - user data is valid, 8 - state is valid
<i>2</i>	index of item
<i>3</i>	index of sub item
<i>4</i>	state. Current state bits: 1 - item has the focus, 2 - item is selected, 4 - item is displayed as if it has been cut, 8 - item is highlighted as a drag and drop target

Value	Description
5	state mask. Specified the state bits to get/set
6	text
7	maximum size of text (use when getting text)
8	image id
9	data. User defined data

Applies to

[ListView](#)

GetItem (TreeView)

Syntax

GetItem *item_var*

Description

The GetItem method returns details about an item of the tree view and returns True if successful.

Parameters

The following table describes the parameters of the GetItem method:

Parameters	Description
<i>item_var</i>	The variable to contain a Treeview item details record. See below.

Values for item_var

The following table describes the Treeview item details record:

Value	Description
1	The mask field specifies which of the other fields in the record contains valid data. It is a combination of the following values. 1 - Text field is valid, 2 - Image field is valid, 4 - Param field is valid, 8 - State field is valid, 16 - Item id field is valid, 32 - Selected image field is valid, 64 - Children count field is valid.
2	The item id of the a line in the treeview

Value	Description
3	The state of bits to be set for the item. The bits which are valid are specified in the state mask and can be a combination of the following values: 2 - Item is selected, 4 - Item is cut, 8 - Item is drop highlighted, 16 - Item is bold, 32 - Item is expanded, 64 - Item has been expanded at least once, 128 - Item has been partially expanded.
4	The state of bits to be retrieved for the item. The bits which are valid are specified in the state mask and can be a combination of the following values: 2 - Item is selected, 4 - Item is cut, 8 - Item is drop highlighted, 16 - Item is bold, 32 - Item is expanded, 64 - Item has been expanded at least once, 128 - Item has been partially expanded.
5	Text entry for the item.
6	Maximum size of the text can (for retrieving text from an item).
7	Image number, normal image index.
8	Selected image number, selected image index.
9	Count of children for an item
10	User specified number associated with this item.

Return Value

True if successful.

Applies to

[TreeView](#)

GetItemData (ComboBox)

Syntax

GetItemData *line_number*

Description

Returns the number associated with the line of the combo box with SetItemData.

Parameters

The following table describes the parameters of the GetItemData method:

Parameters	Description
<i>line_number</i>	The number of the line to retrieve the data from

Return Value

The number stored against the specified line.

Applies to

[ComboBox](#)

GetItemData (ListBox)

Syntax

GetItemData *line_number*

Description

Returns the number associated with the line of the list box with SetItemData.

Parameters

The following table describes the parameters of the GetItemData method:

Parameters	Description
<i>line_number</i>	Line number to return the data from

Return Value

Number stored against this line of the list

Applies to

[ListBox](#)

GetItemData (TreeView)

Syntax

GetItemData *item_id*

Description

The GetItemData method returns a number stored against an item.

Parameters

The following table describes the parameters of the GetItemData method:

Parameters	Description
<i>item_id</i>	Item number of item to retrieve data from

Return Value

The number previously stored against this item

Applies to

[TreeView](#)

GetItemImage (TreeView)

Syntax

GetItemImage *item_id*

Description

The GetItemImage method returns the current images used for an item.

Parameters

The following table describes the parameters of the GetItemImage method:

Parameters	Description
<i>item_id</i>	Item number

Return Value

Number of the image for the specified item

Applies to

[TreeView](#)

GetItemPosition (ListView)

Syntax

GetItemPosition *item_no, x_var, y_var*

Description

The GetItemPosition method gets the top left position of an item and returns True if successful.

Parameters

The following table describes the parameters of the GetItemPosition method:

Parameters	Description
<i>item_no</i>	Index of item to retrieve position from.
<i>x_var</i>	Variable to place x position.
<i>y_var</i>	Variable to place y position.

Return Value

True if successful

Applies to

[ListView](#)

GetItemState (ListView)

Syntax

GetItemState *item_no*, *mask*

Description

The GetItemState method returns the state of an item.

Parameters

The following table describes the parameters of the GetItemState method:

Parameters	Description
<i>item_no</i>	Index of item to get state for
<i>mask</i>	mask of state bits to test. See below

Values for mask

The state bits used in the mask parameter and the return value are:

Value	Description
<i>1</i>	item has the focus
<i>2</i>	item is selected
<i>4</i>	item is displayed as if it has been cut
<i>8</i>	item is highlighted as a drag and drop target

Return Value

The state bits specified in the mask parameter for the item.

Applies to

[ListView](#)

GetItemState (TreeView)

Syntax

GetItemState *item_id*, *state_mask*

Description

The GetItemState method returns the items state.

Parameters

The following table describes the parameters of the GetItemState method:

Parameters	Description
<i>item_id</i>	Item number
<i>state_mask</i>	State bits to check. See below

Values for state_mask

The state bits are follows. Combine them using the | symbol or addition.

Value	Description
2	Item is selected
4	Item is cut
8	Item is drop highlighted
16	Item is bold
32	Item is expanded
64	Item has been expanded at least once
128	Item has been partially expanded.

Return Value

The state bits of the item specified by the state_mask

Applies to

[TreeView](#)

GetItemText (Header)

Syntax

GetItemText *index*

Description

The GetItem Text method retrieves the text from a header item.

Parameters

The following table describes the parameters of the GetItemText method:

Parameters	Description
<i>index</i>	Zero based index for item.

Return Value

The text of the specified item

Applies to

[Header](#)

GetItemText (ListView)

Syntax

GetItemText *item_no*, [*subitem_no*]

Description

The GetItemText method returns the text for item or "" if unsuccessful.

Parameters

The following table describes the parameters of the GetItemText method:

Parameters	Description
<i>item_no</i>	Index for item.
<i>subitem_no</i>	Index for subitem (0 means item label).

Return Value

The text for the item. "" is returned if the text cannot be retrieved.

Applies to

[ListView](#)

GetItemText (TabControl)

Syntax

GetItemText *index*

Description

The GetItemText method returns the label text from a tab or "" if the index is invalid.

Parameters

The following table describes the parameters of the GetItemText method:

Parameters	Description
<i>index</i>	Index of the tab.

Return Value

Text from the tab or ""

Applies to

[TabControl](#)

GetItemText (TreeView)

Syntax

GetItemText *item_id*, [*max_length*]

Description

The GetItemText method returns the text for an item.

Parameters

The following table describes the parameters of the GetItemText method:

Parameters	Description
<i>item_id</i>	ID of item to retrieve the text.
<i>max_length</i>	Maximum length of item text (default is 255 characters).

Return Value

The text for an item

Applies to

[TreeView](#)

GetItemWidth (Header)

Syntax

GetItemWidth *index*

Description

The GetItemWidth method retrieves the width of a header item or returns zero if an error occurs.

Parameters

The following table describes the parameters of the GetItemWidth method:

Parameters	Description
<i>index</i>	Zero based index for item.

Return Value

The width of the item or 0 if an error occurs.

Applies to

[Header](#)

GetNextItem (ListView)

Syntax

GetNextItem *start, flags*

Description

The GetNextItem method returns the next item from the listview and returns the index of the item or -1 if no item was found.

Parameters

The following table describes the parameters of the GetNextItem method:

Parameters	Description
<i>start</i>	Item to look for next item from or -1 to get first item.
<i>flags</i>	Direction and state to search for. See below.

Values for flags

The flags are made up of a search direction and one or more state flags.

Value	Description
256	Search: Above current item
512	Search: Below the current item
1024	Search: To left of current item
2048	Search: To right of current item
0	Item state: Specifying zero for the state flags matches all items
1	Item state: Item with focus
2	Item state: Selected
4	Item state: Cut
8	Item state: Drop highlighted

Return Value

The index of the item or -1 if no item was found.

Applies to

[ListView](#)

GetNextItem (TreeView)

Syntax

GetNextItem *relationship, item_id*

Description

The GetNextItem method returns an item with the specified relationship to item_id.

Parameters

The following table describes the parameters of the GetNextItem method:

Parameters	Description
<i>relationship</i>	Relationship to item_id of item to return. See below.
<i>item_id</i>	Id of item

Values for relationship

Value	Description
0	Root item of the tree (set item_id to 0).
1	Next sibling item.
2	Previous sibling item
4	First child of the item.
5	First visible item (set item_id to 0).
6	Next visible item after this item.
7	Previous visible item.
8	Item with the drag and drop highlight.
9	Item with the current selection.

Return Value

Item id of the next item

Applies to

[TreeView](#)

GetOrigin (ListView)

Syntax

GetOrigin *x_var*, *y_var*

Description

The GetOrigin method gets the co-ordinates of the view origin and returns True if successful or False if view is in list or report mode

Parameters

The following table describes the parameters of the GetOrigin method:

Parameters	Description
<i>x_var</i>	Name of variable to receive x co-ordinate.
<i>y_var</i>	Name of variable to receive y co-ordinate.

Return Value

True if successful or False if view is in list or report mode

Applies to

[ListView](#)

GetRangeAttribute (Grid)

Syntax

GetRangeAttribute *from_row, from_col, to_row, to_col, attribute*

Description

The GetRangeAttribute method returns a dynamic array of values of the specified attribute in the range of cells.

Parameters

The following table describes the parameters of the GetRangeAttribute method:

Parameters	Description
<i>from_row</i>	Start row.
<i>from_col</i>	Start column.
<i>to_row</i>	End row.
<i>to_col</i>	End column.
<i>attribute</i>	Attribute to return. See SetCellAttribute method for currently supported attributes and their values value Value for the attribute.

Return Value

Dynamic array of the values for the specified attribute

Applies to

[Grid](#)

GetRangeAutoSize (Grid)

Syntax

GetRangeAutoSize *from_row, from_col, to_row, to_col*

Description

The GetRangeAutoSize method returns a dynamic array containing whether or not the height of cells within the specified range is able to grow to fit the text.

Parameters

The following table describes the parameters of the GetRangeAutoSize method:

Parameters	Description
<i>from_row</i>	Start row.
<i>from_col</i>	Start column.
<i>to_row</i>	End row.
<i>to_col</i>	End column.

Return Value

Dynamic array of true or false values (1 or 0).

Applies to

[Grid](#)

GetRangeBackColor (Grid)

Syntax

GetRangeBackColor *from_row, from_col, to_row, to_col*

Description

The GetRangeBackColor method returns a dynamic array containing the background color for cells within the specified range.

Parameters

The following table describes the parameters of the GetRangeBackColor method:

Parameters	Description
<i>from_row</i>	Start row.
<i>from_col</i>	Start column.
<i>to_row</i>	End row.
<i>to_col</i>	End column.

Return Value

Dynamic array of the background colors

Applies to

[Grid](#)

GetRangeBorder (Grid)

Syntax

GetRangeBorder *from_row, from_col, to_row, to_col, position*

Description

The GetRangeBorder method returns the color, style, and width of the border for cells within the specified range.

Parameters

The following table describes the parameters of the GetRangeBorder method:

Parameters	Description
<i>from_row</i>	Start row
<i>from_col</i>	Start column
<i>to_row</i>	End row
<i>to_col</i>	End column.
<i>position</i>	Border to check: 0 - Left, 1 - Right, 2 - Top, 3 - Bottom

Return Value

Dynamic array of border styles

Applies to

[Grid](#)

GetRangeControl (Grid)

Syntax

GetRangeControl *from_row, from_col, to_row, to_col*

Description

The **GetRangeControl** method returns a dynamic array containing the cell control type (for example: push button, radio button, check box, combo box, etc.) for cells within the specified range.

Parameters

The following table describes the parameters of the **GetRangeControl** method:

Parameters	Description
<i>from_row</i>	Start row.
<i>from_col</i>	Start column.
<i>to_row</i>	End row.
<i>to_col</i>	End column.

Return Value

Dynamic array of the controls in the range

Applies to

[Grid](#)

GetRangeEnabled (Grid)

Syntax

GetRangeEnabled *from_row, from_col, to_row, to_col*

Description

The GetRangeEnabled method returns a dynamic array containing whether or not cells within the specified range are enabled.

Parameters

The following table describes the parameters of the GetRangeEnabled method:

Parameters	Description
<i>from_row</i>	Start row.
<i>from_col</i>	Start column.
<i>to_row</i>	End row.
<i>to_col</i>	End column.

Return Value

Dynamic array of containing true (1) or false (0) for each cell in the range

Applies to

[Grid](#)

GetRangeFloat (Grid)

Syntax

GetRangeFloat *from_row, from_col, to_row, to_col*

Description

The GetRangeFloat method returns a dynamic array describing whether or not cells within the specified range are allowed to overlay the cell to the right of it.

Parameters

The following table describes the parameters of the GetRangeFloat method:

Parameters	Description
<i>from_row</i>	Start row.
<i>from_col</i>	Start column.
<i>to_row</i>	End row.
<i>to_col</i>	End column.

Return Value

Dynamic array specify which cells allow floating

Applies to

[Grid](#)

GetRangeForeColor (Grid)

Syntax

GetRangeForeColor *from_row, from_col, to_row, to_col*

Description

The GetRangeForeColor method returns a dynamic array containing the foreground colors for cells within the specified range.

Parameters

The following table describes the parameters of the GetRangeForeColor method:

Parameters	Description
<i>from_row</i>	Start row.
<i>from_col</i>	Start column.
<i>to_row</i>	End row.
<i>to_col</i>	End column.

Return Value

Dynamic array of the foreground colors for the range

Applies to

Grid

GetRangeJustify (Grid)

Syntax

GetRangeJustify *from_row, from_col, to_row, to_col*

Description

The GetRangeJustify method returns a dynamic array containing the justification for cells within the specified range.

Parameters

The following table describes the parameters of the GetRangeJustify method:

Parameters	Description
<i>from_row</i>	Start row.
<i>from_col</i>	Start column.
<i>to_row</i>	End row.
<i>to_col</i>	End column.

Return Value

Dynamic array of justifications in the specifed range of cells

Applies to

[Grid](#)

GetRangeList (Grid)

Syntax

GetRangeList *from_row, from_col, to_row, to_col*

Description

The GetRangeList method returns a dynamic array containing a carriage return separated list of options for a combo box or the text for a button control.

Parameters

The following table describes the parameters of the GetRangeList method:

Parameters	Description
<i>from_row</i>	Start row.
<i>from_col</i>	Start column.
<i>to_row</i>	End row.
<i>to_col</i>	End column.

Return Value

Dynamic array containings lists from the specified range of cells

Applies to

[Grid](#)

GetRangeReadOnly (Grid)

Syntax

GetRangeReadOnly *from_row, from_col, to_row, to_col*

Description

The **GetRangeReadOnly** method returns a dynamic array containing whether or not cells within the specified range is read-only.

Parameters

The following table describes the parameters of the **GetRangeReadOnly** method:

Parameters	Description
<i>from_row</i>	Start row.
<i>from_col</i>	Start column.
<i>to_row</i>	End row.
<i>to_col</i>	End column.

Return Value

Dynamic array of true (1) or false(0) read-only values for the specified range of cells

Applies to

[Grid](#)

GetRangeValue (Grid)

Syntax

GetRangeValue *from_row, from_col, to_row, to_col*

Description

The GetRangeValue method returns a dynamic array containing the values in cells within the specified range.

Parameters

The following table describes the parameters of the GetRangeValue method:

Parameters	Description
<i>from_row</i>	Start row.
<i>from_col</i>	Start column.
<i>to_row</i>	End row.
<i>to_col</i>	End column.

Return Value

Dynamic array of the values for the specified range

Applies to

[Grid](#)

GetRowHeight (Grid)

Syntax

GetRowHeight *row*

Description

The GetRowHeight method returns the height of a row.

Parameters

The following table describes the parameters of the GetRowHeight method:

Parameters	Description
<i>row</i>	The row to return the height of.

Return Value

Height of the specified row

Applies to

[Grid](#)

GetSel (ListBox)

Syntax

GetSel *line_number*

Description

Returns a value greater than 0 if the line is selected in a multiple selection listbox.

Parameters

The following table describes the parameters of the GetSel method:

Parameters	Description
<i>line_number</i>	Line number to check for a selection

Return Value

A number greater than zero if the line is selected

Applies to

[CheckListBox](#), [ListBox](#)

GetSelectedCount (ListView)

Syntax

GetSelectedCount

Description

The GetSelectedCount method returns the number of selected items.

Parameters

None

Return Value

The number of selected items

Applies to

[ListView](#)

GetTick (TrackBar)

Syntax

GetTick *index*

Description

Returns the position of the tick for the index or -1 if the index is out of range

Parameters

The following table describes the parameters of the GetTick method:

Parameters	Description
<i>index</i>	Index of the tick. The first tick is at index 0.

Return Value

Position of the tick for the index or -1.

Applies to

[TrackBar](#)

GetTopIndex (ListView)

Syntax

GetTopIndex

Description

The GetTopIndex method returns the top index of an item when in list or report mode

Parameters

None

Return Value

Top index of an item when in list or report mode

Applies to

[ListView](#)

HideCols (Grid)

Syntax

HideCols *hide_from*, [*number*], [*flag*]

Description

The HideCols method hides or displays the columns you specify.

Parameters

The following table describes the parameters of the HideCols method:

Parameters	Description
<i>hide_from</i>	The first column to hide.
<i>number</i>	The number of columns to hide (default is 1).
<i>flag</i>	Flag must be one of the following: True - Hide columns (the default), False - Show columns.

Applies to

[Grid](#)

HideRows (Grid)

Syntax

HideRows *hide_from*, [*number*], [*flag*]

Description

The HideRows method hides or shows the rows you specify.

Parameters

The following table describes the parameters of the HideRows method:

Parameters	Description
<i>hide_from</i>	The first row hide.
<i>number</i>	The number of rows to hide (default is 1).
<i>flag</i>	Flag must be one of the following: True - Hide rows (the default), False - Show rows.

Applies to

[Grid](#)

InitStorage (ComboBox)

Syntax

InitStorage *number_of_items*, *number_of_bytes*

Description

Increases the amount of storage used for the list part of the combobox.

Use this if a lot of items are going to be added and an estimate of the size can be made as it will speed up loading of the list box.

Returns the total number of items that space has been pre-allocated for with InitStorage.

Note: These sizes do not fix the number of items that can be added to the list box.

Parameters

The following table describes the parameters of the InitStorage method:

Parameters	Description
<i>number_of_items</i>	Number of items to initialise storage for
<i>number_of_bytes</i>	Number of bytes to initialise storage for

Return Value

The total number of items that space has been pre-allocated for with InitStorage.

Applies to

[ComboBox](#)

InitStorage (ListBox)

Syntax

InitStorage *number_of_items*, *number_of_bytes*

Description

Increases the amount of storage used for the list .

Use this if a lot of items are going to be added and an estimate of the size can be made as it will speed up loading of the list box.

Returns the total number of items that space has been pre-allocated for with InitStorage.

Note: These sizes do not fix the number of items that can be added to the list box.

Parameters

The following table describes the parameters of the InitStorage method:

Parameters	Description
<i>number_of_items</i>	Number of items to initialise storage for
<i>number_of_bytes</i>	Number of bytes to initialise storage for

Return Value

The total number of items that space has been pre-allocated for with InitStorage.

Applies to

[CheckListBox](#), [ListBox](#)

InsPosFromPoint (ListBox)

Syntax

InsPosFromPoint *x*, *y*

Description

Returns the index of the position to insert a new line at the specified pixel position over the listbox. This returns 0 if the position is over an empty area at the end of a list box or over half way down the last item in the list. The result from this method can be used directly in the InsertLine method to insert a new line at the returned position.

Parameters

The following table describes the parameters of the InsPosFromPoint method:

Parameters	Description
<i>x</i>	x pixel position
<i>y</i>	y pixel position

Return Value

Nearest line number or 0 if the position is over an empty section of the list box or over half way down the last item in the list.

Applies to

[CheckListBox](#), [ListBox](#)

InsertCols (Grid)

Syntax

InsertCols *insert_at*, [*number*]

Description

The InsertCols method inserts into a grid the number of columns you specify.

Parameters

The following table describes the parameters of the InsertCols method:

Parameters	Description
<i>insert_at</i>	The column to insert new columns before.
<i>number</i>	The number of columns to insert (default 1).

Applies to

[Grid](#)

InsertColumn (ListView)

Syntax

InsertColumn *column*, *colinfo*

Description

The InsertColumn method insert a new column into the listview and returns the index of the new column or -1.

Note: Left most (first) column is always left justified regardless of the format flags.

Parameters

The following table describes the parameters of the InsertColumn method:

Parameters	Description
<i>column</i>	Column to insert.
<i>colinfo</i>	Listview column information record.

Values for colinfo

The following table describes the column information record:

Value	Description
<i>1</i>	Mask: 1 - format, 2 - width, 4 - text, 8 - sub item
<i>2</i>	Format: 0 - Left, 1 - Right, 2 - Center
<i>3</i>	Width.
<i>4</i>	Text.
<i>5</i>	Max length of text (when getting text).
<i>6</i>	Sub item.

Return Value

Index of the new column or -1

Applies to

[ListView](#)

InsertItem (Header)

Syntax

InsertItem *index*, *hd_item*

Description

The InsertItem method inserts a new item into the header and returns the index value of the insertion location or -1 if there is an error.

Parameters

The following table describes the parameters of the InsertItem method:

Parameters	Description
<i>index</i>	Zero based index to insert. If index is greater than the last current index, the item is inserted at the end of the header.
<i>hd_item</i>	Header item details record.

Values for hd_item

Header item detail record consists of the following fields

Value	Description
<i>1</i>	The mask field specifies which of the other fields in the record contains valid data. It is a combination of the following values. 1 - width field is valid, 2 - text field is valid, 4 - format field is valid, 8 - user data is valid
<i>2</i>	Width of item.
<i>3</i>	Text of item.
<i>4</i>	Reserved. Must be "" or 0.
<i>5</i>	Text max, number, maximum size the text can be (for retrieving text from an item).

Value	Description
6	Format flags for item. This specifies the format of the data. Justification (check the return value & 3): 0 - Left justified, 1 - Right justified, 2 - Centered. Other flags: 16384 - Item contains text (required when setting/inserting an item)
7	User data. User defined number

Return Value

Index of insert location or -1 if an error occurred.

Applies to

[Header](#)

InsertItem (ListView)

Syntax

InsertItem *lvitem*

Description

The InsertItem method sets the item number field of lvitem to specify the location to insert. Sub-item can not be inserted to the subitem field must be 0 or "". It returns the index of the new item or -1.

Parameters

The following table describes the parameters of the InsertItem method:

Parameters	Description
<i>lvitem</i>	Details of item to be inserted in a Listview item details record

Values for lvitem

The following fields describe the list view item record:

Value	Description
1	mask. Specifies which other fields are valid. A combination of: 1 - Text field is valid, 2 - Image id is valid, 4 - user data is valid, 8 - state is valid
2	index of item
3	index of sub item
4	state. Current state bits: 1 - item has the focus, 2 - item is selected, 4 - item is displayed as if it has been cut, 8 - item is highlighted as a drag and drop target
5	state mask. Specified the state bits to get/set

Value	Description
6	text
7	maximum size of text (use when getting text)
8	image id
9	data. User defined data

Return Value

Index of the new item or -1.

Applies to

[ListView](#)

Example

Inserting an new item into a list view

```
lv_item = ""

Rep lv_item, 1, 11
Rep lv_item, 2,0      ;* Insert at position 0
Rep lv_item, 3,0      ;* Column position - always 0 for an insertion
Rep lv_item, 4,0      ;* State
Rep lv_item, 5,0      ;* State mask
Rep lv_item, 6,"Text for icon"
Rep lv_item, 7,30     ;* Maximum size of text
Rep lv_item, 8,0      ;* Icon to use from image list
Rep lv_item, 9,""     ;* User defined data

* Then insert it into the list view
MyDlg.MyListView.InsertItem Item

* If you have multiple columns you need to put the text in them
separetly
MyDlg.MyListView.SetItemText 0, 1, "First column of item 0"
MyDlg.MyListView.SetItemText 0, 2, "Second column of item 0"
```


InsertItem (TreeView)

Syntax

InsertItem *tv_item*, [*parent*], [*position*]

Description

The InsertItem method inserts a new item into a treeview with its details specified by the text, image, selected image, state and IParam fields of the Treeview item details record.

InsertItem returns the item ID of the inserted item or 0 if unsuccessful. Note: The tree is not redrawn until the redraw property is set to true

Parameters

The following table describes the parameters of the InsertItem method:

Parameters	Description
<i>tv_item</i>	Tree view item details record. See below.
<i>parent</i>	The item ID of the parent for to insert the item. To insert at the root of the tree, use 0 or ROOT. This is the default.
<i>position</i>	The position to insert the item. This is the item id after which to insert the new item or one of the following values: "FIRST" - Insert at the beginning, "LAST" or "" - Insert at the end (the default), "SORT" - Insert in alphabetical order.

Values for tv_item

The following table describes the Treeview item details record:

Value	Description
1	The mask field specifies which of the other fields in the record contains valid data. It is a combination of the following values. 1 - Text field is valid, 2 - Image field is valid, 4 - Param field is valid, 8 - State field is valid, 16 - Item id field is valid, 32 - Selected image field is valid, 64 - Children count field is valid.
2	The item id of the a line in the treeview
3	The state of bits to be set for the item. The bits which are valid are specified in the state mask and can be a combination of the following values: 2 - Item is selected, 4 - Item is cut, 8 - Item is drop highlighted, 16 - Item is bold, 32 - Item is expanded, 64 - Item has been expanded at least once, 128 - Item has been partially expanded.
4	The state of bits to be retrieved for the item. The bits which are valid are specified in the state mask and can be a combination of the following values: 2 - Item is selected, 4 - Item is cut, 8 - Item is drop highlighted, 16 - Item is bold, 32 - Item is expanded, 64 - Item has been expanded at least once, 128 - Item has been partially expanded.
5	Text entry for the item.
6	Maximum size of the text can (for retrieving text from an item).
7	Image number, normal image index.
8	Selected image number, selected image index.
9	Count of children for an item
10	User specified number associated with this item.

Return Value

Item ID of the inserted item or 0 if unsuccessful.

Applies to

[TreeView](#)

InsertItemText (Header)

Syntax

InsertItemText *index, text, [width]*

Description

The InsertItemText method inserts a new header item with the specified text and width and returns the item index where the text was inserted.

Parameters

The following table describes the parameters of the InsertItemText method:

Parameters	Description
<i>index</i>	Zero based index for item
<i>text</i>	Text to insert.
<i>width</i>	Width for item (default is 64)

Return Value

Index where the item was inserted or -1

Applies to

[Header](#)

InsertItemText (TabControl)

Syntax

InsertItemText *index, text*

Description

The InsertItemText method inserts a new tab with the specified label text and returns the index of the new tab if successful, otherwise -1.

Parameters

The following table describes the parameters of the InsertItemText method:

Parameters	Description
<i>index</i>	Index for new tab.
<i>text</i>	Text for the tab.

Return Value

Index of the new tab is successful, otherwise -1

Applies to

[TabControl](#)

InsertLine (ComboBox)

Syntax

InsertLine *before_line*, *text*

Description

Inserts a line before the specified line number and returns the line number of the new line.

Parameters

The following table describes the parameters of the InsertLine method:

Parameters	Description
<i>before_line</i>	The number of the line to insert before
<i>text</i>	The text to insert

Return Value

The line number of the new line.

Applies to

[ComboBox](#)

InsertLine (ListBox)

Syntax

InsertLine *before_line*, *text*

Description

Inserts a line before the specified line number and returns the line number of the new line.

Parameters

The following table describes the parameters of the InsertLine method:

Parameters	Description
<i>before_line</i>	Line number to insert the text before
<i>text</i>	Line text to insert

Return Value

The line number of the new line

Applies to

[CheckListBox](#), [ListBox](#)

InsertLine (TreeView)

Syntax

InsertLine *text*, [*image*], [*sel_image*], [*parent*], [*postion*]

Description

The InsertLine method inserts a new item into a treeview and returns the item ID of the inserted item or 0 if unsuccessful.

Parameters

The following table describes the parameters of the InsertLine method:

Parameters	Description
<i>text</i>	Text for the new item.
<i>image</i>	Index number for normal image.
<i>sel_image</i>	Index number for selected image.
<i>parent</i>	Item ID of the parent for to insert the item under or 0 or ROOT (the default) for the root of the tree. position The position to insert the item. This is the item id after which to insert the new item or one of the following val-ues: • FIRST - Insert at the beginning. • LAST or "" - Insert at the end (the default). • SORT - Insert in alphabetical order.
<i>postion</i>	The position to insert the item. This is the item id after which to insert the new item or one of the following values: "FIRST" - Insert at the beginning, "LAST" or "" - Insert at the end (the default), "SORT" - Insert in alphabetical order.

Return Value

item ID of the inserted item or 0 if unsuccessful.

Applies to

[TreeView](#)

InsertRows (Grid)

Syntax

InsertRows *insert_at*, [*number*]

Description

The InsertRows method inserts the number of rows you specify into a grid.

Parameters

The following table describes the parameters of the InsertRows method:

Parameters	Description
<i>insert_at</i>	The row to insert new rows before.
<i>number</i>	The number of rows to insert (default is 1)

Applies to

[Grid](#)

ItemFromPoint (ListBox)

Syntax

ItemFromPoint *x*, *y*

Description

Returns the index of the nearest line to the specified pixel position over the listbox

Parameters

The following table describes the parameters of the ItemFromPoint method:

Parameters	Description
<i>x</i>	x pixel position
<i>y</i>	y pixel position

Return Value

Line number of the nearest line to the specified position

Applies to

[CheckListBox](#), [ListBox](#)

ItemFromPoint (TreeView)

Syntax

ItemFromPoint *x*, *y*

Description

This returns the `item_id` of the item nearest to the specified pixel position over the control or 0 if there is no item at the specified position

Parameters

The following table describes the parameters of the ItemFromPoint method:

Parameters	Description
<i>x</i>	x co-ordinate in pixels
<i>y</i>	y co-ordinate in pixels

Return Value

Item id nearest to the specified pixel position over the control or 0 if there is no item at the specified position.

Applies to

[TreeView](#)

Version

4.1.1 Original

Line (ComboBox)

Syntax

Line *line_number*

Description

Returns the text of the specified line or "" if the line number is incorrect.

Parameters

The following table describes the parameters of the Line method:

Parameters	Description
<i>line_number</i>	The line number to return the text from

Return Value

Text of the specified line or "" if the line number is incorrect.

Applies to

[ComboBox](#)

Line (EditText)

Syntax

Line line_number

Description

Returns the text of the line whose number has been specified.

Parameters

The following table describes the parameters of the Line method:

Parameters	Description
<i>line_number</i>	Line number to return text from

Return Value

Text of the line whose number has been specified.

Applies to

[EditText](#)

Line (ListBox)

Syntax

Line line_number

Description

Returns the text of the specified line or "" if the line number is incorrect.

Parameters

The following table describes the parameters of the Line method:

Parameters	Description
<i>line_number</i>	Number of line to retrieve the text from

Return Value

Text of the specified line or "" if the line number is incorrect.

Applies to

[CheckListBox](#), [ListBox](#)

LineCount (EditText)

Syntax

LineCount

Description

Returns the number of lines in the edit control.

Parameters

None

Return Value

The number of lines in the edit control.

Applies to

[EditText](#)

LineFromChar (EditText)

Syntax

LineFromChar *char_index*

Description

Returns the line number for the specified position.

Parameters

The following table describes the parameters of the LineFromChar method:

Parameters	Description
<i>char_index</i>	Character index

Return Value

The line number that contains the specified character position

Applies to

[EditText](#)

LineIndex (EditText)

Syntax

LineIndex *line_number*

Description

Returns the character index of the first character on the specified line

Parameters

The following table describes the parameters of the LineIndex method:

Parameters	Description
<i>line_number</i>	Line number

Return Value

Character index of the first character on the specified line

Applies to

[EditText](#)

LineLength (EditText)

Syntax

LineLength *char_index*

Description

Returns the length of the line which contains the specific character index.

Parameters

The following table describes the parameters of the LineLength method:

Parameters	Description
<i>char_index</i>	Character index

Return Value

The length of the line containing the specified character index

Applies to

[EditText](#)

LineScroll (EditText)

Syntax

LineScroll *columns, lines*

Description

Scrolls the edit control by the specified columns and lines. Returns TRUE if this message is sent to a multiline edit control.

Parameters

The following table describes the parameters of the LineScroll method:

Parameters	Description
<i>columns</i>	Number of columns to scroll
<i>lines</i>	Number of lines to scroll

Return Value

True if this message is sent to a multiline edit control.

Applies to

[EditText](#)

Open (Animate)

Syntax

Open *filename*

Description

The Open method opens an AVI file.

Parameters

The following table describes the parameters of the Open method:

Parameters	Description
<i>filename</i>	The file name to open

Return Value

True if file was opened successfully, otherwise false

Applies to

[Animate](#)

Paste

Syntax

Paste

Description

Pastes text from the clipboard to the control.

Parameters

None

Applies to

[ComboBox](#), [EditText](#), [Grid](#)

Version

4.0.2 Added to Grid control

Play (Animate)

Syntax

Play [*from_frame*], *to_frame*, *repeat_count*

Description

The Play method plays an AVI file.

Parameters

The following table describes the parameters of the Play method:

Parameters	Description
<i>from_frame</i>	Specifies the starting frame to play.
<i>to_frame</i>	Specifies the ending frame to play.
<i>repeat_count</i>	Specifies the number of times to repeat playing the AVI file

Applies to

[Animate](#)

ReplaceSel (EditText)

Syntax

ReplaceSel *text*, [*can_undo*]

Description

Replaces the current selection with the specified text. Set `can_undo` to true if the Undo method is allowed to undo the change. If there is no current selection the data is inserted at the caret position.

Parameters

The following table describes the parameters of the ReplaceSel method:

Parameters	Description
<i>text</i>	Text to replace the selection with
<i>can_undo</i>	Set to true if the Undo method is allowed to undo the change.

Applies to

[EditText](#)

ResizeColToFit (Grid)

Syntax

ResizeColToFit *from_col, to_col*

Description

The ResizeColToFit method resizes to fit the data the columns you specify.

Parameters

The following table describes the parameters of the ResizeColToFit method:

Parameters	Description
<i>from_col</i>	First row to resize.
<i>to_col</i>	Last row to resize.

Applies to

[Grid](#)

ResizeRowToFit (Grid)

Syntax

ResizeRowToFit *from_row, to_row*

Description

The ResizeRowToFit method resizes the rows you specify to fit the data.

Parameters

The following table describes the parameters of the ResizeRowToFit method:

Parameters	Description
<i>from_row</i>	The first row to resize.
<i>to_row</i>	The last row to resize

Applies to

[Grid](#)

RowCount (TabControl)

Syntax

RowCount

Description

The RowCount method returns the number of rows in the tab control.

Parameters

None

Return Value

Number of rows in the tab control

Applies to

[TabControl](#)

Scroll (EditText)

Syntax

Scroll *scroll_type*

Description

Scrolls the edit control a number of lines depending on *scroll_type*.

Parameters

The following table describes the parameters of the Scroll method:

Parameters	Description
<i>scroll_type</i>	Amount to scroll. See below

Values for scroll_type

Value	Description
<i>0</i>	Up one line
<i>1</i>	Down one line
<i>2</i>	Up one page
<i>3</i>	Down one page

Applies to

[EditText](#)

ScrollCaret (EditText)

Syntax

ScrollCaret

Description

Scrolls the edit control so the caret is visible.

Parameters

None

Applies to

[EditText](#)

ScrollTo (Graphic)

Syntax

ScrollTo *x*, *y*

Description

Scrolls the image to the specified position.

The image must have been defined with the WS_HSCROLL and WS_VSCROLL styles.

Parameters

The following table describes the parameters of the ScrollTo method:

Parameters	Description
<i>x</i>	Column position to scroll the image to in pixels
<i>y</i>	Row position to scroll the image to in pixels

Applies to

[Graphic](#)

ScrollToCell (Grid)

Syntax

ScrollToCell *row, col*

Description

The ScrollToCell method scrolls to display a cell you specify and returns True if the display was successful.

Parameters

The following table describes the parameters of the ScrollToCell method:

Parameters	Description
<i>row</i>	Row number.
<i>col</i>	Column number.

Applies to

[Grid](#)

Seek (Animate)

Syntax

Seek *frame*

Description

The Seek method moves to the specified frame.

Parameters

The following table describes the parameters of the Seek method:

Parameters	Description
<i>frame</i>	Specifies the frame to move to.

Applies to

[Animate](#)

SelCount (ListBox)

Syntax

SelCount

Description

Returns the number of lines that are selected.

Parameters

None

Return Value

Number of lines selected

Applies to

[CheckListBox](#), [ListBox](#)

SelLine (ComboBox)

Syntax

SelLine *start_line*, *text*

Description

Selects the line of text which start with the specified text and returns the line number selected. *start_line* specifies the line before the first line to be searched and can be 0 to search the entire list

Parameters

The following table describes the parameters of the SelLine method:

Parameters	Description
<i>start_line</i>	Specifies the line before the first line to be searched and can be 0 to search the entire list.
<i>text</i>	Start of line text to match against

Applies to

[ComboBox](#)

SelLine (ListBox)

Syntax

SelLine *start_line*, *text*

Description

Selects the line of text which start with the specified text and returns the line number selected.

Parameters

The following table describes the parameters of the SelLine method:

Parameters	Description
<i>start_line</i>	Specifies the line before the first line to be searched. Can be 0 to search the entire list.
<i>text</i>	The text to match the beginning of the line

Return Value

The line number selected.

Applies to

[CheckListBox](#), [ListBox](#)

SelRange (ListBox)

Syntax

SelRange *select, start_line, end_line*

Description

Selects or unselects the lines in the list from *start_line* to *end_line* in a multiple selection listbox depending on the value of the *select* flag. Returns 1 if selection successful otherwise 0.

Parameters

The following table describes the parameters of the SelRange method:

Parameters	Description
<i>select</i>	True to select lines, False to deselect lines.
<i>start_line</i>	First line to select/deselect
<i>end_line</i>	Last line to select/deselect

Return Value

True if selection was successful

Applies to

[CheckListBox](#), [ListBox](#)

Select (EditText)

Syntax

Select *from_char_index*, *to_char_index*

Description

Selects the specifed section of the edit control.

Parameters

The following table describes the parameters of the Select method:

Parameters	Description
<i>from_char_index</i>	Character index of first character in selection
<i>to_char_index</i>	Character index of last character to include in the selection

Applies to

[EditText](#)

Select (TrackBar)

Syntax

Select *redraw, start, end*

Description

Selects an area of the trackbar.

The trackbar must have the TBS_ENABLESELRANGE style.

Parameters

The following table describes the parameters of the Select method:

Parameters	Description
<i>redraw</i>	True to redraw the selection immediately
<i>start</i>	Start of the selected range
<i>end</i>	End of the selected range

Applies to

[TrackBar](#)

SelectAll (Grid)

Syntax

SelectAll *select*

Description

The SelectAll method adds or removes all of the cells to/from the list of selected cells and highlights the cells.

Parameters

The following table describes the parameters of the SelectAll method:

Parameters	Description
<i>select</i>	True to select the cell, false to deselect the cell

Applies to

[Grid](#)

Version

4.0.2

SelectCell (Grid)

Syntax

SelectCell *row, column, select*

Description

The SelectCell method adds or removes the specified cell to/from the list of selected cells and highlights the cell.

Parameters

The following table describes the parameters of the SelectCell method:

Parameters	Description
<i>row</i>	The row of the cell to select
<i>column</i>	The column of the cell to select
<i>select</i>	True to select the cell, false to deselect the cell

Applies to

[Grid](#)

Version

4.0.2

SelectInfo (EditText)

Syntax

SelectInfo *start_var, end_var, selection*

Description

Sets the specified variables to the start and end position of the current selection. If there is no current selection the variable are set to the caret position. Returns true if there is a selection.

Parameters

The following table describes the parameters of the SelectInfo method:

Parameters	Description
<i>start_var</i>	Character position of start of selection
<i>end_var</i>	Character position of start of selection
<i>selection</i>	True if there is a selection

Applies to

[EditText](#)

SelectInfo (Grid)

Syntax

SelectInfo

Description

The SelectInfo method returns a dynamic array containing a list of currently selected cells.

Parameters

None

Return Value

Dynamic array of the ranges of cells selected

Applies to

[Grid](#)

SelectItem (TreeView)

Syntax

SelectItem *option, item_id*

Description

The SelectItem method select an item in a treeview and returns True if successful.

Parameters

The following table describes the parameters of the SelectItem method:

Parameters	Description
<i>option</i>	Can be one of the following: 5 - Display it as first visible item in the control, 8 - Set the drag and drop highlight, 9 - Select the item
<i>item_id</i>	ID of the item to select.

Return Value

True if successful.

Applies to

[TreeView](#)

SelectRange (Grid)

Syntax

SelectRange *fromRow*, *fromCol*, *toRow*, *toColumn*, *select*

Description

The SelectRange method adds or removes the specified range of cells to/from the list of selected cells and highlights the cells.

Parameters

The following table describes the parameters of the SelectRange method:

Parameters	Description
<i>fromRow</i>	Start row
<i>fromCol</i>	Start column
<i>toRow</i>	End row
<i>toColumn</i>	End column
<i>select</i>	True to select the cell, false to deselect the cell

Applies to

Grid

Version

4.0.2 Original

SetCellAttribute (Grid)

Syntax

SetCellAttribute *row, col, attribute, value*

Description

The SetCellAttribute method sets the attributes of a cell.

Parameters

The following table describes the parameters of the SetCellAttribute method:

Parameters	Description
<i>row</i>	Row number
<i>col</i>	Column number
<i>attribute</i>	Attribute to set.
<i>value</i>	Value to set. Depends on the attribute being set

Values for attribute

The following attributes can be set:

Value	Description
<i>ToolTipText</i>	Sets tooltip text.
<i>DateFormatType</i>	Format for a date control. 0 - Locale time format, 1 - Locale short date format, 2 - Locale long date format, 3 - Custom format.
<i>DateValidMode</i>	Date validation (use with DateMin, DateMax). 0 - None, 1 - Time, 2 - Date, 3 - Date and time.
<i>DateMin</i>	Minimum date/time allowed.
<i>DateMax</i>	Maximum date/time allowed.

Value	Description
<i>NoEdit</i>	See NoEdit property of the DateTime Control
<i>CustomFormat</i>	See the CustomFormat property of the DateTime Control

Applies to

[Grid](#)

Example

The following example shows the use of the SetCellAttribute method:

```
CustMain.OrderDate.SetCellAttribute 3,2, "DateFormatType", 2
```

SetCellAutoSize (Grid)

Syntax

SetCellAutoSize *row, column, flag*

Description

The SetCellAutoSize method specifies whether a specified cell height will grow to fit the text.

Parameters

The following table describes the parameters of the SetCellAutoSize method:

Parameters	Description
<i>row</i>	Row number.
<i>column</i>	Column number.
<i>flag</i>	True to allow cell height to grow to fit the text.

Applies to

[Grid](#)

SetCellBackColor (Grid)

Syntax

SetCellBackColor *row, column, flag*

Description

The SetCellBackColor method sets the background color for a cell.

Parameters

The following table describes the parameters of the SetCellBackColor method:

Parameters	Description
<i>row</i>	Row number.
<i>column</i>	Column number.
<i>flag</i>	A color.

Applies to

[Grid](#)

SetCellBorder (Grid)

Syntax

SetCellBorder *row, column, position, color, [style], [width]*

Description

The SetCellBorder method sets the color, style, and width of a cell border.

Parameters

The following table describes the parameters of the SetCellBorder method:

Parameters	Description
<i>row</i>	row number
<i>column</i>	Column number
<i>position</i>	Border to change: 0 - Left, 1 - Right, 2 - Top, 3 - Bottom, -1 All
<i>color</i>	Color for the border. See the following RGB Color Names table.
<i>style</i>	See the Draw Pen command for list of styles.
<i>width</i>	The width of the border (style PEN_Solid only).

Applies to

[Grid](#)

SetCellControl (Grid)

Syntax

SetCellControl *row, column, control*

Description

The SetCellControl method sets the control type (for example: push button, radio button, check box, combo box, etc.) for a cell in a grid.

Parameters

The following table describes the parameters of the SetCellControl method:

Parameters	Description
<i>row</i>	Row number.
<i>column</i>	Column number
<i>control</i>	Control type for the cell. See below

Values for control

Value	Description
<i>DateTime</i>	Date and time. Note - To set the format for a date/time control, use the SetCellAttribute method.
<i>DateTimeNoCal</i>	Date and time control without a drop-down calendar.
<i>EditText</i>	edit box.
<i>Text</i>	Static text.
<i>PushButton</i>	Push button. SetCellList sets button text
<i>RadioButton</i>	Radio buttons. SetCellList sets text for buttons

Value	Description
<i>CheckBox</i>	Check box. SetCellList sets text for check box
<i>Combobox</i>	Combo box. SetCellList set list of options
<i>DropList</i>	Combo Box (no user editing of value). Use SetCellList to set the list of options.

Applies to

[Grid](#)

Example

The following example sets the control type to "DropList":

```
GridTest.Grid1.SetCellControl 2,2, DropList  
GridTest.Grid1.SetCellList 2,2, opt 1:cr:opt 2:cr:opt 3
```

SetCellEnabled (Grid)

Syntax

SetCellEnabled *row, column, flag*

Description

The SetCellEnabled method sets whether or not a cell is enabled.

Parameters

The following table describes the parameters of the SetCellEnabled method:

Parameters	Description
<i>row</i>	Row number.
<i>column</i>	Column number.
<i>flag</i>	True for enabled.

Applies to

[Grid](#)

SetCellFloat (Grid)

Syntax

SetCellFloat *row, column, flag*

Description

The SetCellFloat method specifies whether a cell is allowed to overlay cells to the right of it.

Note: The FloatingCells Property must be set before this method has any effect.

Parameters

The following table describes the parameters of the SetCellFloat method:

Parameters	Description
<i>row</i>	Row number
<i>column</i>	Column number
<i>flag</i>	True - Allows the cell to overlay cells to the right of it.

Applies to

[Grid](#)

SetCellFont (Grid)

Syntax

SetCellFont *row, column, name, size, [style]*

Description

The SetCellFont method specifies the font name, size, and style for a cell.

Parameters

The following table describes the parameters of the SetCellFont method:

Parameters	Description
<i>row</i>	Row number.
<i>column</i>	Column number.
<i>name</i>	Name of font.
<i>size</i>	Point size of font.
<i>style</i>	Style for the font can be one of the following values

Values for style

Value	Description
<i>FONT_Light</i>	Sets a lightweight font.
<i>FONT_Normal</i>	Sets a normal-weight font.
<i>FONT_Bold</i>	Sets a bold font.
<i>FONT_Heavy</i>	Sets a heavyweight (bold) font.
<i>FONT_Italic</i>	Sets an italic font.
<i>FONT_Underline</i>	Underlines the text.
<i>FONT_StrikeOut</i>	Places a line through the text.

Applies to

[Grid](#)

SetCellForeColor (Grid)

Syntax

SetCellForeColor *row, column, flag*

Description

The SetCellForeColor method sets the foreground color of a cell.

Parameters

The following table describes the parameters of the SetCellForeColor method:

Parameters	Description
<i>row</i>	Row number.
<i>column</i>	Column number.
<i>flag</i>	The color to set. See the following RGB Color Name table.

Applies to

[Grid](#)

SetCellJustify (Grid)

Syntax

SetCellJustify *row, column, flag*

Description

The SetCellJustify method sets the text justification for a cell.

Parameters

The following table describes the parameters of the SetCellJustify method:

Parameters	Description
<i>row</i>	Row number.
<i>column</i>	Column number.
<i>flag</i>	The type of justification: L - Left, R - Right, C - Centered

Applies to

[Grid](#)

SetCellList (Grid)

Syntax

SetCellList *row, column, list*

Description

The SetCellList method sets a carriage return separated list of options for a combo box or the text for a button control.

Parameters

The following table describes the parameters of the SetCellList method:

Parameters	Description
<i>row</i>	Row number.
<i>column</i>	Column number.
<i>list</i>	A carriage return separated list of options for a combo box or the text for a button control.

Applies to

[Grid](#)

SetCellReadOnly (Grid)

Syntax

SetCellReadOnly *row, column, flag*

Description

The SetCellReadOnly method sets a cell to be read-only.

Parameters

The following table describes the parameters of the SetCellReadOnly method:

Parameters	Description
<i>row</i>	Row number.
<i>column</i>	Column number.
<i>flag</i>	1 to make the cell read-only, 0 to make it writable.

Applies to

[Grid](#)

SetCellValue (Grid)

Syntax

SetCellValue *row, column, value*

Description

The SetCellValue method sets the text for a specified cell.

Parameters

The following table describes the parameters of the SetCellValue method:

Parameters	Description
<i>row</i>	Row number.
<i>column</i>	Column number.
<i>value</i>	Text for the cell.

Applies to

[Grid](#)

SetCheck (CheckListBox)

Syntax

SetCheck *line*, *set*

Description

This method allows an individual check mark to be set or cleared in a check list box.

Parameters

The following table describes the parameters of the SetCheck method:

Parameters	Description
<i>line</i>	The number of the line in the check list box.
<i>set</i>	True to set the check or False to clear the check mark from the specifed check list box line.

Applies to

[CheckListBox](#)

Example

Check the third line of the days check list box

```
MeetingDays.days.SetCheck 3, true
```

SetColWidth (Grid)

Syntax

SetColWidth *col*, *width*

Description

The SetColWidth method sets the width for a specified column.

Parameters

The following table describes the parameters of the SetColWidth method:

Parameters	Description
<i>col</i>	Column number
<i>width</i>	Width of the column

Applies to

[Grid](#)

SetColumn (ListView)

Syntax

SetColumn *column*, *colinfo*

Description

The SetColumn method sets information about a column and returns True if successful.

Note: Left most (first) column is always left justified regardless of the format flags.

Parameters

The following table describes the parameters of the SetColumn method:

Parameters	Description
<i>column</i>	Index of column.
<i>colinfo</i>	Listview column information record. See below.

Values for colinfo

The following table describes the column information record:

Value	Description
<i>1</i>	Mask: 1 - format, 2 - width, 4 - text, 8 - sub item
<i>2</i>	Format: 0 - Left, 1 - Right, 2 - Center
<i>3</i>	Width.
<i>4</i>	Text.
<i>5</i>	Max length of text (when getting text).
<i>6</i>	Sub item.

Return Value

True if successful

Applies to

[ListView](#)

SetColumnWidth (ListView)

Syntax

SetColumnWidth *column, width*

Description

The SetColumnWidth method sets the width of a specified column.

Parameters

The following table describes the parameters of the SetColumnWidth method:

Parameters	Description
<i>column</i>	Column number to modify the width.
<i>width</i>	Width of column or one of the following: -1 to automatically size the column, -2 to size the column to the size of the column label Field Description Column Information Record

Applies to

[ListView](#)

SetCurrentCell (Grid)

Syntax

SetCurrentCell *row, column*

Description

The SetCurrentCell method sets the current cell in the grid and returns true if the current cell moved. The value can then be interrogated modified with the CellValue property.

Parameters

The following table describes the parameters of the SetCurrentCell method:

Parameters	Description
<i>row</i>	Row number.
<i>column</i>	Column number

Applies to

[Grid](#)

SetCurrentText (Grid)

Syntax

SetCurrentText *text*

Description

The SetCurrentText method sets the text for the current cell within the ValidateCell method.

Parameters

The following table describes the parameters of the SetCurrentText method:

Parameters	Description
<i>text</i>	New text for the cell.

Applies to

[Grid](#)

Example

In the following example, the SetCurrentText method is used to output convert the existing value of a cell.

```
* Change entries to upper case when validating a cell
val = GridTest.Grid1.GetCurrentText()
GridTest.Grid1.SetCurrentText Oconv(val, "MCU")
```

SetImageList (ListView)

Syntax

SetImageList *list_id*, *image_list*

Description

The SetImageList method sets the image list to use with a list view. The image list is used to store all the images that are shown in the list view.

Parameters

The following table describes the parameters of the SetImageList method:

Parameters	Description
<i>list_id</i>	ID of the image list: 0 - Image list with large icons, 1 - Image list with small icons, 2 - Image list with state images.
<i>image_list</i>	Name of image list.

Applies to

[ListView](#)

Related Script Commands

ImageList Load

SetImageList (TreeView)

Syntax

SetImageList *which_list*, *image_list_name*

Description

The SetImageList method sets the image lists to use for the icons in the treeview.

Parameters

The following table describes the parameters of the SetImageList method:

Parameters	Description
<i>which_list</i>	Which image list to set: 0 - normal, 2 - state.
<i>image_list_name</i>	Image list name.

Applies to

[TreeView](#)

SetItem (Header)

Syntax

SetItem *index*, *hd_item*

Description

The SetItem method sets the properties of a header item from the details specified in the Header item details record and returns a nonzero value if successful.

Parameters

The following table describes the parameters of the SetItem method:

Parameters	Description
<i>index</i>	Zero based index of item to set.
<i>hd_item</i>	Header item details record.

Values for *hd_item*

The header item details record consists of the following fields:

Value	Description
<i>1</i>	The mask field specifies which of the other fields in the record contains valid data. It is a combination of the following values. 1 - width field is valid, 2 - text field is valid, 4 - format field is valid, 8 - user data is valid
<i>2</i>	Width of item.
<i>3</i>	Text of item.
<i>4</i>	Reserved. Must be "" or 0.
<i>5</i>	Text max, number, maximum size the text can be (for retrieving text from an item).

Value	Description
6	Format flags for item. This specifies the format of the data. Justification (check the return value & 3): 0 - Left justified, 1 - Right justified, 2 - Centered. Other flags: 16384 - Item contains text (required when setting/inserting an item)
7	User data. User defined number

Return Value

Returns non-zero if successful

Applies to

[Header](#)

SetItem (ListView)

Syntax

SetItem *lv_item*

Description

The SetItem method sets the details of an item or subitem and returns True if successful.

Parameters

The following table describes the parameters of the SetItem method:

Parameters	Description
<i>lv_item</i>	lvitem Details of item to be set in a Listview item details record. See below.

Values for lv_item

The following fields describe the list view item record:

Value	Description
1	mask. Specifies which other fields are valid. A combination of: 1 - Text field is valid, 2 - Image id is valid, 4 - user data is valid, 8 - state is valid
2	index of item
3	index of sub item
4	state. Current state bits: 1 - item has the focus, 2 - item is selected, 4 - item is displayed as if it has been cut, 8 - item is highlighted as a drag and drop target
5	state mask. Specified the state bits to get/set
6	text

Value	Description
7	maximum size of text (use when getting text)
8	image id
9	data. User defined data

Return Value

Returns true if successful

Applies to

[ListView](#)

SetItem (TreeView)

Syntax

SetItem *tv_item*

Description

The SetItem method sets the details of an item in the tree view and returns True if successful.

Parameters

The following table describes the parameters of the SetItem method:

Parameters	Description
<i>tv_item</i>	Treeview item details record. See below

Values for tv_item

The following table describes the Treeview item details record:

Value	Description
<i>1</i>	The mask field specifies which of the other fields in the record contains valid data. It is a combination of the following values. 1 - Text field is valid, 2 - Image field is valid, 4 - Param field is valid, 8 - State field is valid, 16 - Item id field is valid, 32 - Selected image field is valid, 64 - Children count field is valid.
<i>2</i>	The item id of the a line in the treeview

Value	Description
3	The state of bits to be set for the item. The bits which are valid are specified in the state mask and can be a combination of the following values: 2 - Item is selected, 4 - Item is cut, 8 - Item is drop highlighted, 16 - Item is bold, 32 - Item is expanded, 64 - Item has been expanded at least once, 128 - Item has been partially expanded.
4	The state of bits to be retrieved for the item. The bits which are valid are specified in the state mask and can be a combination of the following values: 2 - Item is selected, 4 - Item is cut, 8 - Item is drop highlighted, 16 - Item is bold, 32 - Item is expanded, 64 - Item has been expanded at least once, 128 - Item has been partially expanded.
5	Text entry for the item.
6	Maximum size of the text can (for retrieving text from an item).
7	Image number, normal image index.
8	Selected image number, selected image index.
9	Count of children for an item
10	User specified number associated with this item.

Return Value

True if successful.

Applies to

[TreeView](#)

SetItemCount (ListView)

Syntax

SetItemCount *count*

Description

The SetItemCount method to specify the eventual number of items it will have when adding a large number of items.

Parameters

The following table describes the parameters of the SetItemCount method:

Parameters	Description
<i>count</i>	Number of items.

Applies to

[ListView](#)

SetItemData (ComboBox)

Syntax

SetItemData *line_number, data*

Description

Set an number against the specified line of a list box. This number is associated with the text for the line and not the line number so it will stay with the text even when lines are inserted or deleted before this line.

Parameters

The following table describes the parameters of the SetItemData method:

Parameters	Description
<i>line_number</i>	The number of the line in which to store the data
<i>data</i>	The number to store against the specified line

Applies to

[ComboBox](#)

SetItemData (ListBox)

Syntax

SetItemData *line_number, data*

Description

Set an number against the specified line of a list box. This number is associated with the text for the line and not the line number so it will stay with the text even when lines are inserted or deleted before this line.

Parameters

The following table describes the parameters of the SetItemData method:

Parameters	Description
<i>line_number</i>	The line number to store the data number against
<i>data</i>	The number to store

Applies to

[ListBox](#)

SetItemData (TreeView)

Syntax

SetItemData *item_id*, *data*

Description

The SetItemData method sets a number against an item and returns True if successful.

Parameters

The following table describes the parameters of the SetItemData method:

Parameters	Description
<i>item_id</i>	ID of item
<i>data</i>	The number to store against this item

Return Value

True if successful.

Applies to

[TreeView](#)

SetItemImage (TreeView)

Syntax

SetItemImage *item_id*, *image*, [*sel_image*]

Description

The SetItemImage method sets the image or images for an item.

Parameters

The following table describes the parameters of the SetItemImage method:

Parameters	Description
<i>item_id</i>	ID of the item
<i>image</i>	Index of the image.
<i>sel_image</i>	Index of the image when it is in a selected state.

Applies to

[TreeView](#)

SetItemPosition (ListView)

Syntax

SetItemPosition *item_no*, *x*, *y*

Description

The SetItemPosition method moves a specified item to the required position.

Parameters

The following table describes the parameters of the SetItemPosition method:

Parameters	Description
<i>item_no</i>	Index of item to move.
<i>x</i>	New x co-ordinate.
<i>y</i>	New y co-ordinate.

Applies to

[ListView](#)

SetItemSize (TabControl)

Syntax

SetItemSize *width, depth*

Description

The SetItemSize method sets the item size for fixed size tabs.

Parameters

The following table describes the parameters of the SetItemSize method:

Parameters	Description
<i>width</i>	New width.
<i>depth</i>	New depth or height.

Applies to

[TabControl](#)

SetItemState (ListView)

Syntax

SetItemState *item_no*, *set_mask*, [*clear_mask*]

Description

The SetItemState method sets the state of an item and returns True if the mask modified.

Parameters

The following table describes the parameters of the SetItemState method:

Parameters	Description
<i>item_no</i>	Item to change the state.
<i>set_mask</i>	Bits to set. See below
<i>clear_mask</i>	Bits to clear. See set_mask below for list of bits.

Values for set_mask

Value	Description
1	item has the focus
2	item is selected
4	item is displayed as if it has been cut
8	item is highlighted as a drag and drop target

Return Value

True if the mask has been modified

Applies to

[ListView](#)

SetItemState (TreeView)

Syntax

SetItemState *item_id, state, state_mask*

Description

The SetItemState method sets an items state and returns True if successful.

Parameters

The following table describes the parameters of the SetItemState method:

Parameters	Description
<i>item_id</i>	Item number
<i>state</i>	Bits to set for the state. See Below. Only the bits specified in the state mask are changed.
<i>state_mask</i>	Bits to change in the current state. See state table below for values.

Values for state

State bits. Combine using the "|" symbol or addition

Value	Description
2	Item is selected
4	Item is cut
8	Item is drop highlighted
16	Item is bold
32	Item is expanded
64	Item has been expanded at least once
128	Item has been partially expanded.

Return Value

True if successful.

Applies to

[TreeView](#)

SetItemText (Header)

Syntax

SetItemText *index, text, [width]*

Description

The SetItem method sets the text for a header item and returns a nonzero value if successful.

Parameters

The following table describes the parameters of the SetItemText method:

Parameters	Description
<i>index</i>	Zero based index for item
<i>text</i>	Text to insert SetItem Parameters.DialogBox Header
<i>width</i>	Width for the item.

Return Value

Non-zero if successful

Applies to

[Header](#)

SetItemText (ListView)

Syntax

SetItemText *item_no*, *subitem_no*, *text*

Description

The SetItemText method sets the text for an item and returns True if successful.

Parameters

The following table describes the parameters of the SetItemText method:

Parameters	Description
<i>item_no</i>	Index of item to set text of.
<i>subitem_no</i>	Subitem index or 0 for main label.
<i>text</i>	New text.

Return Value

True if successful.

Applies to

[ListView](#)

SetItemText (TabControl)

Syntax

SetItemText *index*, *text*

Description

The SetItemText method sets the label for a tab and returns True if successful.

Parameters

The following table describes the parameters of the SetItemText method:

Parameters	Description
<i>index</i>	Index of item.
<i>text</i>	Text for the tab.

Return Value

True if successful.

Applies to

[TabControl](#)

SetItemText (TreeView)

Syntax

SetItemText *item_id*, *text*

Description

The SetItemText method sets the text for an item in a treeview and returns True if successful.

Parameters

The following table describes the parameters of the SetItemText method:

Parameters	Description
<i>item_id</i>	ID of item
<i>text</i>	text for the item

Return Value

True if successful.

Applies to

[TreeView](#)

SetItemWidth (Header)

Syntax

SetItemWidth *index*, *width*

Description

The SetItemWidth method set the width of a header item and returns a non-zero value if successful.

Parameters

The following table describes the parameters of the SetItemWidth method:

Parameters	Description
<i>index</i>	Zero based index for item.
<i>width</i>	Width of the item.

Return Value

Non-zero if successful

Applies to

[Header](#)

SetPadding (TabControl)

Syntax

SetPadding *x*, *y*

Description

The SetPadding method sets the space around the text of a tab.

Parameters

The following table describes the parameters of the SetPadding method:

Parameters	Description
<i>x</i>	Horizontal padding.
<i>y</i>	Vertical padding.

Applies to

[TabControl](#)

SetRangeAttribute (Grid)

Syntax

SetRangeAttribute *from_row, from_col, to_row, to_col, attribute, value*

Description

The SetRangeAttribute method sets an attribute for cells within the specified range.

See SetCellAttribute method for currently supported attributes and their values.

Parameters

The following table describes the parameters of the SetRangeAttribute method:

Parameters	Description
<i>from_row</i>	Start row.
<i>from_col</i>	Start column.
<i>to_row</i>	End row.
<i>to_col</i>	End column.
<i>attribute</i>	Attribute to set. See SetCellAttribute method for currently supported attributes and their values
<i>value</i>	Value for the attribute.

Applies to

[Grid](#)

SetRangeAutoSize (Grid)

Syntax

SetRangeAutoSize *from_row, from_col, to_row, to_col, flag*

Description

The SetRangeAutoSize method allows a specified range of cells to grow in height to fit the text.

Parameters

The following table describes the parameters of the SetRangeAutoSize method:

Parameters	Description
<i>from_row</i>	Start row.
<i>from_col</i>	Start column.
<i>to_row</i>	End row.
<i>to_col</i>	End column. flag True - Allow cell height to grow to fit the text.
<i>flag</i>	True - Allow cell height to grow to fit the text.

Applies to

[Grid](#)

SetRangeBackColor (Grid)

Syntax

SetRangeBackColor *from_row, from_col, to_row, to_col, color*

Description

The SetRangeBackColor method sets the background color for cells within the specified range.

Parameters

The following table describes the parameters of the SetRangeBackColor method:

Parameters	Description
<i>from_row</i>	Start row.
<i>from_col</i>	Start column.
<i>to_row</i>	End row.
<i>to_col</i>	End column.
<i>color</i>	The color to set.

Applies to

[Grid](#)

SetRangeBorder (Grid)

Syntax

SetRangeBorder *from_row, from_col, to_row, to_col, position, color, [style], [width]*

Description

The SetRangeBorder method sets the color, style, and width of a border for cells within the specified range.

Parameters

The following table describes the parameters of the SetRangeBorder method:

Parameters	Description
<i>from_row</i>	Start row.
<i>from_col</i>	Start column.
<i>to_row</i>	End row.
<i>to_col</i>	End column.
<i>position</i>	Border to change: 0 - Left, 1 - Right, 2 - Top, 3 - Bottom,-1 All
<i>color</i>	Color of the border. See the following RGB Color Name table.
<i>style</i>	See the Draw Pen command for list of styles.
<i>width</i>	Width of pen (style PEN_Solid only).

Applies to

[Grid](#)

SetRangeControl (Grid)

Syntax

SetRangeControl *from_row, from_col, to_row, to_col, control*

Description

The SetRangeControl method sets the control type (for example: push button, radio button, check box, combo box, etc.) for cells within the specified range in a grid.

Parameters

The following table describes the parameters of the SetRangeControl method:

Parameters	Description
<i>from_row</i>	Start row.
<i>from_col</i>	Start column.
<i>to_row</i>	End row.
<i>to_col</i>	End column
<i>control</i>	Control type to use. See below

Values for control

Value	Description
<i>EditText</i>	edit box.
<i>Text</i>	Static text.
<i>PushButton</i>	Push button. SetCellList sets button text.
<i>RadioButton</i>	Radio buttons. SetCellList sets text for buttons.
<i>CheckBox</i>	Check box. SetCellList sets text for check box.
<i>ComboBox</i>	Combo Box. SetCellList set list of options.

Value	Description
<i>DropList</i>	Combo Box (no user editing of value). Use SetCellList to set the list of options.
<i>DateTime</i>	Date/Time control. Use SetRangeAttribute to set properties of the control

Applies to

[Grid](#)

SetRangeEnabled (Grid)

Syntax

SetRangeEnabled *from_row, from_col, to_row, to_col, enabled*

Description

The SetRangeEnabled method sets whether or not cells within the specified range is enabled.

Parameters

The following table describes the parameters of the SetRangeEnabled method:

Parameters	Description
<i>from_row</i>	Start row.
<i>from_col</i>	Start column.
<i>to_row</i>	End row.
<i>to_col</i>	End column. flag True for enabled.
<i>enabled</i>	True for enabled.

Applies to

[Grid](#)

SetRangeFloat (Grid)

Syntax

SetRangeFloat *from_row, from_col, to_row, to_col, flag*

Description

The SetRangeFloat method specifies whether cells within the specified range is allowed to overlay.

Note: The FloatingCells Property must be set before this method has any effect

Parameters

The following table describes the parameters of the SetRangeFloat method:

Parameters	Description
<i>from_row</i>	Start row.
<i>from_col</i>	Start column.
<i>to_row</i>	End row.
<i>to_col</i>	End column.
<i>flag</i>	True to allow cells to overlay cells to the right of it.

Applies to

[Grid](#)

SetRangeFont (Grid)

Syntax

SetRangeFont *from_row, from_col, to_row, to_col, name, size, [style]*

Description

The SetRangeFont method specifies the font name, size, and style for cells within the specified range.

Parameters

The following table describes the parameters of the SetRangeFont method:

Parameters	Description
<i>from_row</i>	Start row.
<i>from_col</i>	Start column.
<i>to_row</i>	End row.
<i>to_col</i>	End column.
<i>name</i>	Name of the font.
<i>size</i>	Point size of the font.
<i>style</i>	The style of font to use: See below

Values for style

Value	Description
<i>FONT_Light</i>	Sets a lightweight font.
<i>FONT_Normal</i>	Sets a normal-weight font.
<i>FONT_Bold</i>	Sets a bold font.
<i>FONT_Heavy</i>	Sets a heavyweight (bold) font.
<i>FONT_Italic</i>	Sets an italic font.
<i>FONT_Underline</i>	Underlines the text.

Value	Description
<i>FONT_StrikeOut</i>	Places a line through the text.

Applies to

[Grid](#)

SetRangeForeColor (Grid)

Syntax

SetRangeForeColor *from_row, from_col, to_row, to_col, value*

Description

The SetRangeForeColor method sets the foreground color for cells within the specified range.

Parameters

The following table describes the parameters of the SetRangeForeColor method:

Parameters	Description
<i>from_row</i>	Start row.
<i>from_col</i>	Start column.
<i>to_row</i>	End row.
<i>to_col</i>	End column.
<i>value</i>	The color to set.

Applies to

[Grid](#)

SetRangeJustify (Grid)

Syntax

SetRangeJustify *from_row, from_col, to_row, to_col, flag*

Description

The SetRangeJustify method sets the text justification for cells within the specified range.

Parameters

The following table describes the parameters of the SetRangeJustify method:

Parameters	Description
<i>from_row</i>	Start row.
<i>from_col</i>	Start column.
<i>to_row</i>	End row.
<i>to_col</i>	End column.
<i>flag</i>	The type of justification: L - Left, R - Right, C - Centered

Applies to

[Grid](#)

SetRangeList (Grid)

Syntax

SetRangeList *from_row, from_col, to_row, to_col, list*

Description

The SetRangeList method sets a carriage return separated list of options for a combo box or the text for a button control.

Parameters

The following table describes the parameters of the SetRangeList method:

Parameters	Description
<i>from_row</i>	Start row.
<i>from_col</i>	Start column.
<i>to_row</i>	End row.
<i>to_col</i>	End column.
<i>list</i>	A carriage return separated list of options for a combo box or the text for a button control.

Applies to

[Grid](#)

SetRangeReadOnly (Grid)

Syntax

SetRangeReadOnly *from_row, from_col, to_row, to_col, flag*

Description

The SetRangeReadOnly method sets cells within the specified range to be read-only.

Parameters

The following table describes the parameters of the SetRangeReadOnly method:

Parameters	Description
<i>from_row</i>	Start row.
<i>from_col</i>	Start column.
<i>to_row</i>	End row.
<i>to_col</i>	End column.
<i>flag</i>	1 to make the cell read-only, 0 to make it writable.

Applies to

[Grid](#)

SetRangeValue (Grid)

Syntax

SetRangeValue *from_row, from_col, to_row, to_col, value*

Description

The SetRangeValue method sets the text for cells within the specified range.

Parameters

The following table describes the parameters of the SetRangeValue method:

Parameters	Description
<i>from_row</i>	Start row.
<i>from_col</i>	Start column.
<i>to_row</i>	End row.
<i>to_col</i>	End column.
<i>value</i>	A dynamic array of text values to put in the range of cells.

Applies to

[Grid](#)

SetRowHeight (Grid)

Syntax

SetRowHeight *row, height*

Description

The SetRowHeight method sets the height of a row.

Parameters

The following table describes the parameters of the SetRowHeight method:

Parameters	Description
<i>row</i>	Row number
<i>height</i>	The height for the row.

Applies to

[Grid](#)

SetSel (ListBox)

Syntax

SetSel *select*, *line_number*

Description

Selects or unselects specified line number depending on the value of the select flag in a multiple selection listbox.

Parameters

The following table describes the parameters of the SetSel method:

Parameters	Description
<i>select</i>	True to select the line, False to deselect the line
<i>line_number</i>	Line number of the line to select/deselect

Applies to

[CheckListBox](#), [ListBox](#)

SetTick (TrackBar)

Syntax

SetTick *position*

Description

Sets a tick mark at the specified position

Parameters

The following table describes the parameters of the SetTick method:

Parameters	Description
<i>position</i>	position to set the tick mark at

Applies to

[TrackBar](#)

SortChildren (TreeView)

Syntax

SortChildren *item_id*

Description

The SortChildren method sorts the children of a branch and returns True if successful.

Parameters

The following table describes the parameters of the SortChildren method:

Parameters	Description
<i>item_id</i>	Item ID of the branch to sort.

Return Value

True if successful.

Applies to

[TreeView](#)

SortCols (Grid)

Syntax

SortCols *by_row1*, [*by_row2*], [*by_rowN*]

Description

The SortCols method sorts the grid by the specified value in the row.

Parameters

The following table describes the parameters of the SortCols method:

Parameters	Description
<i>by_row1</i>	The rows can be specified as either a number or a string made up of the following components, which modify the sort order and specifies the data to be sorted:
<i>by_row2</i>	Second row to sort by
<i>by_rowN</i>	Nth row to sort by. Up to 10 rows can be specified for the sort

Values for *by_row1*

Value	Description
<i>number</i>	The number of the row.
<i>A</i>	ascending sort (the default)
<i>D</i>	Descending sort
<i>N</i>	Numeric sort (Default autodetect)
<i>X</i>	Alpha numeric sort (Default autodetect)
<i>T</i>	Date (Default autodetect)
<i>C</i>	Case sensitive (not case sensitive by default).

Applies to

Grid

SortRows (Grid)

Syntax

SortRows *by_col1*, [*by_col2*], [*by_colN*]

Description

The SortRows method sorts the grid by the specified columns.

Parameters

The following table describes the parameters of the SortRows method:

Parameters	Description
<i>by_col1</i>	The columns can be specified as either a number or a string made up of the following components which modify the sort order and specifies the data to be sorted:
<i>by_col2</i>	Second column to sort by
<i>by_colN</i>	Nth column to sort by. Up to 10 columns can be used

Values for by_col1

Value	Description
<i>number</i>	The number of the row.
<i>A</i>	ascending sort (the default)
<i>D</i>	Descending sort
<i>N</i>	Numeric sort (Default autodetect)
<i>X</i>	Alpha numeric sort (Default autodetect)
<i>T</i>	Date (Default autodetect)
<i>C</i>	Case sensitive (not case sensitive by default).

Applies to

Grid

Example

In the following example, the grid is first sorted by column 1 and then by column 2 in descending order.

```
GridTest.Grid1.SortRows "1,2DN"
```

StepIt (ProgressBar)

Syntax

StepIt

Description

The StepIt method moves the progress bar by a step increment

Parameters

None

Applies to

[ProgressBar](#)

Stop (Animate)

Syntax

Stop

Description

The Stop method stops playing the AVI file.

Parameters

None

Applies to

[Animate](#)

TickCount (TrackBar)

Syntax

TickCount

Description

Returns the number of ticks defined on the trackbar.

Parameters

None

Return Value

Number of ticks on the Track bar

Applies to

[TrackBar](#)

Undo

Syntax

Undo

Description

Undoes the last edit of the control.

Parameters

None

Applies to

[EditText](#)

Update (ListView)

Syntax

Update *item_no*

Description

The Update method updates a specified item and returns True is successful.

Note: If the listview has the AutoArrange style set, the listview is re-arranged.

Parameters

The following table describes the parameters of the Update method:

Parameters	Description
<i>item_no</i>	Index of item to set text.

Return Value

True if successful

Applies to

[ListView](#)

VisibleCount (TreeView)

Syntax

VisibleCount

Description

The VisibleCount method returns a count of the maximum number of complete items that can display.

Parameters

None

Return Value

Count of the maximum number of complete items that can display.

Applies to

[TreeView](#)

Script Control Events

Events are actions or changes of state of a control.

To run a script when an event is raised by a control use the DialogBox OnEvent script command.

Some events pass extra parameters to the script that give extra information on the event.

The DialogBox Reply command is used in conjunction with events that have information passed back into the control.

Activate (Grid)

Syntax

Activate

Event Number: 3

Description

Called when the grid is activated

This is an Active X event so it must be referred to by its number in the DialogBox OnEvent command.

Parameters

None

Applies to

[Grid](#)

BeginLabelEdit (ListView)

Syntax

BeginLabelEdit *item_no*

Description

Item is about to be edited. Return True to stop editing

Parameters

The following table describes the parameters of the BeginLabelEdit event:

Parameters	Description
<i>item_no</i>	Number of item about to be edited

Applies to

[ListView](#)

BeginLabelEdit (TreeView)

Syntax

BeginLabelEdit *item_id, state, text, data*

Description

Label is about to be edited

Parameters

The following table describes the parameters of the BeginLabelEdit event:

Parameters	Description
<i>item_id</i>	ID of item whose label is to be edited
<i>state</i>	Item state
<i>text</i>	text of the item
<i>data</i>	Number stored against the item

Applies to

[TreeView](#)

BeginTrack (Header)

Syntax

BeginTrack *item_no*

Description

Start of column resize.

Return True to prevent tracking.

Parameters

The following table describes the parameters of the BeginTrack event:

Parameters	Description
<i>item_no</i>	item number of the header item being tracked

Applies to

[Header](#)

ButtonClick (Grid)

Syntax

ButtonClick *row, column*

Event Number: 5

Description

Called when a user selects any of the button controls in the grid (e.g. push buttons, radio buttons, check boxes)

This is an Active X event so it must be referred to by its number in the DialogBox OnEvent command.

Parameters

The following table describes the parameters of the ButtonClick event:

Parameters	Description
<i>row</i>	Row containing button that was clicked
<i>column</i>	Column containing button that was clicked

Applies to

[Grid](#)

CellClick (Grid)

Syntax

CellClick *row, column, flags, x, y*

Event Number: 6

Description

Called when the left mouse button is clicked on a grid.

This is an Active X event so it must be referred to by its number in the DialogBox OnEvent command.

Parameters

The following table describes the parameters of the CellClick event:

Parameters	Description
<i>row</i>	row where the click occurred
<i>column</i>	column where the click occurred
<i>flags</i>	Shift/control key state flags
<i>x</i>	x position in pixels
<i>y</i>	y position in pixels

Applies to

[Grid](#)

CellRightClick (Grid)

Syntax

CellRightClick *row, column, flags, x, y*

Event Number: 7

Description

Called when the right mouse button is clicked on a grid.

This is an Active X event so it must be referred to by its number in the DialogBox OnEvent command.

Parameters

The following table describes the parameters of the CellRightClick event:

Parameters	Description
<i>row</i>	row number where the click occurred.
<i>column</i>	column number where the click occurred.
<i>flags</i>	Shift/control key state flags
<i>x</i>	x position in pixels
<i>y</i>	y position in pixels

Applies to

[Grid](#)

Change (EditText)

Syntax

Change

Description

The text has been changed

Parameters

None

Applies to

[EditText](#)

Changed (DateTime)

Syntax

Changed

Description

The value of the date time control has changed

Parameters

None

Applies to

[DateTime](#)

Click

Syntax

Click

Description

Left mouse clicked on the control

Parameters

None

Applies to

[CheckBox](#), [ColorButton](#), [Draw](#), [Graphic](#), [GraphicButton](#), [PushButton](#), [RadioButton](#), [TreeView](#)

Click (ListView)

Syntax

Click *item_no*, *subitem_no*, *x*, *y*

Description

Left mouse click.

Parameters

The following table describes the parameters of the Click event:

Parameters	Description
<i>item_no</i>	Item that has been clicked upon
<i>subitem_no</i>	Column number that has been clicked upon
<i>x</i>	x co-ordinate in pixels
<i>y</i>	y co-ordinate in pixels

Applies to

[ListView](#)

CloseUp (ComboBox)

Syntax

CloseUp

Description

The dropdown list of the combobox has been closed

Parameters

None

Applies to

[ComboBox](#)

ColumnClick (ListView)

Syntax

ColumnClick *column_no*

Description

Click on a column header.

Parameters

The following table describes the parameters of the ColumnClick event:

Parameters	Description
<i>column_no</i>	Number of column clicked upon

Applies to

[ListView](#)

CurrentCell (Grid)

Syntax

CurrentCell *row, column*

Event Number: 8

Description

Called when the cursor moves to a new cell in the grid

This is an Active X event so it must be referred to by its number in the DialogBox OnEvent command.

Parameters

The following table describes the parameters of the CurrentCell event:

Parameters	Description
<i>row</i>	The new row
<i>column</i>	The new column

Applies to

[Grid](#)

DblClick

Syntax

DblClick

Description

Left mouse double clicked on the control.

Combo Boxes controls only receive this event if they have the CBS_SIMPLE style.

Button controls only receive this event if they have the BS_NOTIFY style.

Parameters

None

Applies to

[CheckBox](#), [CheckListBox](#), [ColorButton](#), [ComboBox](#), [ListBox](#), [PushButton](#), [RadioButton](#), [TreeView](#)

DbClick (ListView)

Syntax

DbClick *item_no*, *subitem_no*, *x*, *y*

Description

Left mouse double click.

Parameters

The following table describes the parameters of the DbClick event:

Parameters	Description
<i>item_no</i>	Number of item double clicked upon
<i>subitem_no</i>	Column clicked upon
<i>x</i>	x co-ordinate
<i>y</i>	y co-ordinate

Applies to

[ListView](#)

Deactivate (Grid)

Syntax

Deactivate

Event Number: 4

Description

Called when the grid is deactivated.

This is an Active X event so it must be referred to by its number in the DialogBox OnEvent command.

Parameters

None

Applies to

[Grid](#)

DividerDbClick (Header)

Syntax

DividerDbClick *item_no*

Description

Double click on the divider between headings

Parameters

The following table describes the parameters of the DividerDbClick event:

Parameters	Description
<i>item_no</i>	item number

Applies to

[Header](#)

Drag (Divider)

Syntax

Drag *position*

Event Number: 2

Description

A drag is in progress.

This is an Active X event so it must be referred to by its number in the DialogBox OnEvent command.

Parameters

The following table describes the parameters of the Drag event:

Parameters	Description
<i>position</i>	The current position of the top/left of the divider in pixels

Applies to

[Divider](#)

Example

Allow divider to be dragged across a dialog box

```
* Added Between DialogBox Create/EndCreate
AxControl Divider1,"wIntegrate.Divider.1",77,1,6,141
* Event: Drag = 2
OnEvent Divider1,2,"DividerTest.Divider1.Left = position", position
```

DragEnd

Syntax

DragEnd *handle, drop_effect*

Description

The dragging of data from this control has finished.

Parameters

The following table describes the parameters of the DragEnd event:

Parameters	Description
<i>handle</i>	The handle is the item index, id or character position of the selection being dragged
<i>drop_effect</i>	The drop effect when the drag finished. See below.

Vaues for drop_effect

The drop effect will be one of the following.

Value	Description
<i>DE_MOVE</i>	the data is being moved
<i>DE_COPY</i>	the data is being copied

Applies to

[EditText](#), [ListBox](#), [ListView](#), [TreeView](#)

Example

Delete the item dragged if the drag was a move

```
OnEvent ListView1, "DragEnd", "If drop_effect & DE_MOVE Then  
DragLV.ListView1.DeleteItem item", item, drop_effect
```

Version

4.2 Original version

DragEnded (Divider)

Syntax

DragEnded *position*

Event Number: 3

Description

A drag has finished

This is an Active X event so it must be referred to by its number in the DialogBox OnEvent command.

Parameters

The following table describes the parameters of the DragEnded event:

Parameters	Description
<i>position</i>	Current position of the top/left of the divider in pixels

Applies to

[Divider](#)

DragEnter

Syntax

DragEnter *source, keystate, x, y*

Description

The mouse cursor that is dragging some data has entered this control.

Use DialogBox Reply to specify the the drop effect(s) that can be used in this control.

The drop effect can be:

DE_NONE - no drop effect, you can not drop data on this control

DE_MOVE - data can be moved to this control

DE_COPY - data can be copied to this control

DE_MOVE|DE_COPY - data can be copied or moved to this control

If DialogBox Reply is not used the drop effect is move unless the Control key is held down when it is copy.

Parameters

The following table describes the parameters of the DragEnter event:

Parameters	Description
<i>source</i>	The name of the dialog box control the data has come from or "" if the data has not come from a control
<i>keystate</i>	The state of the mouse buttons and shift keys. See below.
<i>x</i>	x co-ordinate of the drag cursor over the control
<i>y</i>	y co-ordinate of the drag cursor over the control

Vaues for keystate

The keystate is a combination of:

Value	Description
<i>MK_LBUTTON</i>	left mouse button down
<i>MK_RBUTTON</i>	right mouse button down
<i>MK_MBUTTON</i>	middle mouse button down
<i>MK_SHIFT</i>	shift button is held down
<i>MK_CONTROL</i>	control button is held down

Applies to

[Animate](#), [CheckBox](#), [CheckListBox](#), [ColorButton](#), [ComboBox](#), [DateTime](#), [EditText](#), [Graphic](#), [GraphicButton](#), [Header](#), [ListBox](#), [ListView](#), [ProgressBar](#), [PushButton](#), [RadioButton](#), [ScrollBar](#), [TabControl](#), [TrackBar](#), [TreeView](#), [UpDownButton](#)

Example

Restrict data that can be dropped here

```
* Only allow data dragged from the control (List2) to be dropped here.
OnEvent List2, "DragEnter", "If source # 'DragList.List2' Then
DialogBox Reply DragList,DE_NONE", source
```

Version

4.2 Original

DragLeave

Syntax

DragLeave

Description

Dragged data is leaving a control

Parameters

None

Applies to

[Animate](#), [CheckBox](#), [CheckListBox](#), [ColorButton](#), [ComboBox](#), [DateTime](#), [EditText](#), [Graphic](#), [GraphicButton](#), [Header](#), [ListBox](#), [ListView](#), [ProgressBar](#), [PushButton](#), [RadioButton](#), [ScrollBar](#), [TabControl](#), [TrackBar](#), [TreeView](#), [UpDownButton](#)

DragOver

Syntax

DragOver *source, keystate, x, y*

Description

The mouse cursor is dragging some data over this control.

Use DialogBox Reply to specify the the drop effect(s) that can be used in this control.

The drop effect can be:

DE_NONE - no drop effect, you can not drop data on this control

DE_MOVE - data can be moved to this control

DE_COPY - data can be copied to this control

DE_MOVE|DE_COPY - data can be copied or moved to this control

If DialogBox Reply is not used the drop effect is move unless the Control key is held down when it is copy.

Parameters

The following table describes the parameters of the DragOver event:

Parameters	Description
<i>source</i>	The name of the dialog box control the data has come from or "" if the data has not come from a control
<i>keystate</i>	The state of the mouse buttons and shift keys. See below.
<i>x</i>	x co-ordinate of the drag cursor over the control
<i>y</i>	y co-ordinate of the drag cursor over the control

Vaues for keystate

The keystate is a combination of:

Value	Description
<i>MK_LBUTTON</i>	left mouse button down
<i>MK_RBUTTON</i>	right mouse button down
<i>MK_MBUTTON</i>	middle mouse button down
<i>MK_SHIFT</i>	shift button is held down
<i>MK_CONTROL</i>	control button is held down

Applies to

[Animate](#), [CheckBox](#), [CheckListBox](#), [ColorButton](#), [ComboBox](#), [DateTime](#), [EditText](#), [Graphic](#), [GraphicButton](#), [Header](#), [ListBox](#), [ListView](#), [ProgressBar](#), [PushButton](#), [RadioButton](#), [ScrollBar](#), [TabControl](#), [TrackBar](#), [TreeView](#), [UpDownButton](#)

Example

Restrict data that can be dropped here

```
* Only allow data dragged from the control (List2) to be dropped here.
OnEvent List2, "DragEnter", "If source # 'DragList.List2' Then
DialogBox Reply DragList,DE_NONE", source
```

Version

4.2 Original

DragStart

Syntax

DragStart *handle*

Description

Data is being dragged from this control.

The data to be dragged can be changed by setting the DragData property during the DragStart event.

The type of drag allowed can be set using the DialogBox Reply command with the following values:

DE_NONE - no dragging allowed

DE_MOVE - data can be moved from this control

DE_COPY - data can be copied from this control

DE_MOVE|DE_COPY - data can be copied or moved from this control

Parameters

The following table describes the parameters of the DragStart event:

Parameters	Description
<i>handle</i>	The handle is the item index, id or character position of the selection being dragged.

Applies to

[EditText](#), [ListBox](#), [ListView](#), [TreeView](#)

Example

Only allow data to be copied from this list box

```
OnEvent List3,"DragStart", "DialogBox Reply DragList,DE_COPY"
```

Drag index number from a list box

* Make the list view drag the item index instead the text value

```
OnEvent ListView1, "DragStart","DragLV.ListView1.DragData = item",item
```

Version

4.2 Original

DragStarted (Divider)

Syntax

DragStarted *position*

Event Number: 1

Description

A drag has started

This is an Active X event so it must be referred to by its number in the DialogBox OnEvent command.

Parameters

The following table describes the parameters of the DragStarted event:

Parameters	Description
<i>position</i>	Current position of the top/left of the divider in pixels.

Applies to

[Divider](#)

Drop

Syntax

Drop *value, source, x, y, drop_effect*

Description

Data has been dropped on this control.

Use DialogBox Reply with the value 0 if the drop can not be completed.

Parameters

The following table describes the parameters of the Drop event:

Parameters	Description
<i>value</i>	The text dropped on this control
<i>source</i>	The name of the dialog box control that the data is being dragged from or "" if the data has not come from another control.
<i>x</i>	x co-ordinate of where the data has been dropped.
<i>y</i>	y co-ordinate of where the data has been dropped.
<i>drop_effect</i>	Drop effect at the time the data is dropped. See below.

Vaues for drop_effect

Drop effects are:

Value	Description
<i>DE_MOVE</i>	the data is being moved
<i>DE_COPY</i>	the data is being copied

Applies to

[Animate](#), [CheckBox](#), [CheckListBox](#), [ColorButton](#), [ComboBox](#), [DateTime](#), [EditText](#), [Graphic](#), [GraphicButton](#), [Header](#), [ListBox](#), [ListView](#), [ProgressBar](#), [PushButton](#), [RadioButton](#), [ScrollBar](#), [TabControl](#), [TrackBar](#), [TreeView](#), [UpDownButton](#)

Example

Add dropped text to the end of a list box

```
* Text dropped on list box List1 will be added to the end of the list
OnEvent List1, "Drop", "DropTest.List1.AddLine value", value
```

Drop text at mouse position in an edit control

```
* The following subroutine is called by the drop event from an edit
control
* to drop the text at the mouse position
Sub OnEditDrop()

With DropTest.Edit1
idx = Field(.CharFromPos(x,y),",",1)
.Select idx,idx
.ReplaceSel value
EndWith
DropTest.Msg = "Text dropped at mouse position in edit control"

EndSub
```

Version

4.2 Original

DropDown (ComboBox)

Syntax

DropDown

Description

The dropdown list of the combobox has been dropped

Parameters

None

Applies to

[ComboBox](#)

EditChange (ComboBox)

Syntax

EditChange

Description

The edit section of the combo box has been changed

Parameters

None

Applies to

[ComboBox](#)

EditUpdate (ComboBox)

Syntax

EditUpdate

Description

The edit section of the combo box has been changed, but not yet displayed

Parameters

None

Applies to

[ComboBox](#)

EndEditing (Grid)

Syntax

EndEditing *row, column*

Description

The EndEditing event is called when a user finishes editing a cell.

Parameters

The following table describes the parameters of the EndEditing event:

Parameters	Description
<i>row</i>	The row of the cell that is being edited
<i>column</i>	The column of the cell that is being edited

Applies to

[Grid](#)

Version

4.1

EndLabelEdit (ListView)

Syntax

EndLabelEdit *item_no*, *new_text*

Description

Item has been edited. Returns False to reject the editing.

Parameters

The following table describes the parameters of the EndLabelEdit event:

Parameters	Description
<i>item_no</i>	Item number of item being edited
<i>new_text</i>	New text for item label

Applies to

[ListView](#)

EndLabelEdit (TreeView)

Syntax

EndLabelEdit *item_id*, *text*, *data*

Description

Editing of label is finishing

Parameters

The following table describes the parameters of the EndLabelEdit event:

Parameters	Description
<i>item_id</i>	ID of item being edited
<i>text</i>	New text of item
<i>data</i>	Number stored against the item

Applies to

[TreeView](#)

EndTrack (Header)

Syntax

EndTrack *item_no*, *width*

Description

Tracking of the header size has finished.

Parameters

The following table describes the parameters of the EndTrack event:

Parameters	Description
<i>item_no</i>	Item number of the header item that is being tracked
<i>width</i>	The new width of the header item

Applies to

[Header](#)

EqualsPressed (Calculator)

Syntax

EqualsPressed

Event Number: 1

Description

The equals key or enter button has been pressed

This is an Active X event so it must be referred to by its number in the DialogBox OnEvent command.

Parameters

None

Applies to

[Calculator](#)

ErrSpace (ComboBox)

Syntax

ErrSpace

Description

The combo box ran out of space when adding an item

Parameters

None

Applies to

[ComboBox](#)

ErrSpace (EditText)

Syntax

ErrSpace

Description

Edit control failed to add some text

Parameters

None

Applies to

[EditText](#)

HScroll (EditText)

Syntax

HScroll

Description

The edit control has been scrolled horizontally

Parameters

None

Applies to

[EditText](#)

ItemChanged (ListView)

Syntax

ItemChanged *item_no, subitem_no, new_state, old_state, changed*

Description

This event is sent when an item has been changing.

Note: The parameters for this event are only available with Windows 98/2000 or later or where Internet Explorer 3 or later has been installed.

Parameters

The following table describes the parameters of the ItemChanged event:

Parameters	Description
<i>item_no</i>	Number of item that has been changed
<i>subitem_no</i>	Number of sub item (column) that has changed if appropriate
<i>new_state</i>	The new state of the item. See table below.
<i>old_state</i>	The state of the item before the change. See the table for the new_state below
<i>changed</i>	Flag specifying what has changed. See Table below.

Vaues for new_state

The following values are added together for the state flags

Value	Description
<i>1</i>	Item has the focus
<i>2</i>	Item is selected
<i>4</i>	Item is displayed as if it has been cut

Value	Description
8	Item is highlighted as a drag and drop target

Vaues for changed

The following values are added together for the changed flag

Value	Description
1	Text has changed
2	Image id has changed
4	User data has changed
8	State has changed

Applies to

[ListView](#)

Example

Detecting the selection has changed

```
* Add event to dialog box create for list view LV
ListView LV,5,4,233,89, LVS_ICON | LVS_ALIGNTOP | WS_BORDER |
WS_TABSTOP
OnEvent LV,"ItemChanged",'Library LV.Lib;OnLvChanged item, new_state,
changed',item, sub_item, new_state, old_state, changed
...
* Subroutine called on change
Sub OnLvChanged(item, new_state,changed)

If changed & 8 Then
    * State has changed
    If new_state & 2 Then
        * Selection has changing
        Print "Selection has changed to item ":item
    EndIf
EndIf
```


EndSub

Item selected on Windows 95

```
* Selection change on any Windows version
*
* During creation of dialog box
ListView FileList, 7,74,267,75, LVS_LIST | LVS_ALIGNTOP |
LVS_SINGLESEL | LVS_SORTASCENDING | WS_BORDER | WS_TABSTOP | WS_GROUP
OnEvent FileList,"ItemChanged","MyDlg_OnSelFileChange"
...
* Set edit control when selection has changed
Sub MyDlg_OnSelFileChange()

sel_item = MyDlg.FileList.GetNextItem(-1, 2)
If sel_item # -1 Then
    sel_text = MyDlg.FileList.GetItemText(sel_item);
    MyDlg.FileName = sel_text
EndIf

EndSub
```

Version

5.0.1 Original version

ItemChanging (ListView)

Syntax

ItemChanging *item_no, subitem_no, new_state, old_state, changed*

Description

This event is sent when an item is changing. Return False to prevent the modification.

Note: The parameters for this event are only available with Windows 98/2000 or later or where Internet Explorer 3 or later has been installed.

Parameters

The following table describes the parameters of the ItemChanging event:

Parameters	Description
<i>item_no</i>	Number of item about to be modified
<i>subitem_no</i>	Number of sub item (column) that has changed if appropriate
<i>new_state</i>	The new state of the item. See table below.
<i>old_state</i>	The state of the item before the change. See the table for the new_state below
<i>changed</i>	Flag specifying what has changed. See Table below.

Vaues for new_state

The following values are added together for the state flags

Value	Description
<i>1</i>	Item has the focus
<i>2</i>	Item is selected
<i>4</i>	Item is displayed as if it has been cut

Value	Description
8	Item is highlighted as a drag and drop target

Vaues for changed

The following values are added together for the changed flag

Value	Description
1	Text has changed
2	Image id has changed
4	User data has changed
8	State has changed

Applies to

[ListView](#)

Example

Detecting the selection is changing

```
* Add event to dialog box create for list view LV
ListView LV,5,4,233,89, LVS_ICON | LVS_ALIGNTOP | WS_BORDER |
WS_TABSTOP
OnEvent LV,"ItemChanging",'Library LV.Lib;OnLvChanging item,
new_state, changed',item, sub_item, new_state, old_state, changed
...
* Subroutine called on change
Sub OnLvChanging(item, new_state,changed)

If changed & 8 Then
    * State has changed
    If new_state & 2 Then
        * Selection is changing
        Print "Selection is changing to item ":item
    EndIf
EndIf
```

EndSub

Version

5.0.1 Original version

ItemClick (Header)

Syntax

ItemClick *item_no*

Description

Header item has been clicked.

Only occurs if the HDS_BUTTONS style has been specified.

Parameters

The following table describes the parameters of the ItemClick event:

Parameters	Description
<i>item_no</i>	Item number that was clicked upon

Applies to

[Header](#)

ItemDbClick (Header)

Syntax

ItemDbClick *item_no*

Description

Double click on a header item.

Only occurs if the HDS_BUTTONS style has been specified.

Parameters

The following table describes the parameters of the ItemDbClick event:

Parameters	Description
<i>item_no</i>	Item number of header item where the double click occurred

Applies to

[Header](#)

ItemExpanded (TreeView)

Syntax

ItemExpanded *action, item_id, state, data*

Description

Item has been expanded

Parameters

The following table describes the parameters of the ItemExpanded event:

Parameters	Description
<i>action</i>	The expansion action. See below:
<i>item_id</i>	Item id
<i>state</i>	State of item
<i>data</i>	The number stored against the item

Vaues for action

Value	Description
<i>1</i>	COLLAPSE
<i>2</i>	EXPAND
<i>3</i>	TOGGLE
<i>16384</i>	EXPANDPARTIAL
<i>32768</i>	COLLAPSERESET

Applies to

[TreeView](#)

ItemExpanding (TreeView)

Syntax

ItemExpanding *action, item_id, state, data*

Description

Item is expanding

Parameters

The following table describes the parameters of the ItemExpanding event:

Parameters	Description
<i>action</i>	The expanding action. See below.
<i>item_id</i>	ID of the item
<i>state</i>	The items current state
<i>data</i>	The number stored against the data

Vaues for action

Value	Description
<i>1</i>	COLLAPSE
<i>2</i>	EXPAND
<i>3</i>	TOGGLE
<i>16384</i>	EXPANDPARTIAL
<i>32768</i>	COLLAPSERESET

Applies to

[TreeView](#)

KeyDown

Syntax

KeyDown *virtual_key_code*

Description

key was pressed

Parameters

The following table describes the parameters of the KeyDown event:

Parameters	Description
<i>virtual_key_code</i>	Key code of the key that was pressed

Applies to

[ListView](#), [TabControl](#), [TreeView](#)

KeyUp (Calculator)

Syntax

KeyUp *key, shift_state*

Event Number: -604

Description

A key has been released.

This is an Active X event so it must be referred to by its number in the DialogBox OnEvent command.

Parameters

The following table describes the parameters of the KeyUp event:

Parameters	Description
<i>key</i>	The key code of the released key
<i>shift_state</i>	The shift state of the key when it was released

Applies to

[Calculator](#)

KillFocus

Syntax

KillFocus

Description

Control has lost the focus

Button controls only receive this event if they have the BS_NOTIFY style.

Parameters

None

Applies to

[CheckBox](#), [CheckListBox](#), [ColorButton](#), [ComboBox](#), [DateTime](#), [EditText](#), [ListBox](#), [ListView](#), [PushButton](#), [RadioButton](#), [TreeView](#)

MaxText (EditText)

Syntax

MaxText

Description

The maximum amount of text allowed for the edit control has been exceeded

Parameters

None

Applies to

[EditText](#)

ModifyCell (Grid)

Syntax

ModifyCell *row, column*

Description

The ModifyCell event is called when a user is modifying a cell.

It is only called for cells which are edited and the result put back in the cell not cells which are immediately changed.

E.g. This would be called for a cell with a drop down list but not for a check box.

Parameters

The following table describes the parameters of the ModifyCell event:

Parameters	Description
<i>row</i>	The row number of the cell being modified
<i>column</i>	The column number of the cell being modified

Applies to

Grid

Version

4.1 Original

OutOfMemory

Syntax

`OutOfMemory`

Description

The control could not allocate enough memory.

Parameters

None

Applies to

[TreeView](#)

Return

Syntax

Return

Description

Return key was pressed

Parameters

None

Applies to

[ListView](#), [TreeView](#)

RightClick

Syntax

RightClick

Description

Right mouse button clicked upon the control

Parameters

None

Applies to

[TreeView](#)

RightClick (ListView)

Syntax

RightClick *item_no*, *subitem_no*, *x*, *y*

Description

Right mouse click.

Parameters

The following table describes the parameters of the RightClick event:

Parameters	Description
<i>item_no</i>	The item number
<i>subitem_no</i>	The column number
<i>x</i>	x co-ordinate
<i>y</i>	y co-ordinate

Applies to

[ListView](#)

RightDblClick

Syntax

RightDblClick

Description

Right mouse button was double clicked upon the control

Parameters

None

Applies to

[TreeView](#)

RightDbClick (ListView)

Syntax

RightDbClick *item_no*, *subitem_no*, *x*, *y*

Description

Right mouse double click.

Parameters

The following table describes the parameters of the RightDbClick event:

Parameters	Description
<i>item_no</i>	Item number
<i>subitem_no</i>	Column number
<i>x</i>	x co-ordinate
<i>y</i>	y co-ordinate

Applies to

[ListView](#)

Scroll (TrackBar)

Syntax

Scroll

Description

The slider has been moved by the user

Parameters

None

Applies to

[TrackBar](#)

SelCancel (ListBox)

Syntax

SelCancel

Description

The selection from the list box has been cancelled

List box must have the LBS_NOTIFY style to receive this event.

Parameters

None

Applies to

[CheckListBox](#), [ListBox](#)

SelChange (ComboBox)

Syntax

SelChange

Description

The selection has changed

Parameters

None

Applies to

[ComboBox](#)

SelChange (ListBox)

Syntax

SelChange

Description

The selection is about to change.

List box must have the LBS_NOTIFY style to receive this event.

Parameters

None

Applies to

[CheckListBox](#), [ListBox](#)

SelChanged (TabControl)

Syntax

SelChanged

Description

The selection has changed.

Parameters

None

Applies to

[TabControl](#)

SelChanged (TreeView)

Syntax

SelChanged *action, old_item_id, old_item_state, old_item_data, new_item_id, new_item_state, new_item_data*

Description

Selected item has changed

Parameters

The following table describes the parameters of the SelChanged event:

Parameters	Description
<i>action</i>	Cause of selection. See below.
<i>old_item_id</i>	ID of item selection has changed from
<i>old_item_state</i>	state of item selection has changed from
<i>old_item_data</i>	Number stored against the item selection has changed from
<i>new_item_id</i>	ID of item selection has changed to
<i>new_item_state</i>	state of item selection has changed to
<i>new_item_data</i>	The number stored against the item the selection has changed to

Applies to

[TreeView](#)

SelChanging (TabControl)

Syntax

`SelChanging`

Description

The selection is changing.

Parameters

None

Applies to

[TabControl](#)

SelChanging (TreeView)

Syntax

SelChanging *action, old_item_id, old_item_state, old_item_data, new_item_id, new_item_state, new_item_data*

Description

Selected item is changing

Parameters

The following table describes the parameters of the SelChanging event:

Parameters	Description
<i>action</i>	Cause of selection. See below.
<i>old_item_id</i>	ID of item selection is changing from
<i>old_item_state</i>	state of item selection is changing from
<i>old_item_data</i>	Number stored against the item selection is changing from
<i>new_item_id</i>	ID of item selection is changing to
<i>new_item_state</i>	state of item selection is changing to
<i>new_item_data</i>	The number stored against the item the selection is changing to

Applies to

[TreeView](#)

SelEndCancel (ComboBox)

Syntax

`SelEndCancel`

Description

The selection from the list box has been cancelled

Parameters

None

Applies to

[ComboBox](#)

SelEndOK (ComboBox)

Syntax

SelEndOK

Description

A selection from the list box has been chosen

Parameters

None

Applies to

[ComboBox](#)

SetFocus

Syntax

SetFocus

Description

The control has received the input focus.

Button controls only receive this event if they have the BS_NOTIFY style.

Parameters

None

Applies to

[CheckBox](#), [CheckListBox](#), [ColorButton](#), [ComboBox](#), [DateTime](#), [EditText](#), [ListBox](#), [ListView](#), [PushButton](#), [RadioButton](#), [TreeView](#)

Related Script Commands

DialogBox OnEvent, DialogBox Default

Version

4.1.0 Added to EditText control

Start (Animate)

Syntax

Start

Description

AVI file has started playing

Parameters

None

Applies to

[Animate](#)

StartEditing (Grid)

Syntax

StartEditing *row, column*

Event Number: 1

Description

Called when a user starts editing a cell.

Use DialogBox Reply *dlg_name*, False to stop the modification.

This is an Active X event so it must be referred to by its number in the DialogBox OnEvent command.

Parameters

The following table describes the parameters of the StartEditing event:

Parameters	Description
<i>row</i>	The row of the cell that is being edited
<i>column</i>	The column of the cell that is being edited

Applies to

[Grid](#)

Stop (Animate)

Syntax

Stop

Description

AVI file has stopped playing.

Parameters

None

Applies to

[Animate](#)

Track (Header)

Syntax

Track *item_no*, *width*

Description

The header size if being tracked.

Use DialogBox Reply to return True to finish the tracking.

Parameters

The following table describes the parameters of the Track event:

Parameters	Description
<i>item_no</i>	Item number of the header item being tracked
<i>width</i>	Current track width

Applies to

[Header](#)

Update (EditText)

Syntax

Update

Description

The text has been changed (received before the screen update)

Parameters

None

Applies to

[EditText](#)

VScroll (EditText)

Syntax

VScroll

Description

The edit control has been scrolled vertically

Parameters

None

Applies to

[EditText](#)

ValidateCell (Grid)

Syntax

ValidateCell *row, column*

Event Number: 2

Description

Called when a cells contents needs validation.

Use DialogBox Reply *dlg_name*, False to stop the modification.

To get the value entered into the cell to be validated use the GetCurrentText method.

This is an Active X event so it must be referred to by its number in the DialogBox OnEvent command.

Parameters

The following table describes the parameters of the ValidateCell event:

Parameters	Description
<i>row</i>	The row that needs validating
<i>column</i>	The column that needs validating

Applies to

Grid

Related Script Commands

DialogBox Reply

Reference - Script Menu Options

The following section describes the names of the menu options that are built into the application.

Use the Dialog command to specify which menu option is going to be updated/used. After this command the values on the dialog can be modified using the set/store/configure and update commands and the dialog can be shown with the show command, run without displaying the dialog with the invoke command. The current values of the parameters can be interrogated with the Get function.

There are three kinds of dialog:

Global - The variables can be set and retrieved without the prior use of the Dialog command. Most global variables have their values saved in the session file.

Local - The dialog name must be specified before these are used.

Application - These variables are read when the session first starts up and are initially the same for all sessions.

EditCopy

Type

Local

Default Menu Location

Edit Copy

Description

Copy selected text to the clipboard

Variables

This menu option has no associated variables

EditCopySpecial

Type

Local

Default Menu Location

Edit Copy Special

Description

Copys the current seletion in the format and at the location specified i the Edit Copy Special To menu.

Variables

This menu option has no assosiated variables

EditCopyTo

Type

Local

Default Menu Location

Edit Copy Special To

Description

This dialog sets the parameters to use in the Edit Copy Special menu option.

Variables

The following table describes the variables available for the EditCopyTo menu option:

Variable	Description
<i>CopyTo</i>	Destination for the next copy special.
<i>Format</i>	The format to put the copied data in
<i>FileName</i>	The name of the local file. Used when the format variable is set to File or Edit only.
<i>ReverseColors</i>	Reverse all the colors when createing a bitmap
<i>BlackAndWhite</i>	Set the text to ba balck and white only when creating a bitmap

Values for CopyTo

Must be one of the following literal values

Value	Description
<i>Clipboard</i>	Windows clipboard
<i>Printer</i>	Currently assigned printer

Value	Description
<i>File</i>	PC File whose name is specified by the FileName parameter
<i>Edit</i>	PC File whose name is specified by the FileName parameter is created and then opened in the current editor

Values for Format

Must be one of the following values:

Value	Description
<i>Text</i>	Plain text
<i>Table</i>	Text parsed into columns with tabs separating each column
<i>Bitmap</i>	Windows bitmap
<i>HTML</i>	A web page

Version

4.0.3 Reverse Colors and BlackAndWhite variables added

EditPaste

Type

Local

Default Menu Location

Edit Paste

Description

Paste text from the clipboard to the host

Variables

This menu option has no associated variables

EditPlay

Type

Local

Default Menu Location

Edit Play

Description

Play back a file or recording to the host or screen.

Variables

The following table describes the variables available for the EditPlay menu option:

Variable	Description
<i>FileName</i>	The name of the file to play back
<i>Local</i>	True to send the file to the local emulation. False sends the file to the remote host.
<i>Pause</i>	True to pause a local play back when a form feed is encountered
<i>Text</i>	True to only send text characters to the host. i.e. Strip out all control characters except carriage return.

Related Script Commands

Play Start, Play Stop, Play Abort

EditRecord

Type

Local

Default Menu Location

Edit Record

Description

Turns on and off recording.

If this menu option is displayed using the Show command the dialog box will be displayed allowing the user to set up the recording.

If this menu option is displayed using the Invoke command the recording will start with no user intervention.

The recording is turned off by a second invokeation of the menu option with Show or Invoke.

Variables

The following table describes the variables available for the EditRecord menu option:

Variable	Description
<i>RecordTo</i>	The destination for the recording. Must be "File" or "Printer".
<i>Format</i>	What is to be recorded. See below
<i>PrintDirect</i>	True to send the recorded bytes straight to the printer with no formatting.
<i>FileName</i>	The file name to use when recording to a file

Values for Format

The record format must be one of the following literal values

Value	Description
<i>"RawData"</i>	The bytes that come from the host.
<i>"Screens"</i>	The text of each screen. Recorded just before a new screen is shown.
<i>"Keystrokes"</i>	The keys pressed
<i>"ControlCodes"</i>	A display of what came from the host and how it was interpreted by the emulation.

EditSearch

Type

Local

Default Menu Location

Edit Search

Description

This menu options searches the screen memory for the specified text and selects it and scrolls it into view if it is found.

The search starts from the last selection.

Variables

The following table describes the variables available for the EditSearch menu option:

Variable	Description
<i>SearchFor</i>	The text to search for
<i>MatchCase</i>	True if the text found must have the same case as the search for text.
<i>WholeWord</i>	True if the search text must match a whole word on the screen.
<i>Direction</i>	The direction for the search. Must be "Up" or "Down"

EditSelectAll

Type

Local

Default Menu Location

Edit Select All

Description

Select all of the screen memory

Variables

This menu option has no associated variables

EditSelectScreen

Type

Local

Default Menu Location

Not on menu

Description

Selects the current screen

Variables

This menu option has no associated variables

EditSelectWindow

Type

Local

Default Menu Location

Edit Select Window

Description

Selects the area of screen showing in the session window.

Variables

This menu option has no associated variables

FileAnother

Type

Local

Default Menu Location

File Another

Description

Opens a new session from a configuration file or script.

Variables

The following table describes the variables available for the FileAnother menu option:

Variable	Description
<i>FileName</i>	The name of the file to open

FileDelete

Type

Local

Default Menu Location

Not on the menu

Description

Deletes the specified file

Variables

The following table describes the variables available for the FileDelete menu option:

Variable	Description
<i>FileName</i>	The name of the file to delete

Related Script Commands

File Delete

FileEdit

Type

Local

Default Menu Location

File Edit

Description

Edits a text file using the text editor setup in Setup Preferences

Variables

The following table describes the variables available for the FileEdit menu option:

Variable	Description
<i>FileName</i>	The name of the text file to edit

Related Script Commands

Run

FileExit

Type

Local

Default Menu Location

File Exit

Description

Exits the current session.

If this is run with the Show command it will display the confirmation dialog box if it is configured in Setup Application.

If this is run with the Invoke command the session will exit with no prompting.

Variables

This menu option has no associated variables

FileExitAll

Type

Local

Default Menu Location

File Exit All

Description

Exits all the running sessions.

Variables

This menu option has no associated variables

FileOpen

Type

Local

Default Menu Location

File Open

Description

Opens a configuration file or script.

Variables

The following table describes the variables available for the FileOpen menu option:

Variable	Description
<i>FileName</i>	The name of the file to open.

FilePrint

Type

Local

Default Menu Location

File Print

Description

Prints a text file

Variables

The following table describes the variables available for the FilePrint menu option:

Variable	Description
<i>FileName</i>	The name of the text file to print

FilePrinterSetup

Type

Global

Default Menu Location

File Printer Setup

Description

Sets up the printer and printer options for printing from this application.

Variables

The following table describes the variables available for the FilePrinterSetup menu option:

Variable	Description
<i>Printer</i>	The name of the printer to use. Set to "" to use the default Windows printer.
<i>PrinterFont</i>	The font to use for the printout. Specify as the font name, point size. Setting this parameter to "" will use the default font for the currently selected printer.
<i>PrinterOrientation</i>	Set the orientation for the printout. Must be "Portrait" or "Landscape"
<i>PrinterCharMap</i>	True to use the character mapping from the emulation when printing.
<i>PrinterDirect</i>	True to send the bytes for the printout directly to the printer without any processing when slave printing.
<i>PrinterPrefix</i>	The bytes (in backslash) format to send to the printer at the beginning of a printout that is using direct printing.

FileSaveAs

Type

Local

Default Menu Location

File Save As

Description

Save the current configurations with the specified file name.

Variables

The following table describes the variables available for the FileSaveAs menu option:

Variable	Description
<i>FileName</i>	File name to save the current configuration with

HelpAbout

Type

Local

Default Menu Location

Help About

Description

Show the about this application dialog box

Variables

This menu option has no associated variables

HelpIndex

Type

Local

Default Menu Location

Help Contents

Description

Show the contents page of the applications help.

Variables

This menu option has no associated variables

HelpOnHelp

Type

Local

Default Menu Location

Help Help On Help

Description

Shows the Windows on line help on how to use the help system

Variables

This menu option has no associated variables

HelpStatus

Type

Local

Default Menu Location

Help Status

Description

Shows a dialog that shows the display and keyboard status and allows the user to change them.

Variables

This menu option has no associated variables

HelpSupportLog

Type

Local

Default Menu Location

Help Support Information

Description

Shows a dialog box with various information that may be useful when contact customer support.

Variables

This menu option has no associated variables

RunBridgeCopy

Type

Local

Default Menu Location

Run Bridge Copy File

Description

Copy files from one session to another

Variables

The following table describes the variables available for the RunBridgeCopy menu option:

Variable	Description
<i>SourceFile</i>	The name of the file holding the source data.
<i>SourceAccount</i>	The name of the account that contains the file. If this is left blank, wIntegrate uses the current account.
<i>SourceItems</i>	IDs of source items to copy. Use * to specify all items, or use a SELECT, GET-LIST or similar statement.
<i>SourceFields</i>	Dictionary names specifying the fields to take from the source items. Use * to specify all fields.
<i>Target</i>	The session name from Setup Preferences.
<i>TargetFile</i>	The file on the target machine.
<i>TargetAccount</i>	The name of the account on the target machine that contains the file.

Variable	Description
<i>TargetItems</i>	The item IDs to use on the target machine. Leave this field blank if you are using the original item IDs. If you do not specify enough item IDs for the incoming records, wIntegrate uses original item IDs.
<i>TargetFields</i>	The dictionary names containing attribute numbers specifying the destination for the incoming fields. These must not specify an attribute number of 0 (zero), otherwise the file transfer is aborted with an error message.
<i>Overwrite</i>	"Yes", "No" or "Combine"
<i>Inform</i>	True or false flag. If true, the user must acknowledge a message when the transfer is completed.
<i>Timeout</i>	Number of seconds to wait for a response. If not set, the default is 5.
<i>Retries</i>	The maximum number of errors allowed. If not set, the default is 3.
<i>AutoExit</i>	True or false flag. If true, the File Transfer Monitor box disappears at the end of the transfer without the user having to click on the Exit button.

RunClearBackPages

Type

Local

Default Menu Location

Run Clear Back Pages

Description

Clears the screen memory of all of the back pages

Variables

This menu option has no associated variables

RunExportFile

Type

Local

Default Menu Location

Run Export File

Description

Export a file to the host

Variables

The following table describes the variables available for the RunExportFile menu option:

Variable	Description
<i>DosFile</i>	File to be exported to the host
<i>Account</i>	Host account name for the export. Set to "" for the current account
<i>File</i>	Name of the host file for the export
<i>Items</i>	Items or selection to use for the export
<i>Fields</i>	Fields to export
<i>Format</i>	File format to produce for the export
<i>Inform</i>	Show a dialog box when the export has finished
<i>Timeout</i>	Time out in seconds
<i>Retries</i>	Retries to make before finishing the transfer
<i>AutoExit</i>	Set to true to automatically close the file transfer monitor when the transfer has completed

Variable	Description
<i>Translate</i>	Set import formats to use the translation table, "None", "Ascii" or "All"
<i>Translation</i>	Set the translation table
<i>Overwrite</i>	Overwrite local file: "Yes", "No", "Combine"
<i>UseFormattingInformation</i>	Use conversion, length and justification information from host dictionary items
<i>MultipleDisks</i>	True or false flag. If true, the system prompts for multiple disks.

Version

4.1 Added new dictionary options

RunImportFile

Type

Local

Default Menu Location

Run Import File

Description

Run an import file transfer

Variables

The following table describes the variables available for the RunImportFile menu option:

Variable	Description
<i>Account</i>	Host account name for the import. Set to "" for the current account
<i>File</i>	Name of the host file for the import
<i>Items</i>	Items or selection to use for the import
<i>Fields</i>	Fields to import
<i>DosFile</i>	File to be imported to on the local machine
<i>Format</i>	File format to produce for the import
<i>Inform</i>	Set to true to show a message box when the import has finished
<i>Timeout</i>	Time out in seconds
<i>Retries</i>	The number of retries to recover from an error before aborting
<i>AutoExit</i>	Set to true to automatically close the file transfer monitor when the transfer has completed

Variable	Description
<i>Translate</i>	Set import formats to use the translation table, "None", "Ascii" or "All"
<i>Mode</i>	Type of import to use: "Normal" - Import straight data fields with error checking, "Capture" - Import data from a host listing (no error checking), "Reformat" - Create work file on host and import with error checking
<i>FieldDescriptions</i>	Import field descriptions as first record of import
<i>NumberConversion</i>	Turn on automatic number conversion during import
<i>NumberSeparator</i>	Separator character for thousands used in number conversion
<i>NumberCurrency</i>	Currency characters used in number conversion
<i>NumberDecimal</i>	Decimal character used in number conversion
<i>ExplodeValues</i>	Explode multi-value fields for normal/reformat mode imports
<i>RepeatValues</i>	Repeat single-value fields for items containing multi-values in normal/reformat mode imports
<i>UseFormattingInformation</i>	Use conversion, length and justification information from host dictionary items
<i>LeftJustifiedIsText</i>	Force left justified fields to be imported as text/string type
<i>RightJustifiedIsNumeric</i>	Force right justified fields to be imported as numbers

Version

4.1 Added new dictionary options

RunKermitFinish

Type

Local

Default Menu Location

Run Server Kermit Command Finish

Description

Shuts down the remote Kermit server

Variables

This menu option has no associated variables

RunKermitGetFile

Type

Local

Default Menu Location

Run Kermit Server Command Get File

Description

Get a file from a Kermit Server

Variables

The following table describes the variables available for the RunKermitGetFile menu option:

Variable	Description
<i>DosFile</i>	The file to be imported to on the local machine
<i>AutoExit</i>	Set to true to automatically close the file transfer monitor when the transfer has completed
<i>Timeout</i>	Set time out for kermit and x modem
<i>Retries</i>	Set retries for kermit and x modem
<i>Overwrite</i>	Overwrite the local file
<i>KermitTimeout</i>	Set timeout for kermit
<i>KermitRetries</i>	Set retries for kermit
<i>KermitPacketLength</i>	Set packet length for Kermit
<i>File</i>	File to be imported on the remote system

RunKermitSendFile

Type

Local

Default Menu Location

Run Kermit Server Command Send File

Description

Send a file to a Kermit Server

Variables

The following table describes the variables available for the RunKermitSendFile menu option:

Variable	Description
<i>DosFile</i>	The file to export to on the remote machine
<i>AutoExit</i>	Set to true to automatically close the file transfer monitor when the transfer has completed
<i>Timeout</i>	Set time out for kermit and x modem
<i>Retries</i>	Set retries for kermit and x modem
<i>Overwrite</i>	Overwrite the remote file
<i>KermitTimeout</i>	Set timeout for kermit
<i>KermitRetries</i>	Set retries for kermit
<i>PreserveCase</i>	Preserve the case of the file name

RunKermitServer

Type

Local

Default Menu Location

Run Kermit Server

Description

Starts a Kermit server on the session.

Variables

This menu option has no associated variables

RunProgram

Type

Local

Default Menu Location

Run Program

Description

Run an external application

Variables

The following table describes the variables available for the RunProgram menu option:

Variable	Description
<i>FileName</i>	The name of the program file to run. This may include a path specification.
<i>Arguments</i>	The arguments to append to the program name as the program executes.

RunReceiveFile

Type

Local

Default Menu Location

Run Receive Binary File

Description

Import a binary file

Variables

The following table describes the variables available for the RunReceiveFile menu option:

Variable	Description
<i>Protocol</i>	This sets the protocol for the binary file transfer (see table below)
<i>DosFile</i>	The file to be imported to on the local machine
<i>AutoExit</i>	Set to true to automatically close the file transfer monitor when the transfer has completed
<i>Timeout</i>	Set time out for kermit and x modem
<i>Retries</i>	Set retries for kermit and x modem
<i>Overwrite</i>	Overwrite the local file
<i>KermitRetries</i>	Set retries for kermit
<i>KermitTimeout</i>	Set timeout for kermit
<i>KermitPacketLength</i>	Set packet length for Kermit
<i>XModemTimeout</i>	Set timeout of x modem
<i>XModemRetries</i>	Set retries for x modem

Variable	Description
<i>XModemChecksum</i>	Set checksum type for x modem (1 = checksum, 2 = CRC)
<i>ZModemTimeout</i>	Set timeout of z modem
<i>ZModemRetries</i>	Set retries for z modem
<i>ZModemChecksum</i>	Set checksum type for z modem (2 = 2 byte CRC, 4 = 4 byte CRC)

Values for Protocol

Value	Description
<i>Kermit</i>	Kermit
<i>Xmodem</i>	Xmodem
<i>YModem</i>	Ymodem
<i>ZModem</i>	Zmodem
<i>Xmodem 1K</i>	Xmodem with 1k buffer and CRC Checksum. This is for compatability with version 3.x only.

RunResizeWindow

Type

Local

Default Menu Location

Run Resize Window

Description

Resizes the current session window so that it shows all of the screen.

Variables

This menu option has no associated variables

RunRestartPort

Type

Local

Default Menu Location

Run Restart Port

Description

Reset the screen and port

Variables

This menu option has no associated variables

RunScript

Type

Local

Default Menu Location

Run Script

Description

The Run Script menu options is used to run a script file.

Variables

The following table describes the variables available for the RunScript menu option:

Variable	Description
<i>FileName</i>	The name of the script file to run

Related Script Commands

Script

RunSendFile

Type

Local

Default Menu Location

Run Send Binary File

Description

Send a binary file using Kermit, XModem, YModem, or ZModem.

Variables

The following table describes the variables available for the RunSendFile menu option:

Variable	Description
<i>Protocol</i>	This sets the protocol for the binary file transfer (see table below)
<i>DosFile</i>	The file to be exported from the local machine
<i>AutoExit</i>	Set to true to automatically close the file transfer monitor when the transfer has completed
<i>Timeout</i>	Set time out for kermit and x modem
<i>Retries</i>	Set retries for kermit and x modem
<i>KermitTimeout</i>	Set timeout for kermit
<i>KermitRetries</i>	Set retries for kermit
<i>PreserveCase</i>	Preserve case of file name
<i>XModemTimeout</i>	Set timeout of x modem
<i>XModemRetries</i>	Set retries for x modem
<i>XModemPacketLength</i>	Packet length for x modem

Variable	Description
<i>XModemPadCharacter</i>	Pad character for transfer
<i>ZModemTimeout</i>	Set timeout of z modem
<i>ZModemRetries</i>	Set retries for z modem
<i>ZModemWindowSize</i>	Window size for z modem

Values for Protocol

Value	Description
<i>Kermit</i>	Kermit
<i>Xmodem</i>	Xmodem
<i>YModem</i>	Ymodem
<i>ZModem</i>	Z modem
<i>Xmodem 1k</i>	X Modem with 1k buffer and CRC Checksum. This is for compatability with version 3.x only.

RunSpoolFile

Type

Local

Default Menu Location

Run Spool File

Description

Send a PC file to the host spooler

Variables

The following table describes the variables available for the RunSpoolFile menu option:

Variable	Description
<i>DOSFile</i>	The name of the file to print, with the path specification.
<i>MultipleDisks</i>	True or false flag. If true, the system prompts for multiple disks.
<i>Format</i>	The DOS file format. If not set, wIntegrate takes the format from the DOS file extension.
<i>FieldWidths</i>	The column widths for the printed report.
<i>FieldSeparator</i>	The character to separate columns on the report.
<i>DataPerLine</i>	"OneRecord" or "OneField"
<i>Inform</i>	True or false flag. If true, a box appears when the transfer completes.
<i>Timeout</i>	Number of seconds to wait for a response. If not set, the default is 5.

Variable	Description
<i>Retries</i>	The maximum number of errors allowed. If not set, the default is 3.
<i>AutoExit</i>	True or false flag. If true, the File Transfer Monitor box disappears at the end of the transfer without clicking on the Exit button.
<i>Translate</i>	"None", "ASCII" or "All"
<i>Translation</i>	The host and PC strings are in double-quotation marks and separated by a comma.

ScriptInfo

Type

Local

Default Menu Location

Not on a menu

Description

This menu options shows information about the items maintained by the script engine that exist outside of an individual script.

This includes the global variables, dialog boxes, menus and cached libraries.

Variables

This menu option has no associated variables

ScrollAcrossWidth

Type

Local

Default Menu Location

Run Scroll Across Width

Description

Scroll the columns across by the width of the screen

Variables

This menu option has no associated variables

ScrollBackWidth

Type

Local

Default Menu Location

Run Scroll Back Width

Description

Scroll the columns back by the width of the screen

Variables

This menu option has no associated variables

ScrollDownLine

Type

Local

Default Menu Location

Run Scroll Down Line

Description

Scrolls down one line

Variables

This menu option has no associated variables

ScrollEndPage

Type

Local

Default Menu Location

Run Scroll End Page

Description

Scrolls to the last page of screen memory.

Variables

This menu option has no associated variables

ScrollLeftColumn

Type

Local

Default Menu Location

Run Scroll Left Column

Description

Scroll screen left by one column

Variables

This menu option has no associated variables

ScrollNextPage

Type

Local

Default Menu Location

Run Scroll Next Page

Description

Scrolls forward one page

Variables

This menu option has no associated variables

ScrollPrevPage

Type

Local

Default Menu Location

Run Scroll Prev Page

Description

Scroll up one page

Variables

This menu option has no associated variables

ScrollRightColumn

Type

Local

Default Menu Location

Run Scroll Right Column

Description

Scroll one column to the right

Variables

This menu option has no associated variables

ScrollTopPage

Type

Local

Default Menu Location

Run Scroll Top Page

Description

Scrolls up to the first page of screen memory

Variables

This menu option has no associated variables

ScrollUpLine

Type

Local

Default Menu Location

Run Scroll Up Line

Description

Scroll up one line

Variables

This menu option has no associated variables

SetupApplication

Type

Application

Default Menu Location

Setup Application

Description

This menu options sets up application preferences that apply to all the running sessions.

The values from this dialog are stored in the Windows Registry in the Local version and to a user specific file in the Thin Client version.

Setting the values of the variables in this dialog will only effect the current session unless Invoke SetupApplication is run, this causes the settings to update the registry.

Variables

The following table describes the variables available for the SetupApplication menu option:

Variable	Description
<i>ConfirmExit</i>	True to show the confirm exit dialog box
<i>SaveChanges</i>	True to set the save changes when the session is exited
<i>SnapToSize</i>	True to automatically resize the program so an exact number of screen columns and lines are shown
<i>ConnectMessage</i>	Location to display messages from the current connection. Must be "Screen", "Status Bar" or "Dialog"

Variable	Description
<i>ShowSessionCopyDialog</i>	True to show the dialog box to warn the user when a session is being opened for a second time.
<i>Editor</i>	The editor to be used to edit text
<i>BitmapEditor</i>	The editor used to edit bitmaps
<i>HTMLEditor</i>	The editor used to edit HTML documents
<i>PreSessionOpenScript</i>	Script to run before session script
<i>PostSessionOpenScript</i>	Script to run after the session script
<i>StartUpSessions</i>	A list of all the sessions that are set up to start up when the application is started.
<i>Language</i>	Sets the language to display the dialogs/menus and messages. The new language will not take full effect until the application is closed and restarted. This setting applies to the Local version only.
<i>SurrogateFont</i>	Set the name of the font used to display utf16 surrogate characters. This setting applies to the Local version only.

Version

5.0 Invoke saves settings. StartUpSessions variable added

5.2 Language/SurrogateFont variables added

SetupCharacter

Type

Global

Default Menu Location

Setup Character

Description

Sets up the font to use and character processing for the screen

Variables

The following table describes the variables available for the SetupCharacter menu option:

Variable	Description
<i>Font</i>	Any valid fixed pitch font name, as displayed in the Fonts box, followed by the width and depth, e.g., Font: IBSfont, 6, 11
<i>ExtraLineSpace</i>	The number of pixels extra spacing to place between lines
<i>InvalidCharacters</i>	How to display characters with ascii code less than 32 and greater than 126. Must be one of: "Display", "DoNotDisplay", "DecimalDisplay".
<i>AutoScaling</i>	Set to true to automatically scale the font to fill as much of the session window as possible.
<i>DrawGraphics</i>	Set to true to draw the graphics characters. Use this when not using the "IBSFont" or when you want vertical lines to join when extra line spacing is being used.

SetupColors

Type

Global

Default Menu Location

Setup Colors

Description

This menu options specifies how the colors and monochrome effects are displayed on the main screen.

This menu option can also be spelled SetupColours.

Variables

The following table describes the variables available for the SetupColors menu option:

Variable	Description
<i>Effect_<name></i>	Set the colors and styles to use for the effect. The effect names are Normal, Dim, Reverse, Flash, Underline, Bold and Secret. Concatenating multiple names together sets the properties of the combined effects. The value is made up of four fields foreground color, background color, style and combine. See below
<i>Color_<number></i>	Set the color and style to use for a screen color. <number> is the color number from 0 to 15. The value is specified in exactly the same fashion as for the Effect_<name> parameter.

Variable	Description
<i>UseStyle</i>	Set to true to enable the display of the styles set with the Effect_<name> or Color_<number> parameter.
<i>BackgroundImage</i>	The file name of the image to use as the background to the terminal screen. If a full path is not specified the file is assumed to be in the BackGrnd folder. Use "(none)" or "" for no image.
<i>BackgroundDisplay</i>	How to display the background image: "Stretch" to stretch the image to fill the entire background. "Tile" to repeat the image to fill the entire background
<i>EffectIgnoreBold</i>	Version 3 compatibility flag
<i>EffectIgnoreReverse</i>	Version 3 compatibility flag
<i>EffectMode</i>	Version 3 compatibility flag. True to use SB effect mode.
<i>DefaultColors</i>	Set foreground and background colors to use as the default. Specified as a foreground (0 to 15), background (0 to 15).
<i>DefaultColours</i>	Same as DefaultColors
<i>EraseUsingWriteAttrs</i>	Set to true to erase screen with current write attributes, false uses the emulations erase attributes.
<i>DisplayStartField</i>	Specifies how the attribute cell in field attributing emulations is displayed. See table below.
<i>InitialAutoWrap</i>	Set the initial state of autowrap. 0 - on, 1 - off, 2 - delayed
<i>RealFlashing</i>	Version 3 compatability flag. Sets/Unsets real flashing for all the effects with the flash attribute.

Values for Effect_<name>

The four comma separated parts of the effects value are

Value	Description
<i>foreground</i>	The color of the display text
<i>background</i>	The color of background of the text
<i>style</i>	The style is: 0 normal, 1 Raised, 2 Sunken, 3 Text box, 4 border, 5 2 pixel raised, 6 2 pixel sunken. Add 32 to make lines of the effect join up
<i>Combine</i>	How to combine the color with what is on the screen. Real effects: 1 Underline, 2 Flash, 4 Reverse, 5 Bold.

Version

4.0.3 BackgroundImage and BackgroundDisplay variables added

SetupCommunications

Type

Global

Default Menu Location

Setup Communications

Description

This menu option allows you to choose the way to communicate with the host, set up any parameters required and open and close the connection.

The variables for each connection type is described in a separate section.

Variables

The following table describes the variables available for the SetupCommunications menu option:

Variable	Description
<i>Port</i>	The name of the communications port. Can be "PicLan", "Serial" or "Windows Sockets". For compatability with previous versions of the application you can also use "COM1" to "COM8" for serial communications on the specific port.
<i>PortOpen</i>	Set to true to open the port, false to close the port.

SetupCommunications (PicLan)

Type

Global

Default Menu Location

Setup Communications (Setup... button)

Description

This menu options allows you to set up the parameters for PicLan communications.

The variables here are only available when the Port has been set to PicLan in the main Setup Communications menu option.

Variables

The following table describes the variables available for the SetupCommunications (PicLan) menu option:

Variable	Description
<i>Host</i>	The host name as advertised on the network by the Pick system
<i>PortNumber</i>	The connection number or -1 to allow the Pick system to decide which port number to use.

SetupCommunications (Serial)

Type

Global

Default Menu Location

Setup Communications (Setup... button)

Description

This menu options allows you to set up the parameters for Serial communications.

The variables here are only available when the Port has been set to Serial in the main Setup Communications menu option.

Variables

The following table describes the variables available for the SetupCommunications (Serial) menu option:

Variable	Description
<i>PortName</i>	The name of the serial port to use: "COMn" where n is the port number. e.g. COM1, COM2
<i>BaudRate</i>	Set the baud rate for the port. Can be 110, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600.
<i>Parity</i>	Set the parity. Can be "None", "Odd", "Even", "Mark", "Space"
<i>DataBits</i>	Set the number of data bits. Can be 6,7 or 8.
<i>StopBits</i>	Set the stop bits. Can be "1", "1.5" or "2".
<i>Flow</i>	Set the flow control to use: "None", "Software", "Hardware"

Variable	Description
<i>ZeroTopBit</i>	Set to true to clear the top bit of every character received from the host.
<i>ParityCheck</i>	True to enable parity checking on the communications
<i>ReportErrors</i>	True to report any errors encountered during communications.
<i>TransmitDelay</i>	The number of milliseconds to delay between transmitting blocks of characters to the host. Set to 0 (recommended) to disable this feature.
<i>TransmitBlockSize</i>	The maximum number of characters to send to the host in one block. Set to 0 (recommended) for no restriction.

SetupCommunications (Windows Sockets)

Type

Global

Default Menu Location

Setup Communications (Setup... button)

Description

This menu options allows you to set up the parameters for Windows Sockets communications.

The variables here are only available when the Port has been set to Windows Sockets in the main Setup Communications menu option.

Variables

The following table describes the variables available for the SetupCommunications (Windows Sockets) menu option:

Variable	Description
<i>Host</i>	Set the host name or IP address for the communications
<i>TelnetPort</i>	Set the telnet port number. Usually 23.
<i>Telnet</i>	Set to true to enable telnet negotiation with the host
<i>TransmitBlockSize</i>	Set the maximum number of bytes to send to the host in one block. Set to 0 to specify no limit (recommended).
<i>ZeroTopBit</i>	True to clear the top bit of every character received.

Variable	Description
<i>TerminalType</i>	Set the terminal type to report to the host during telnet negotiation. Note that this value is reset when the emulation is changed.
<i>Echo</i>	True to request that the host echos back the data that is sent to it
<i>Binary</i>	True to allow 8 bit communications with the host

SetupEditKeys

Type

Global

Default Menu Location

Setup Keyboard

Description

This uses the same variables as Setup Function keys menu option,

When this dialog is shown it defaults to displaying the edit keys.

Variables

This menu option has no associated variables

SetupFunctionKeys

Type

Global

Default Menu Location

Setup Keyboard

Description

This menu options is used to setup the soft key definitions.

Variables

The following table describes the variables available for the SetupFunctionKeys menu option:

Variable	Description
<i>Key_<name></i>	Set the value for the soft key with the specified name. If the value starts "\"m" then the rest of the definition is a script command, otherwise it is a back slash encoded string of the bytes to send to the host. See below for a list of the key names.
<i>BreakKey</i>	The number of the key that send the communications defined break signal. This number is interpreted so that Ctrl+@ = 0, Ctrl+A = 1, Ctrl+C = 3 etc.
<i>CapsLock</i>	Set the action for the CapsLock key status when the session gets the focus. See below
<i>NumLock</i>	Set the num lock state when the session gets the focus. This has the same options as CapsLock with the exception there is no Force mode.

Values for Key_<name>

The names for the keys are as follows:

Value	Description
<i>C_ or Control_</i>	Place before a key name to specify that the control key is also held down
<i>S_ or Shift_</i>	Place before a key name to specify that the shift key is also held down
<i>A_ or Alt_</i>	Place before a key name to specify that the alt key is also held down
<i>F1 to F24</i>	Function keys
<i>MouseLeft</i>	Left mouse button
<i>MouseRight</i>	Right mouse button
<i>MouseMiddle</i>	Middle mouse button (three-button mouse)
<i>MouseBoth</i>	Pseudo value for both mouse buttons together
<i>Backspace</i>	BACKSPACE key
<i>Tab</i>	TAB key
<i>Clear</i>	CLEAR key
<i>Return</i>	Return key
<i>Escape</i>	ESC key
<i>Spacebar</i>	SPACEBAR
<i>PageUp</i>	PAGE UP key
<i>PageDown</i>	PAGE DOWN key
<i>End</i>	END key
<i>Home</i>	HOME key
<i>LeftArrow</i>	LEFT ARROW key
<i>UpArrow</i>	UP ARROW key
<i>RightArrow</i>	RIGHT ARROW key
<i>DownArrow</i>	DOWN ARROW key

Value	Description
<i>Select</i>	SELECT key
<i>Execute</i>	EXECUTE key
<i>PrintScreen</i>	PRINT SCREEN key
<i>Insert</i>	INS key
<i>Delete</i>	DEL key
<i>Help</i>	HELP key
<i>0 to 9</i>	Number keys
<i>A to Z</i>	Alpha keys
<i>NumPad0 to 9</i>	Numbers on the numeric key pad
<i>Multiply</i>	Multiply key on the numeric key pad
<i>Add</i>	Add key on the numeric key pad
<i>Separator</i>	Separator key on the numeric key pad
<i>Subtract</i>	Subtract key on the numeric key pad
<i>Decimal</i>	Decimal key on the numeric key pad
<i>Divide</i>	Divide key on the numeric key pad
<i>Enter</i>	ENTER on the numeric keypad
<i>CapsLock</i>	CAPS LOCK key
<i>NumLock</i>	NUM LOCK key
<i>ScrollLock</i>	SCROLL LOCK key
<i>National1 to 12</i>	Keyboard specific (OEM) keys
<i>Pause</i>	PAUSE key
<i>Break</i>	Break key

Values for CapsLock

Value	Description
<i>Normal</i>	No special processing
<i>On</i>	Turn the caps lock on
<i>Off</i>	Turn the caps lock off
<i>Recall</i>	Remember the state last time the session had the focus
<i>Force</i>	Use internal software processing for caps lock control

Related Script Commands

Key LoadEmulation, Key ClearAll, Key ClearSet, Key Run

SetupMouse

Type

Global

Default Menu Location

Setup Mouse

Description

This uses the same variables as Setup Function keys menu option,

When this dialog is shown it defaults to displaying the mouse buttons.

Variables

This menu option has no associated variables

SetupPreferences

Type

Global

Default Menu Location

Setup Preferences

Description

Sets various options for a session, including toolbar/scrollbar display and startup scripts.

Variables

The following table describes the variables available for the SetupPreferences menu option:

Variable	Description
<i>Title</i>	Any text for the window title.
<i>VScrollBar</i>	True to turn on the vertical scroll bar
<i>HScrollBar</i>	True to turn on the horizontal scroll bar
<i>StatusBar</i>	True to turn on the status bar
<i>Name</i>	The name for this session (8 characters max).
<i>SavePortOpen</i>	Effects how the PortOpen flags is saved in the session. See table below.
<i>ExitOnPortClose</i>	Set this flag to cause the current session to automatically close whenever the communication port is closed. Note: This setting is ignored if the Event OnPortClose script command has defined a script to be run when the communication port is closed.

Variable	Description
<i>CopyTrim</i>	True to trim off trailing spaces from data copied from the screen to the Windows clipboard
<i>EditInputKeys</i>	Set to true to use the edit keys for editing or navigating when editing data from the EditInput script routine.
<i>LoadScript</i>	Specify the script(s) to be run immediately the session script has been loaded.
<i>MenubarScript</i>	Set the script to modify the menu when the session is loaded.
<i>DeviceLicensing</i>	True to enable device licensing for UniVerse and UniData on UNIX
<i>CommandBar_<n></i>	Setup a command bar. <n> is a number from 1 upwards. Use the CommandBar script commands in preference to this.
<i>IconbarScript</i>	Retained for backward compatability. Use command bars.
<i>IconBar</i>	Retained for backward compatability. Use command bars.
<i>BaseBarScript</i>	Retained for backward compatability. Use command bars.
<i>BaseBar</i>	Retained for backward compatability. Use command bars.
<i>MenuBar</i>	Retained for backward compatability. Use command bars.

Values for SavePortOpen

Value	Description
<i>"Normal"</i>	save the current value of the PortOpen flag
<i>"Open"</i>	always save the port open flag as 1 (open)

Value	Description
<i>"Closed"</i>	always save the port open flag as 0 (closed)

Example

Ensure user gets login prompt every time the session is opened

```
Set SavePortOpen = "Open"
```

Related Script Commands

CommandBar Add, Event OnPortClose

Version

4.2.1 Added SavePortOpen variable

5.1.2 Added Exit on port close

SetupTerminal

Type

Global

Default Menu Location

Setup Terminal

Description

Set up the emulation and screen size

Variables

The following table describes the variables available for the SetupTerminal menu option:

Variable	Description
<i>Emulation</i>	Any terminal emulation file, followed by up to six WIT extensions. Note: This does not load the key definitions, to do this execute the Key LoadEmulation command after this variable has been updated.
<i>Columns</i>	Any numeric value up to 255. Usually 80, but can also be 40, 132, etc.
<i>Lines</i>	Any numeric value up to 255. Usually 24 but may also be 25, 51, 66, etc.
<i>BackPages</i>	A numeric value, subject to screen size. For 80x24 the maximum is 33 back pages.
<i>Cursor</i>	Set the cursor shape. "Block" or "Line"
<i>CursorBlink</i>	True for a blinking cursor, false for a non-blinking cursor
<i>Backspace</i>	Set the back space character. Normal (character 8), Delete (character 127) or any valid key setting.

Variable	Description
<i>HostEncoding</i>	Sets the character encoding used by the host.

Values for HostEncoding

Value	Description
<i>Default</i>	Latin I if Windows is running a single byte character set, otherwise Local
<i>Latin I</i>	The text from the host is copied to the lower byte of the internal Unicode characters
<i>Local</i>	Converts assuming the hosts code page matches the Windows code page.
<i>GB18030</i>	Converts characters to/from the GB18030 character set

Related Script Commands

Key LoadEmulation

Version

5.2 HostEncoding variable added

Appendix A

Scripting Libraries and Modules

This Appendix describes the libraries and modules available in wIntegrate.

wIntegrate provides four libraries, each of which contains a collection of commands and functions you can use with scripts you create. By loading the library in a script, you have access to each command contained in the library within the current script.

The four predefined libraries are:

Library	Description
Icon Library	Contains commands to build toolbars.
Tool Library	Contains commands to create floating tool bars.
Function Key Bar Library	Contains commands to create function key bars.
Server Library	Contains commands to communicate with the host computer.

wIntegrate also provides collections of commands and functions called "modules". You can use these commands and functions to create scripts.

These modules are:

Module	Module
Capture Module	Menus Module
Date and Time Module	Play Module
DDE Module	Screen Module
Dialog Box Module	Select Module
Display Module	Session Module

Module	Module
Draw Module	Session Variables Module
File Module	String Manipulation Module
HotSpot Module	Wait Module

Loading Libraries into Memory

You can load a library into memory, making it available for all scripts in the wIntegrate session. The script library can then be used in any other script without having to reopen and reread the library. This technique is called caching script libraries.

To load a library into memory, use the Library command. For more information, see the Library command in the "Reference - wIntegrate Script Commands" section of this manual.

Icon Library

The icon library commands enable you to create, edit, or enhance toolbars. A toolbar is a special type of dialog box.

Creating a Toolbar

Each toolbar described in the previous section is a script containing commands found in the Icon library, located in "Wintsys\lib". Each Icon library command is a series of script commands, many of which are DialogBox commands. Together, these script commands make up an Icon library command. Each Icon library command is described in detail in the "Reference - wIntegrate Script Commands" section of this manual.

The following example shows the CreateIconBar command, which is actually a script containing a series of DialogBox commands, located in "wintsys\lib\Iconlib.wis":

```
* Iconbar creation routines
* Version 2.6
Sub CreateIconBar($name, rows)
  icon_size = 11
  icon_largesize = 20
  icon_left = 2
  icon_no = 0
  icon_col = icon_left
  icon_row = 1
  DialogBox Create $name, 0, 0, 400, rows*(icon_Size+1) + 1
  Style WS_CHILD
  Font "helv",8
EndSub
.
.
.
```

Note:

You can modify the icon library commands, however, we recommend that you copy and rename the standard Iconlib script before making any modifications. The Iconlib script library is overwritten when you reinstall or update wIntegrate.

Creating a Toolbar

The following procedure illustrates how to create a toolbar by using the Bar_big.wis script located in the "Wintsys\Iconbar" directory.

1. Check if Toolbar Exists

The Bar_big.wis script first checks to see if the icon bar you are creating already exists. If it does, the script terminates. If it does not, the script creates it.

```
If IsShown(BigIcon_bar) Then EndScript
```

2. Load the Iconlib Library

Next, the iconlib library is loaded in the script, as shown in the following example:

```
Library 'wintsys\lib\iconlib'
```

3. Create the Toolbar

Use the CreateIconBar command to start the toolbar.

Syntax:

CreateIconBar *name, no_of_rows*

where *name* is the name of the new toolbar, and *no_of_rows* defines the number of rows for the toolbar. Each toolbar must have a unique name.

The following example illustrates the CreateIconBar command:

```
CreateIconBar BigIcon_bar,2
```

This command creates a toolbar named BigIcon_bar, containing 2 rows.

4. Add Icons

Next, use the `AddIcon` or `AddLargeIcon` command to specify the icon bitmap you want to use and the `wIntegrate` script command you want the icon to invoke. Use the `AddIcon` command to specify bitmap dimensions of 16x16. To add large icons with bitmap dimensions of 32x32, use the `AddLargeIcon` command.

Syntax:

AddIcon *bitmap, command*

AddLargeIcon *bitmap, command*

bitmap specifies the full path of the bitmap (.bmp) or icon (.ico) file you are adding.

command specifies the action the icon performs when you click on it.

Note

The icons used in `wIntegrate` are stored in the "icon" directory and the bitmaps and other images in the "image" directory.

The following code segment illustrates creating a large icon that runs Microsoft Excel by using the `AddLargeIcon` command:

```
AddLargeIcon 'icon\excel.ico','Dialog RunProgram;Set Filename =  
"excel.exe";Set Arguments = "";Invoke'
```

Continue adding icons until all the icons you want on the toolbar are defined.

5. Add Spaces Between Icons

If you want a space between icons or groups of icons, use the `AddLargeSeparator` command.

Syntax:

AddSeparator

AddLargeSeparator

The following example illustrates adding spaces between large icons after you define each icon:

```
.  
.   
.   
AddLargeIcon 'icon\excel.ico','Dialog RunProgram;Set Filename =  
"excel.exe";Set Arguments = "";Invoke'  
AddLargeIcon 'icon\word.ico','Dialog RunProgram;Set Filename =  
"winword.exe";Set Arguments = "";Invoke'  
AddLargeIcon 'icon\control.ico','Invoke FileControlPanel'  
AddLargeIcon 'icon\write.ico','Dialog RunProgram;Set Filename =  
"write.exe";Set Arguments = "";Invoke'  
AddLargeIcon 'icon\pbrush.ico','Dialog RunProgram;Set Filename =  
"pbrush.exe";Set Arguments = "";Invoke'  
AddLargeIcon 'icon\help.ico','Invoke HelpIndex'  
AddLargeSeparator  
.   
.   
.   

```

6. Position Icon to the Next Row

If you want the toolbar to display on more than one row, use the `NextRow` command to move the icon position to the next row.

Syntax:

NextRow

The following program segment illustrates use of the `NextRow` command:

```
.  
.   
.   
col = GetIconColumn()  
AddIcon 'image\i_lineup.bmp',"Invoke ScrollUpLine"  
AddIcon 'image\i_pageup.bmp',"Invoke ScrollPrevPage"  
AddIcon 'image\i_paget.bmp',"Invoke ScrollTopPage"
```

```
NextRow
SetIconColumn col
AddIcon 'image\i_linedn.bmp',"Invoke ScrollDownLine"
AddIcon 'image\i_pagedn.bmp',"Invoke ScrollNextPage"
AddIcon 'image\i_pageb.bmp',"Invoke ScrollEndPage"
AddSeparator
AddIcon                                     'image\i_barsel.bmp',"Script
'wintsys\script\barsel','bar_big' "
.
.
.
```

7. Terminate Creation of the Icon Bar

Use the `EndCreateIconBar` command to terminate the creation of the icon bar.

Syntax:

EndCreateIconBar

The following code segment illustrates the `EndCreateIconBar` command:

```
.
.
.
AddIcon                                     'image\i_barsel.bmp',"Script
'wintsys\script\barsel','bar_big' "
EndCreateIconBar
```

8. Display Toolbar

To display the toolbar, use the `MakeCommandBar` command.

Syntax:

MakeCommandBar *name*

This command registers the icon bar as a command bar and displays it. It also registers the current script as the script that create the command bar.

```
.  
.   
.   
pos = CommandLine  
.   
.   
.   
If pos = "A" Then  
    MakeCommandBar BigIcon_bar  
.   
.   
. 
```

Example

The following example shows the entire script for the Large icon bar. This script is located in "wintsys\Iconbar\Bar_big.wis".

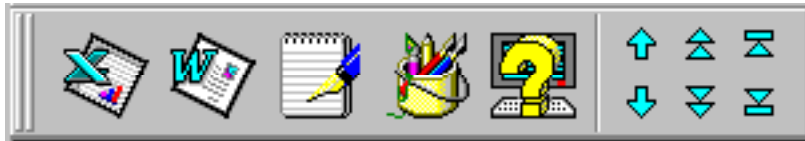
```
* Big Icon bar  
pos = CommandLine  
If IsShown(BigIcon_bar) Then EndScript  
If Not(IsDialog(BigIcon_bar)) Then  
Library 'wintsys\lib\iconlib'  
CreateIconBar BigIcon_bar,2  
AddLargeIcon 'icon\excel.ico','Dialog RunProgram;Set Filename =  
"excel.exe";Set Arguments = "";Invoke'  
AddLargeIcon 'icon\word.ico','Dialog RunProgram;Set Filename =  
"winword.exe";Set Arguments = "";Invoke'  
AddLargeIcon 'icon\control.ico','Invoke FileControlPanel'  
AddLargeIcon 'icon\write.ico','Dialog RunProgram;Set Filename =  
"write.exe";Set Arguments = "";Invoke'  
AddLargeIcon 'icon\pbrush.ico','Dialog RunProgram;Set Filename =  
"pbrush.exe";Set Arguments = "";Invoke'  
AddLargeIcon 'icon\help.ico','Invoke HelpIndex'  
AddLargeSeparator  
col = GetIconColumnn()  
AddIcon 'image\i_lineup.bmp',"Invoke ScrollUpLine"  
AddIcon 'image\i_pageup.bmp',"Invoke ScrollPrevPage"  
AddIcon 'image\i_paget.bmp',"Invoke ScrollTopPage"  
NextRow  
SetIconColumn col  
AddIcon 'image\i_linedn.bmp',"Invoke ScrollDownLine"  
AddIcon 'image\i_pagedn.bmp',"Invoke ScrollNextPage"  
AddIcon 'image\i_pageb.bmp',"Invoke ScrollEndPage"  
AddSeparator
```

```

AddIcon                                     'image\i_barsel.bmp', "Script
'wintsys\script\barsel','bar_big'
EndCreateIconBar
EndIf
If pos = "A" Then
    MakeCommandBar BigIcon_bar
Else If pos = "B" Then
    DialogBox Basebar BigIcon_bar
Else If pos = "I" Then
    DialogBox Iconbar BigIcon_bar
EndIf

```

The preceding script creates the standard wIntegrate Large toolbar:



Icon Library Commands

The following table describes the Icon library commands. For more information on these commands, see the "Reference - wIntegrate Script Commands" section in this manual.

Command	Description
AddCheckBox	Adds a check box to a toolbar.
AddComboBox	Adds a combo box to a toolbar.
AddGap	Adds a small space after an icon.
AddIcon	Adds a new icon to a toolbar.
AddLargeIcon	Adds a large icon to a toolbar.
AddLargeStyleIcon	Adds a large icon with a specified graphic button style to the toolbar.
AddLargeSeparator	Adds a double height separator between icons on a toolbar.

Command	Description
AddSeparator	Adds a single height separator between icons on a toolbar.
AddStyleIcon	Adds an icon with a specified graphic button style to the toolbar.
AddToolTip	Adds a tool tip to the last icon created by using the AddIcon , AddLargeIcon , or AddWideIcon commands. The tool tip is a small rectangular box that wIntegrate displays when you move the mouse over the icon.
AddWideIcon	Adds a new icon to the toolbar. Unlike the AddIcon command, you can specify the width of the icon with AddWideIcon .
AddWideStyleIcon	Adds a wide icon with a specified graphic button style to the toolbar.
CreateIconBar	Starts the creation of a new toolbar.
EndCreateIconBar	Terminates the creation of a toolbar.
GetIconColumn	Returns the position of the next icon on a toolbar.
MakeCommandBar	Makes a dialog box a toolbar.
NextRow	Moves the icon position to the next row.
SetIconColumn	Sets the column position for the next icon on a toolbar.

Tool Library

The tool library commands enable you to create floating toolbars. Unlike regular toolbars, floating toolbars are toolbars which are not attached to the top or bottom of the wIntegrate session. Each toolbar command is a series of script commands, many of which are DialogBox commands. Together, these script commands make up a Tool library command. Each tool library command is described in detail in the "Reference - wIntegrate Script Commands" section of this manual.

The following example shows the CreateToolBar command, which is actually a script containing a series of DialogBox commands, located in "wintsys\lib\toollib.wis":

```
* Floating toolbar creation routines
* Version 2.0
Sub CreateToolBar($name, cols,rows,size)
If size = 'L' Then
tool_size = 20
Else tool_size = 11
tool_no = 0
tool_col = 0
tool_row = 0
tool_EndCol = cols * tool_size
DialogBox Create $name, 5, 5, cols * tool_size, rows*tool_size
STYLE WS_POPUP | WS_CAPTION | WS_SYSMENU | WS_MINIMIZEBOX
Font "helv",8
Caption "ToolBar"
LoseFocus True
EndSub
.
.
.
```

Note

You can modify the toolbar library commands, however, we recommend that you copy and rename the standard toollib script before making modifications. The toollib script library is overwritten when you reinstall or update wIntegrate.

Creating a Floating Toolbar

You cannot use both large and small icons in floating tool bars, as you can in regular toolbars. When you use a toolbar with two or more columns, the first icon appears in the first column, the second icon appears in the second column, the third icon in the first column, etc.

The following procedure illustrates how to create a floating toolbar with large icons.

1. Check if Toolbar Exists

The following script first checks to see if the toolbar you are creating already exists. If it does, the script terminates. If it does not, the script creates it.

```
If IsShown(BigTool_bar) Then EndScript  
If Not(IsDialog(BigTool_bar)) Then
```

2. Load the Toollib Library

Next, the toollib library is loaded in the script, as shown in the following example:

```
Library 'wintsys\lib\toollib'
```

3. Create the Toolbar

Use the CreateToolBar command to start the toolbar.

Syntax:

CreateToolBar *name, no_of_columns, no_of_rows, size*

name is the name of the floating toolbar, *no_of_columns* defines the number of columns for the floating toolbar, *no_of_rows* defines the number of rows in the floating toolbar, and *size* defines the size of the icons. If you want to use 16x16 size icons, leave this parameter blank. To specify 32x32 size icons, use "L". Each floating toolbar must have a unique name.

The following example illustrates the CreateToolBar command using large icons:

```
CreateToolBar BigTool_bar,1,6,"L"
```

The command in the previous example creates an large toolbar named BigTool_bar, containing 1 column with 6 rows.

4. Add Tools

Next, use the AddTool command to specify the icon bitmap you want to use and the wIntegrate script command you want the icon to invoke.

Syntax:

AddTool *bitmap, command*

bitmap specifies the full path of the bitmap (.bmp) or icon (.ico) file you are adding. command specifies the action the icon performs when you click on it.

Note

The icons used in wIntegrate are stored in the "icon" directory and bitmaps are stored in the "image" directory.

The following code segment illustrates creating an icon that runs Microsoft Word by using the AddTool command. Because "L" was specified with the CreateToolBar command, the script creates large icons.

```
AddTool      'icon\word.ico','Dialog      RunProgram;Set      Filename      =  
"winword.exe";Set Arguments = "" ;Invoke'
```

Continue adding icons until you have defined all the icons you want on the tool bar.

5. Add Spaces Between Icons

If you want a space between icons or groups of icons, use the ToolSpace command.

Syntax:

ToolSpace

The following example illustrates adding spaces between the tool bar icons after you define each icon:

```
.  
.   
.   
.   
AddTool      'icon\word.ico','Dialog      RunProgram;Set      Filename      =  
"winword.exe"; Set Arguments = "";Invoke'  
AddTool      'icon\excel.ico','Dialog      RunProgram;Set      Filename      =  
"excel.exe"; Set Arguments = "";Invoke'  
AddTool      'icon\control.ico','Invoke FileControlPanel'  
AddTool      'icon\write.ico','Dialog      RunProgram;Set      Filename      =  
"write.exe"; Set Arguments = "";Invoke'  
AddTool      'icon\pbrush.ico','Dialog      RunProgram;Set      Filename      =  
"pbrush.exe";Set Arguments = "";Invoke'  
ToolSpace
```

6. Terminate Creation of the Toolbar

Use the EndCreate command to terminate the creation of the toolbar.

Syntax:

EndCreate

The following code segment illustrates the EndCreate command:

```
.  
.   
.   
AddTool      'icon\help.ico','Invoke HelpIndex'  
ToolSpace  
EndCreate
```

7. Display Toolbar

To display the toolbar, use the DialogBox Window command.

Syntax:

DialogBox Window *name*

This command displays a dialog box as a window or "modeless" dialog box, allowing several dialog boxes to display at the same time.

The following example illustrates the DialogBox Window command:

```
.
.
.
EndCreate
EndIf
DialogBox Window BigTool_Bar
```

Example

The following example shows the entire script for creating a large toolbar.

```
If IsShown(BigTool_bar) Then EndScript
If Not(IsDialog(BigTool_bar)) Then
Library 'wintsys\lib\toollib'
CreateToolBar BigTool_bar,1,6,"L"
AddTool 'icon\word.ico','Dialog RunProgram;Set Filename =
"winword.exe";Set Arguments = "";Invoke'
AddTool 'icon\excel.ico','Dialog RunProgram;Set Filename =
"excel.exe";Set Arguments = "";Invoke'
AddTool 'icon\control.ico','Invoke FileControlPanel'
AddTool 'icon\write.ico','Dialog RunProgram;Set Filename =
"write.exe";Set Arguments = "";Invoke'
AddTool 'icon\pbrush.ico','Dialog RunProgram;Set Filename =
"pbrush.exe";Set Arguments = "";Invoke'
AddTool 'icon\help.ico','Invoke HelpIndex'
ToolSpace
EndCreate
EndIf
DialogBox Window BigTool_Bar
```

The preceding script creates the following tool bar:



Toolbar commands

Command	Description
CreateToolBar	Starts the creation of a tool bar.
AddTool	Adds a new icon to the tool bar.
ToolSpace	Adds spaces between icons on a tool bar.
DialogBox EndCreate	Terminates the creation of a tool bar.

Function Key Bar Library

The function key bar library commands enable you to create a bar of function keys.

Each function key bar command is a series of script commands, many of which are DialogBox ommands. Together, these script commands make up a function key bar library command. Each library command is described in detail in the "Reference - wIntegrate Script Commands" section of this manual.

The following example shows the CreateFKeyBar command, a script which contains a series of DialogBox commands, located in "wintsys\lib\Fkeylib.wis":

```
.  
.   
.   
.   
Sub CreateFKeyBar($name, no_keys, but_width, type, text)  
If Not(IsDialog($name)) Then  
but_gap = but_width  
col = (but_width - but_gap)/2 + 2  
j = 0  
row = 2  
row_size = 12  
size = row_size + 4  
If type > 0 Then size = 2 * row_size + 4  
DialogBox Create $name 0,100,400,size  
Style WS_CHILD  
Font 8,"helv"  
Loop  
j+=1  
but = "F":j  
but_text = Field(text,'|',j)  
If but_text = "" Then but_text = but  
Pushbutton but_text,$but,col,row,but_width,row_size  
ControlCommand $but,"Key Run ':'but:":"  
col += but_gap  
If (j&3) = 0 then col+=2  
Repeat  
row += row_size+1  
col = (but_width - but_gap)/2 + 2  
If type = 1 Then  
Loop  
j += 1  
but = "F":j
```

```
but_text = Field(text,'|',j)
If but_text = "" Then but_text = but
Pushbutton but_text,$but,col,row,but_width,row_size
ControlCommand $but,"Key Run 'S_F':(j-no_keys):""
col += but_gap
If (j&3) = 0 then col+=2
Until j = no_keys * 2
Repeat
Else If type = 2 Then
j = 0
Loop
j+=1
but = "SF":j
but_text = Field(text,'|',j + no_keys)
If but_text = "" Then but_text = but
Pushbutton but_text,$but,col,row,but_width,row_size
ControlCommand $but,"Key Run 'S_F':j:""
col += but_gap
If (j&3) = 0 then col+=2
Until j = no_keys
Repeat
EndIF
EndCreate
EndIf
EndSub
```

Note

You can modify the function key bar library commands, however, we recommend that you copy and rename the standard Fkeylib script before making modifications. The Fkeylib script library is overwritten when you reinstall or update wIntegrate.

Check if Function Key Bar Exists

The following script first checks to see if the function key bar you are creating already exists. If it does, the script terminates. If it does not, the script creates it.

```
If IsShown(Function_bar) Then EndScript
If Not(IsDialog(Function_bar)) Then
```

Load the Fkeylib Library

Next, the Fkeylib library is loaded in the script, as shown in the following example:

```
Library 'wintsys\lib\Fkeylib'
```

Create the Function Key Bar

Use the CreateFKeyBar command to start the function key bar.

Syntax:

CreateFKeyBar *name, no_keys, but_width, type, text*

name is the name of the tool bar, *no_keys* defines the number of function keys per row, *but_width* defines the width of the function key buttons, *type* defines the type of function key bar, and *text* is the text that appears on the function key bar. Valid types are 0 for a single key strip, 1 for double line shift + key numbered, and 2 for double line shift + key prefixed.

The following example illustrates a script that creates a function key bar for use with VI editor keys:

```
pos = CommandLine
If IsShown(VI_KeyStrip) Then EndScript
If Not(IsDialog(VI_KeyStrip)) Then
Library "wintsys\lib\fkeylib"
CreateFKeyBar                                     VI_KeyStrip,
8,40,0,"Exit|SaveExit|Save|Compile|Goto|Search|Replace|Undo"
EndIf
If pos = "A" Then
Library "wintsys\lib\iconlib"
MakeCommandBar VI_KeyStrip
Else If pos="B" Then
DialogBox BaseBar VI_KeyStrip
Else DialogBox IconBar VI_KeyStrip
EndScript
```

Server Library

The server library consists of commands that enable you to read from and write to files on the host computer. Before you use any of the commands in this library, make sure that the wIntegrate host programs are installed on your host computer. For information on how to load these programs, see the Using wIntegrate manual.

Each server library command is a series of script commands. Together, these script commands make up a Server library command. Each server library command is described in detail in the "Reference - wIntegrate Script Commands" section of this manual.

The following example shows the HRead command, which is actually a script containing a series of commands, located in "wintsys\lib\Serverlib.wis":

```
* Provide PC side of the server routines
* Version 3.0.01 24 Feb 97
Sub HRead($varname, filename, item)
$varname = ""
CheckServer
If Server_Ready Then
Server_Ready = FALSE
Global Server_Record
Enter "READ ":filename:" ":item
Loop
Wait delay 20
Until Server_Ready
Repeat
$varname = Server_Record
EndIf
EndSub
```

Note

You can modify the server library commands, however, we recommend that you copy and rename the standard server script before making modifications. The server script library is overwritten when you reinstall or update wIntegrate.

Load the Server Library

The first step in creating a script that communicates with the host computer is to load the server library in the script, as shown in the following example:

```
Library 'wintsys\lib\server'
```

Each command in the server library checks to see if the server is running, and starts it if necessary.

Use the Server Commands

You can specify a variety of commands and functions in a script to read from host files, write to host files, or return information about a host file. To read records from a host file, use the HRead commands. To write records to a host file, use the HWrite commands. To return information about files from the host machine, use the Files, Fields and Items commands. You can also enable a local printer, execute multiple commands on the host machine, and open or close the server with server library commands.

Close the Server

After you specify all the server library commands you want to execute on the host machine, close the server using the CloseServer command.

Server Library Commands

The following table describes the server library commands:

Command	Description
HRead	Reads a record from a host file.
HReadV	Reads a single attribute from a record in a host file.
HReadU	Reads a single record from a host file and locks the record.

Command	Description
HReadVU	Reads a single attribute from a record in a host file and locks the record.
HWrite	Writes a record to a host file and releases the lock if the record was locked with the HReadU or HReadVU command.
HWriteV	Writes a single attribute to a record in a host file.
HWriteU	Writes a record to a host file and leaves the record locked if the record was locked with the HReadU or HReadVU command.
HWriteVU	Writes a single attribute to a record in the host file and leaves the record locked.
HDelete	Deletes items on the host.
HRelease	Releases a lock from a host file or record.
HCall	Calls a host subroutine.
Files	Sends a list of the files in the current account on the host machine to a list or combo box on the PC.
Fields	Sends a list of the fields in the host file to a list or combo box on the PC.
Items	Sends the host fields to a list or combo box on the PC.
LocalPrint	Turns on the local (PC) printer.
MultiExec	Executes one or more host statements via the host server.
StoreAccess	Stores statements for execution with ExecAccess. This command is used by the WIN.SERVER host subroutine.
ExecAccess	Executes statements stored with the StoreAccess command. This command is used by the WIN.SERVER host subroutine.

Command	Description
GetMachineType	Returns the name of the current operating system on the host machine.
CheckServer	Checks if the server is running, and starts the server if it is not.
OpenServer	Starts the host side of the server.
CloseServer	Stops processes on the host machine.
ServerErrMsg	Returns a description of any error returned by a Server library command.

Server Error Messages

Each call to a server library command returns a numeric status. If the call is successful, it returns 0. A return status other than 0 indicates an error.

The following table describes each error number and associated message:

Error Number	Description
1	Unable to start server
2	Server is busy
3	Incorrect number of arguments to server
4	Invalid file name
5	Invalid item
6	Invalid dictionary name
7	Invalid field name
8	Missing field number
9	Item is locked

Server Library Global Variables

The following table describes the global variables used by the server library:

Variable	Description
Server_Open	If the value is true, the server is running. If the value is false, the server is not running.
Server_Ready	If the value is true, the server can accept commands. If the value is false, the server cannot accept commands.
Server_Error	The last error from the server.

Capture Module

The capture commands provide ways to capture or record data from the host or the keyboard. The Record option on the Edit menu uses the capture commands. Multiple captures can be active at the same time.

Turn Capture On

You must use the Capture On command to start a capture session.

Syntax:

Capture On *name*, [*to*], [*format*], [*buffer_size*], [*options*]

name is the name of the capture session, to defines the destination for the captured data, *format* specifies what is being captured, and *buffer_size* specifies the size of the memory buffer.

Options

Options modify or regulate how the capture works. You can include more than one option by separating them with a bar (|). The following table describes the available options:

Option	Description
CAP_APPEND	Appends captured data to a memory or file capture.
CAP_AUTOOFF	Automatically turns off the capture when the buffer is full.
CAP_CharMapOff	Stops characters from being remapped during capture.
CAP_DIRECT	Sends captured data directly to the printer without formatting.

Option	Description
CAP_KEEP	Keeps the capture on Capture Off.
CAP_SHIFT	Retains at least half the buffer if the buffer is full.
CAP_TermCommandOff	Stops terminal commands contained in the WIT file from being executed during the capture.

Using the Capture Module

To use the capture module, complete the following steps:

1. Execute the Capture On command with appropriate options.
2. Have the program perform whatever actions you want to record for this capture session.
3. End the capture session with the Capture Off command.
4. Use any of the available Capture commands to add to, convert, or get information about a previous capture session.

Capture Commands

The following table provides a brief description of the available Capture commands. For a complete description of each command, see the individual command in the "Reference - wIntegrate Script Commands" section of this manual.

Command	Description
Capture On	Starts capturing data to PC memory, to a printer, or to a PC file.
Capture Off	Stops capturing data.
Capture Delete	Deletes a previous capture.
Capture AddData	Adds data to a capture.

Command	Description
CaptureInfo	Returns information on a capture.
CaptureText	Captures text in a format specified by available options.

Date and Time Module

The Date and Time module provides commands to read and convert the current date and time. You can use these commands any time you want to display or print the current time or date.

Date and Time Commands

The following table briefly describes each of the Date and Time module commands. For a full description, syntax, and usage of each command, see the "Reference - wIntegrate Script Commands" section of this manual.

Command	Description
Date	Returns the internal date.
Time	Returns the internal time.
Stamp	Returns the time and date in the 02:15:30 27 March 1998 format.
Iconv	Converts a numeric time or date into internal format.
Oconv	Converts an internal time or date into external format according to the format you specify.

DDE Module

Use the Dynamic Data Exchange (DDE) module to update information from wIntegrate that you also need in other Windows applications. For example, you can update a Microsoft Excel worksheet with data from the host computer with DDE commands.

Windows applications recognize wIntegrate as "winteg". The topic names for wIntegrate are the same as the session names, for example, "session1".

The following instructions provide an example of using the DDE module:

1. Start the DDE Conversation

Initiate a DDE conversation between wIntegrate and another application such as Microsoft Excel using the DDE Initiate command.

```
DDE Initiate SpreadSheet, "excel", "sheet1", OK
```

2. Set Up a Heading in Excel

Use the DDE Poke command to insert information in the application. This example adds a heading to the Microsoft Excel spreadsheet:

```
DDE Poke SpreadSheet, "r2c2", "Data sent by wIntegrate"
```

3. Send Files to Another Application

Send the file to the application. This example script uses "GetAndSend", a subroutine defined in the script that tells the host what files to send to the PC application:

```
GetAndSend "WIN.PROGS", "MACHINE.TYPE", 1, SpreadSheet, "r4c2"  
GetAndSend "MD", "WHO", 1, SpreadSheet, "r5c2"  
.  
.
```



```
.
Sub GetAndSend(HostFile, HostItem, HostAtt, $LinkName, LinkItem)
HReadyV Value, HostFile, HostItem, HostAtt
If Server_Error Then
    MsgBox ServerErrMsg(Server_Error) : " " : HostFile : ", " :
HostItem
Else
    DDE Poke $LinkName, LinkItem, Value
EndIf
EndSub
```

4. End the DDE Conversation

To end the DDE conversation, use the DDE Terminate command, as shown in the following example:

```
DDE Terminate SpreadSheet
```

DDE Commands

The following table briefly describes each DDE command. For a full description, syntax, and usage of each command, see the individual command in the "Reference - wIntegrate Script Commands" section of this manual.

Command	Description
DDE Advise	Establishes a link between wIntegrate and the server. Whenever the value of the server item changes, the associated global variable automatically updates.
DDEChanged	Returns True if a specified server item has changed.
DDE Execute	Runs a command on the server.
DDE HostLink	Send data to the host whenever the value of a server item changes.
DDE Initiate	Starts up a conversation with another PC application.

Command	Description
DDE Poke	Changes the value of a specified server item.
DDE Terminate	Stops a conversation started by DDE Initiate.
DDE Timeout	Sets the length of time for a command to wait for a response from the other application.
DDE Unadvise	Closes a link established with DDE Advise.
DDERequest	Retrieves data from a DDE connection.

Dialog Box Module

The Dialog Box module enables you to create dialog boxes, windows, regular toolbars and floating toolbars. wIntegrate also provides the Dialog Designer tool to create a dialog box, enabling point-and-click box design and creation. The Dialog Designer tool is described in "Appendix B - Using the Dialog Designer".

Before you begin creating a Dialog Box script, you should know how you want the dialog box to look, and what controls you want to provide in the dialog box.

Dialog Box

A dialog box contains various control buttons and other options that enable users to execute particular commands and tasks. A dialog box can just present information to the user, or users can interact with the computer by answering prompts presented in a dialog box. Typically, dialog boxes are no longer displayed after the requested information is provided by the user.

Dialog Box Elements

Dialog boxes contain controls, giving programmers control over various user actions. Controls gain input focus when you click them with the mouse. Controls include various types of buttons, arrows, and boxes.

See the "Reference - Script Controls" section for a full list of the controls available.

Dialog Box Commands

wIntegrate provides many commands to use when creating dialog boxes. The following table briefly describes the available Dialog Box commands. For detailed information about these commands, see the "Reference - wIntegrate Script Commands" section of this manual.

Command	Description
DialogBox AddToList	Adds a string to a list box or a combo box.
DialogBox Animate	Plays back a simple animation (avi) file in the dialog box.
DialogBox AxControl	Adds an ActiveX control to a dialog box.
DialogBox Caption	Specifies the title of the dialog box.
DialogBox CheckBox	Adds a check box control to a dialog box.
DialogBox CheckListBox	Add a list box where every line has a check box beside it.
DialogBox ColorButton	Adds a push button where the color of the text can be set.
DialogBox ComboBox	Creates a list-type box with various options.
DialogBox Control	Adds a control to the dialog box.
DialogBox ControlCommand	Specifies a script command to execute from a control.
DialogBox Create	Creates a new dialog box. When you execute this command, the script is put into dialog box creation mode.
DialogBox CText	Creates a centered text box.
DialogBox DateTime	Adds a control to allow the entry of dates and times to the dialog box.
DialogBox Default	Attaches a script command to a dialog box. wIntegrate executes the command when the specified control gains input focus.
DialogBox DefPushButton	Creates a default push button control, specifying the default action for the dialog box.
DialogBox Delete	Deletes a dialog box and its associated variables from memory.
DialogBox DeleteFromList	Deletes an entry from a list box or a combo box at a specified position.

Command	Description
DialogBox EditText	Creates a text box for user input.
DialogBox Enable	Enables or disables a specified control in a dialog box.
DialogBox EnableAll	Enables or disables the dialog box or specified controls within the dialog box.
DialogBox End	Closes a dialog box.
DialogBox EndCreate	Ends the creation mode started with DialogBox Create.
DialogBox EventLibrary	Loads a script library into memory that can be used by any script generated from an event on the dialog box.
DialogBox Font	Specifies the font for a dialog box.
DialogBox Graphic	Creates an image control in a dialog box.
DialogBox GraphicButton	Creates a push button control with a graphic image.
DialogBox Grid	Displays a grid of cells, starting at row1, column1. You can change the header information whenever you want by modifying row 0, column 0.
DialogBox GroupBox	Adds a rectangular group box to a dialog box.
DialogBox Header	Adds a header to a dialog box.
DialogBox Icon	Creates a control for the wIntegrate icon.
DialogBox IconBar	Defines a dialog box as a toolbar.
DialogBox InitCommand	Attaches a command to execute when you display the dialog box.
DialogBox InputOK	Used with DialogBox Validate, this command indicates if the input is valid.
DialogBox LimitText	Specifies the maximum number of characters in an edit control or combo box.

Command	Description
DialogBox ListBox	Creates a list box in a dialog box.
DialogBox ListView	Provides a control which can show a list of items, including images and additional information. For example, you can use ListView to produce a display such as the right hand pane in Windows Explorer.
DialogBox LoseFocus	Makes a dialog box lose focus when a button or a list box is activated when you display the dialog box as a window or a toolbar.
DialogBox LText	Creates a left-justified text box in a dialog box.
DialogBox MaxSize	Specifies the maximum size to which a dialog box can be resized.
DialogBox MinIcon	Specifies the icon to use when users minimize a dialog box.
DialogBox MinSize	Specifies the minimum size to which a dialog box can be resized.
DialogBox Move	Changes the position of a dialog box.
DialogBox MoveControl	Move a control on the dialog box.
DialogBox NextControl	Moves the focus in the named dialog box to the next control in the tab order.
DialogBox New	Creates a new dialog box, but does not put the script into creation mode.
DialogBox OnDialogEvent	Specifies the action to take when the dialog box generates an event.
DialogBox OnEvent	Specifies the action to take when a control generates an event.
DialogBox Option	Sets various options that effect dialog box processing.

Command	Description
DialogBox Panel	Divides the dialog box into panel for automatic repositioning of control when resizing.
DialogBox PopupMenu	Pops up a menu relative to the dialog box or a control.
DialogBox PopupWindow	Shows a new dialog relative to a control.
DialogBox ProgressBar	Creates a progress bar in a dialog box.
DialogBox PushButton	Adds a raised rectangular push button to a dialog box. Push buttons initiate commands when selected.
DialogBox RadioButton	Creates a radio button to select one of multiple mutually exclusive options.
DialogBox Reply	Used in conjunction with event handling, specifies the response for an event that requires a response. You must use this command before any command that causes multitasking, or multitasking will have no effect.
DialogBox ResetControls	Enables all the controls in a dialog box to be reset. After running this command, you must remake any selection in a list box or combo box.
DialogBox RText	Creates a right-justified text box in a dialog box.
DialogBox ScrollBar	Creates a scroll bar in a dialog box.
DialogBox ScrollStep	Specifies the amount the window moves with the scroll bar. Use with the DialogBox MaxSize command.
DialogBox Select	Selects a value from the list box or combo box currently displayed.

Command	Description
DialogBox SetColor	Enables users to set the text color and background color for an individual control within a dialog box.
DialogBox SetFont	Specifies the font used by an individual control. You can set the fonts used for a control to be different from the font used by default for the dialog box.
DialogBox SetKey	Defines an action for a key in a dialog box.
DialogBox SetMenu	Enables users to use a previously defined menu in a dialog box. The menu must not be attached to another menu or dialog box.
DialogBox SetRange	Specifies the range of values represented by a scroll bar.
DialogBox SetTabs	Sets tabs in a list box created with the LBS_USETABSTOPS option.
DialogBox SetText	Changes the text of a control.
DialogBox Show	Displays a dialog box, allowing input only to the current dialog box. Use to display the dialog box after you create it.
DialogBox ShowControl	Shows or hides a control.
DialogBox Size	Changes the size of a dialog box.
DialogBox Style	Sets the style of a dialog box.
DialogBox TabControl	Creates a tab control in a dialog box.
DialogBox ToolTip	Sets up the text display when the mouse remains over a specified control for a few seconds. You can change tool tips after you show the dialog box, but at least one tool tip must be defined in the dialog box prior to showing it.
DialogBox TrackBar	Adds a track bar control to the dialog box.

Command	Description
DialogBox TreeView	Displays hierarchical data. Each line of the control may be expanded to show further lines. You can assign images to each line.
DialogBox UpDownButton	Creates an up/down button in the dialog box.
DialogBox Validate	Specifies which command to run when a dialog box loses the input focus. Used with edit, list, and combo boxes.
DialogBox Variable	Sets up a variable for the dialog box, that can then be accessed in the same way as the default value for a control.
DialogBox Window	Displays a dialog box as a window, allowing several dialog boxes to display at the same time.
DialogBox WindowState	Set the state of a displayed dialog box.

Display Module

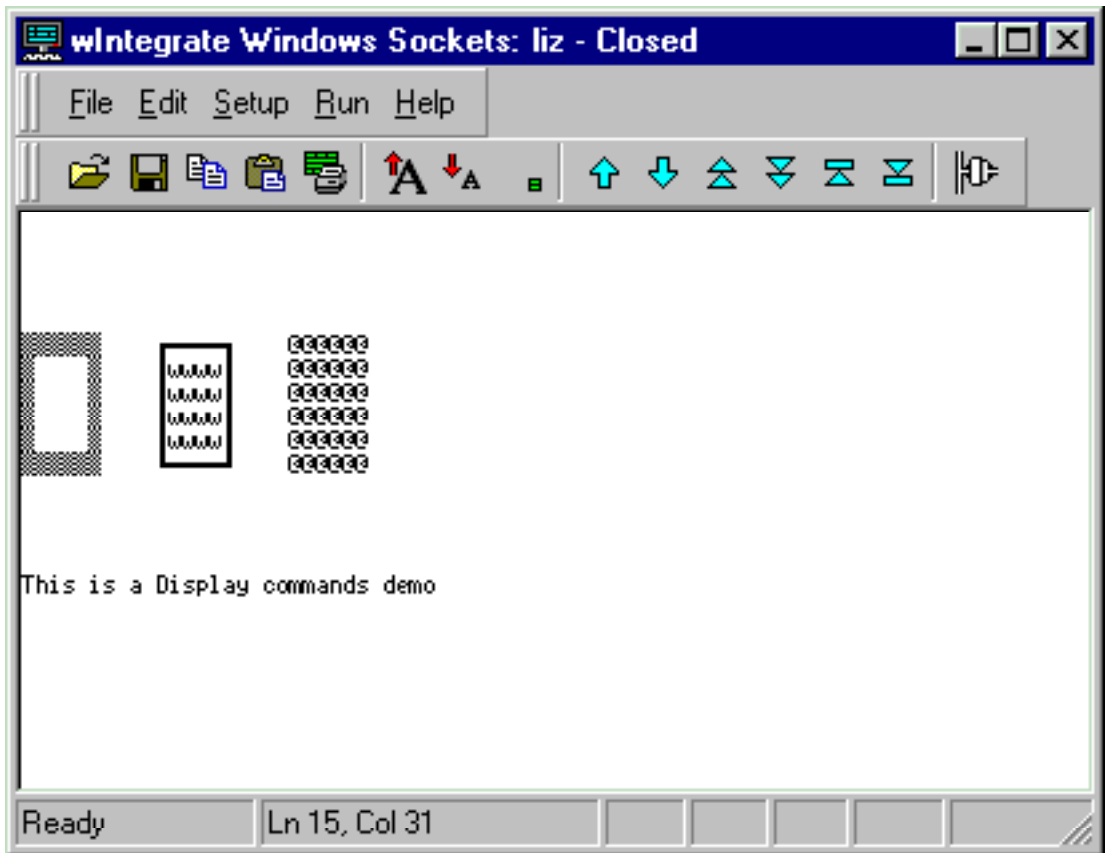
Use the commands in the Display module to display data on the wIntegrate window. You can display boxes, text, fill an area with characters, etc.

The following example illustrates how to use some of the Display module commands. These examples are taken from the Dispbox.wis script located in "Example\Script".

To clear the screen, use the Display FF command. You can place borders, character fill, move the cursor, and display text using other Display commands shown in the example:

```
* Demonstrate Display commands
* Clear the screen
Display FF
* Display a box with a block border
Display Box 0,10,5,5,3
* Display a box with a double line border, filled with w's
Display Box 10,10,15,5,2
Display CharFill 11,9,14,6,"w"
* Display the CharFill command with no box
Display CharFill 20,10,25,5,"@"
* Move the cursor and display text
Display At 0,15
Display Text "This is a Display commands demo"
```

The example above produces the following results:



Display Commands

The following table briefly describes the available Display commands. For detailed information about these commands, see the "Reference - wIntegrate Script Commands" section of this manual.

Command	Description
Display Command	Runs an emulation command (the word Command is optional).
Display At	Moves the cursor to the position you specify.
Display Box	Displays a box on the wIntegrate screen.

Command	Description
Display CharFill	Fills a portion of the screen with a character.
Display Direct	Displays data directly on the screen without going through the emulation.
Display Effect	Selects the effect to display in the wIntegrate window.
Display EffectFill	Fills a portion of the screen with an effect or a color.
Display Text	Displays text on the screen, going through the emulation.

Draw Module

The Draw module provides commands that enable you to place graphics directly on the wIntegrate window. You can also set the colors of the graphics, and, in some cases, styles.

The Draw module is object-based, so that when you draw a line, wIntegrate stores the line as an object rather than individually drawn pixels. Because of this, the drawing is scaled accurately if a user resizes the screen. You can delete or modify individual components of the drawing.

Use the following steps to create a drawing:

1. Using a Drawing as a wIntegrate Control

Use the Draw To command to display draw commands on a specified wIntegrate Drawing control.

Syntax:

Draw To *Screen / draw control*

If you specify Screen, wIntegrate draws directly to the wIntegrate window. draw control specifies the name of the draw control in the dialog box.

2. Set Pen and Brush Styles

Set pen styles using the Draw Pen command. When using the Draw Text command, the foreground color is set with Draw Pen.

Syntax:

Draw Pen *color, [style], [width]*

color sets the color for the pen, style sets the pen style, and width sets the thickness of the pen. For information about available colors, styles, and widths, see the Draw Pen

command in this manual. Set brush styles with the Draw Brush command. When using the Draw Text command, you set the background color with Draw Brush.

Syntax:

Draw Brush *color, style*

color sets the color for the brush, and *style* sets the style for the brush. For information about available colors and styles, see the Draw Brush command in this manual.

3. Draw Lines and Objects

Draw lines and objects with the Draw Line, Draw Rect, Draw Chord, etc. commands. For information about use of these commands, see the individual command in this manual.

4. Draw to the Screen

If you set the Draw To command to a wIntegrate draw control in step 1, use Draw To again to set the output back to the screen (Draw To Screen).

Example

The following example illustrates several of the draw commands and how they can be used together. Before you begin using the Draw commands, you should have a good idea of how you want the screen to look. This way, you can place your components correctly.

```
*Program to demonstrate wIntegrate Draw commands
*Set the border to blue, the background to LightGreen
Draw Pen RGB_Blue
Draw Brush RGB_LightCyan
*Draw a rectangle
Draw Rect 30, 100, 600, 200
*Set the font and pen color for the text
Draw Font "Times New Roman", 20,20,Font_Heavy|Font_Italic
Draw Pen RGB_None
```

```
Draw Text 75,140, "wIntegrate"  
*Place a wIntegrate image in the rectangle  
Draw Image 250,140,300,260,appdir:"icon\wint_pie.ico"  
*Set the border (Pen) and fill (Brush) for the ellipse  
Draw Pen RGB_Black, PEN_Dot  
Draw Ellipse 330,110,410,190  
*Set the border (Pen) and fill (Brush) for the polygon  
Draw Pen RGB_White, PEN_Solid, 4  
Draw Brush RGB_LightMagenta  
Draw Polygon 450,110,415,110,450,180  
*Set the border (Pen) and fill (Brush) for the pie segment  
Draw Pen RGB_Red  
Draw Brush RGB_Yellow  
Draw Pie 480,110,580,190,240,115,120,230
```

The previous script results in the following:



Draw Module Commands

The following table briefly describes each Draw module command. For a full description of a command, including syntax and usage, see the "Reference - wIntegrate Script Commands" section in this manual.

Command	Description
Draw Arc	Draws an arc of an ellipse.
Draw Brush	Sets the color and style used by other Draw commands.

Command	Description
Draw Chord	Draws a chord of an ellipse.
Draw Ellipse	Draws an ellipse within a rectangle.
Draw Erase	Erases objects drawn by any of the draw commands.
Draw Font	Sets the font name and size.
Draw Image	Draws an image in a rectangle.
Draw Line	Draws a line in a specific position.
Draw Move	Moves the current cursor position without drawing.
Draw Pen	Sets the color, style, and thickness of the pen.
Draw Pie	Draws a pie segment of an ellipse.
Draw Polygon	Draws a polygon with up to 5 vertices.
Draw Rect	Draws a rectangle using coordinates.
Draw Size	Sets the size of the drawing.
Draw Text	Draws text on the screen.
Draw To	Specifies whether to set a drawing to the terminal screen or to a wIntegrate control.

File Module

The File module consists of commands and functions that manipulate PC files in a variety of ways.

Some commands must be used together, such as File Open, File Read, File Write, and File Close commands. Others can be used any time you need to list files or get information on a file.

The steps below outline how to browse available files, open a file, set a pointer to read a particular item in the file, then close it. The examples are taken from the Fileread script, located in "Example\Script\Fileread.wis".

1. Browse Files

Start by using the File Browse command to view the files from which you can select, as shown in the following example:

```
* Demonstrate some file commands
* Version 2.0
defaultfile = "wistestfileread.wis"
filename = FileBrowse("Test of file info/read commands", defaultfile,
0, "Scripts (*.wis)|*.wis|All (*.*)|*.*")
```

2. Open a File

You open the file you select by using the File Open command:

```
If filename # "" Then
msg = "Info: " : FileInfo(filename)
File Open f_test, filename
```

3. Read the File

To select information to read from the file, use the File Read command. In the

following example, File Read read the first 10 bytes of the selected file:

```
File Read f_test, r_test, 10
Ins msg, "Ascii: " : r_test
.
.
.
```

4. Set a Pointer in the File

In the next example, the File Pointer command sets a position in the file, then the File Read command retrieves data from the specified location within the file.

```
File Pointer f_test, 0
File Read f_test, r_test, 10, DF_BS
Ins msg, "Backslash: " : r_test
File Pointer f_test, 0
File Read f_test, r_test, 10, DF_HEX
Ins msg, "Hex: " : r_test
File Pointer f_test, 0
File Read f_test, r_test, 10, DF_FT
Ins msg, "FT: " : r_test
```

5. Close the File

When you finish with a file, use the File Close command to close it. In the following example, the MessageBox command displays a message box, and the File Close command closes the file.

```
MessageBox msg
File Close f_test
EndIf
```

The previous script displays this message box:



File Commands

The following table lists the Draw module commands. For a complete description, syntax, and usage of each command, see the "Reference - wIntegrate Script Commands" section in this manual.

Command	Description
FileBrowse	Displays the Windows File selection dialog box.
File Close	Closes a file opened by File Open.
File Copy	Copies a file to another file with a different path.
File Create	Creates a new PC file.
File Createdir	Creates a new PC directory.
File Delete	Deletes a PC file.
File Deletedir	Deletes a PC directory.
FileExist	Checks to see if a file exists.
FileInfo	Returns information about a file based on options you specify.
FileList	Returns a list of matching files and/or subdirectories.

Command	Description
File Move	Moves or renames a file.
File Open	Opens a file.
File Pointer	Positions a pointer in the file based on the number of bytes you specify.
File Read	Reads data from a previously opened file.
File Write	Writes the value of a variable to an open file.

HotSpot Module

The HotSpot module provides commands that enable portions of the terminal screen to respond to mouse clicks. Hot spots override mouse actions you specify with Setup Mouse.

Use the following steps to create a hot spot:

1. Add a Hot Spot

To add a hot spot to the screen, use the HotSpot Add command.

Syntax:

HotSpot Add *left, top, right, bottom, click_text, double_text*

left is the screen column position to start the hot spot, *top* is the screen row position to start the hot spot, *right* is the screen column position for the hot spot to extend on the right, *bottom* is the row position for the hot spot to extend on the bottom, *click_text* defines the action to take place when you click the hot spot once, and *double_text* defines the action to take place when you click the hot spot twice.

Note

The text for a mouse action must be preceded by "\m". This specifies the text after the "\m" as a script command. If you do not use the "\m", the text is sent to the host.

The following example illustrates the HotSpot Add command:

```
HotSpot Add 0,12,79,23, "\mMessageBox 'click on the top half of the  
screen'", "\mMessage Box 'double click on the top half of the screen'"
```

2. Modify a Hot Spot

If you want hot spots to display in color or with an effect, use the HotSpot Display command, after first creating the hot spot with the HotSpot Add command.

Syntax:

Hotspot Display *flag, fore_col, back_col*

flag specifies the hot spot effect, *fore_col* specifies the foreground color, and *back_col* specifies the background color. For more information about the HotSpot Display command, see the individual command in this manual.

The following example specifies a hot spot with a red foreground and a yellow background that overrides the screen colors:

```
HotSpot Display 64, COL_Red, COL_Yellow; * Display as red on yellow
```

3. Turn Hot Spot On or Off

To temporarily turn hot spots on or off, use the HotSpot On or HotSpot Off commands. In the following example, wIntegrate creates a hot spot, waits 5 seconds and turns it off, then waits 5 seconds and turns it on:

```
HotSpot Add 40,0,79,11 "\mMessageBox 'Top right hotspot clicked'"  
Wait Delay "5s"  
HotSpot Off  
Wait Delay "5s"  
HotSpot On
```

4. Delete a Hot Spot

To delete one hot spot at a time, use the HotSpot Delete command, specifying the particular location of the hot spot you want to delete. To delete all hot spots, use the HotSpot DeleteAll command.

```
HotSpot Delete 40,0,79,11
```

Hot Spot Commands

The following table briefly describes each hot spot command. For a full description of each command, include description, syntax, and usage, see the "Reference - wIntegrate Script Commands" section in this manual.

Command	Description
HotSpot Add	Adds a hot spot to a dialog box.
HotSpot AutoDelete	Automatically deletes a hot spot when a form feed is received from the host.
HotSpot Delete	Deletes a single hot spot at a specified location.
HotSpot DeleteAll	Deletes all currently defined hot spots.
Hotspot Display	Displays hot spots in colors and effects.
HotSpot On	Turns on hot spot processing after processing has been terminated by the HotSpot Off command.
HotSpot On	Turns off hot spot processing.

Menus Module

wIntegrate provides the Menus module to modify or create new menus. The steps in this section describe how to create a menu to take the place of the main wIntegrate menu.

1. Create a Menu

Start the creation of a new menu with the Menu Create command.

Syntax:

Menu Create *menu*

menu specifies the name of the menu you want to create. The name you specify does not appear on the menu bar. Secondary menus are attached to this menu after they are created, so no other definition is needed after the Menu Create command.

Use the EndCreate command to end menu creation, as in the following example:

```
Menu Create NewMenu  
EndCreate
```

2. Create Secondary Menus

Create secondary menus by using the Menu Create command. The name you specify appears on the main menu bar.

To add menu items to each secondary menu, use the Menu AddItem command. You can abbreviate this command to AddItem.

Syntax:

Menu AddItem *menu, item, text, command, [position]*

menu is the name of the menu to which this item is attached. *item* specifies the name

of the item you are adding. text specifies the text to display for this item. command specifies the script command to execute when you choose the item. position is either a menu item name in quotation marks or the position number of the menu item that follows the added item.

An ampersand (&) underlines the letter after it and makes it a shortcut to select the item.

When you finish adding all items to the secondary menu, end creation of the secondary menu with the EndCreate command.

```
Menu Create Unix
AddItem ListFiles, "List &Files", "Enter '!ls -l |more'"
AddItem ListUsers, "&List Users", "Enter '!who |more'"
AddItem DiskSpace, "&Disk Space", "Enter '!df |more'"
AddItem ListProcess, "List &Process", "Enter '!ps -ef |more'"
EndCreate
Menu Create UniData
AddItem ListLocalFiles, "List Local &Files", "Enter 'LISTFL'"
AddItem ListUDTUsers, "&List Users", "Enter 'LISTUSER'"
AddItem DiskUDTSpace, "&Disk Space", "Enter 'AVAIL'"
EndCreate
Menu Create DosMenu
AddItem Excel, "&Excel", 'Dialog RunProgram;Set Filename =
"excel.exe";Set Arguments = "";Invoke'
AddItem ListDir, "List &Directory", 'Dialog RunProgram;Set Filename =
"command.com";Set Arguments = "dir";Invoke'
EndCreate
```

3. Create the Standard Secondary Menu

The last secondary menu you specify always returns the user to the default wIntegrate menu. The AddItem command enables you to call a script. The script called in this example is shown at the end of this procedure.

```
Menu Create Standard
AddItem RestoreStandard, "Standard &Menu", "Script 'pref.wis'"
EndCreate
```

4. Attach Secondary Menus

Attach the secondary menus to the new menu by using the MenuAttach command. Attach the menus in the order you want them to appear on the menu bar.

Syntax:

Menu Attach *menu {submenu / item / menu_cmd}, "text", ["position"]*

menu is the name of the menu to which you are attaching the secondary menus. *submenu* is the name of the secondary menu to attach. *item* is the secondary menu item name to attach. *menu_cmd* is the wIntegrate menu command to attach. *text* specifies the text representing the attached item to display on the menu. *position* defines the location for the attached item. The new item is attached after the menu item name or item number that you specify here. The default position is the end of the menu.

In the following example, the secondary menus defined above are added to the new menu.

```
Menu Attach NewMenu, Unix, "UNI&X"  
Menu Attach NewMenu, UniData, "&UniData"  
Menu Attach NewMenu, DosMenu, "Dos&Menu"  
Menu Attach NewMenu, Standard, "&Default"
```

5. Make Main Menu

To replace the default menu with the new menu, designate the new menu as the main menu with the Menu Main command: Menu Main NewMenu

6. Script for Default wIntegrate Menu

The new menu is now complete, but you must write another script that the default secondary menu calls. The script is named *pref.wis*. This example assumes it is located in the default wIntegrate directory. If scripts are located in a directory other than the default, you must specify the entire path to call the script.

The following example uses the Menu Main Default command to recall the

wIntegrate menu, then runs the newsess.wis script located in the "wintsys\script" directory:

```
Menu Main Default  
Script 'wintsys/script/newsess.wis'
```

7. Run the Script

To run the script in a wIntegrate session, you can use the Run Script menu command, or call the script from another script.

The following example illustrates the menu created with this script:



Menu Commands

The following table briefly describes each wIntegrate Menu command. For more information about each command, including a description, usage, and syntax, see the "Reference - wIntegrate Script Commands" section in this manual.

Command	Description
Menu AddItem	Creates a menu item and adds it to a menu.
Menu Attach	Attaches an existing menu, menu item, or menu command to a menu.
MenuChecked	Verifies that a menu is checked, and returns true if it is.
Menu Command	Changes the action of a menu when users click on it.
Menu Create	Creates an empty menu and places the script in menu creation mode.

Command	Description
Menu Delete	Deletes a menu after it has been detached from another menu.
Menu DeleteItem	Deletes a menu item.
Menu Detach	Detaches a menu item, secondary menu, or wIntegrate command.
Menu Enable	Enables or disables a menu position.
MenuEnabled	Returns true if menu is enabled.
Menu EndCreate	Finishes creation of the current menu.
Menu Help	Adds a line of help text on the status bar when the menu option is selected.
Menu Main	Replaces the wIntegrate main menu with the specified menu.
Menu New	Creates an empty menu.
Menu OnClose	Runs the wIntegrate command you specify when you close a menu.

Play Module

The Play module commands enable you to manipulate or retrieve information about the playback of a capture. Use the EditRecord to record subsequent keystrokes or data, then play the recorded file using the EditPlay command. Inside a script, capture or record data, then play the data back using Dialog EditPlay command. The Play commands can stop, start, abort, or retrieve information about the playback.

Refer to the Capture module commands in this appendix for a complete listing of the Capture module commands.

Play Commands

The following table provides a brief description of the commands in the Play module. For more information about each command, including a full description, syntax, and usage, see the "Reference- wIntegrate Script Commands" section in this manual.

Command	Description
Play Abort	Stops the current playback.
PlayInfo	Checks to see if a playback is already running.
Play Start	Restarts the playback after the Play Stop command.
Play Stop	Stops the current playback.

Screen Module

The Screen Module provides commands that enable you to vary your screen display. Using these commands and functions, you can save screens or portions of screens for later use, and retrieve information about the current screen display.

The following procedures outline a simple example of how to use a few of the Screen Module commands.

1. Store or Save a Screen

In the following example, screen colors are set and text is defined to display. When wIntegrate displays the screen you want to save, use the Screen Store or Screen Save command. The Screen Store command saves the screen in memory, while the Screen Save command saves the screen to a PC file.

```
* Set regular screen colors
Color COL_Green, COL_Black
Display FF
*Set the font and pen color for the text
Draw Font "Times_Roman", 30,30,Font_Heavy|Font_Italic
Draw Pen RGB_Green
Draw Text 75,140, "wIntegrate"
* Store the screen
Screen Store reg
```

2. Display a Stored Screen

After the screen is stored or saved, the example script displays a message box that gives the user a choice. If the user chooses “OK,” wIntegrate executes a query statement. If the user chooses "No", a stored screen displays using the Screen Restore command.

```
* Message box for the report
reply = MessageBox("Run 'Student?'", , MB_OKCANCEL | MB_ICONQUESTION)
If reply = "OK" Then
    Enter "SORT STUDENT NAME BY NAME"
Else
```

```
Screen Restore reg  
EndIf
```

3. Delete a Screen from Memory

To delete the stored screen from memory, use the Screen Remove command.

```
Screen Remove reg
```

Screen Module Commands

The following table briefly describes each Screen module command. For a more information about each command, including a full description, syntax, and usage, see the "Reference - wIntegrate Script Commands" section in this manual.

Command	Description
Screen CopyPage	Copies data from a backpage to the current screen.
Screen Load	Displays a screen previously saved with the Screen Save com-mand.
Screen Off	Stops updating to the screen.
Screen On	Restores updating to the screen after wIntegrate executes a Screen Off command.
Screen Remove	Releases the memory used by a Screen Store command.
Screen Restore	Displays a stored screen.
Screen Save	Saves an area of the terminal screen to a PC disk.
Screen ScrollRegion	Restricts terminal output and scrolling to a specified region of the screen.

Command	Description
Screen Store	Stores elements in a specified area of the terminal screen in memory.
ScreenText	Returns text from the screen area you specify.
ScreenWord	Returns the specified word or character from the screen, depending on the format you specify.

Select Module

The Select Module commands select screens or areas of screens. Selecting part of a screen could help you differentiate the selected area from the rest of the screen. You can then perform options, such as saving or copying the screen, for later use.

The following procedure illustrates how to use some of the Select Module commands.

1. Select Part of the Screen

Select an area of the screen to highlight an important piece of information or a mandatory entry line by using the Select Area command:

```
Select Area 0, 0, 79, 10
```

2. Move the Cursor

Move the starting position of the selected area to highlight a smaller area of the screen by using the Select Start command:

```
Select Start 0, 5
```

3. Extend the Selection

Enlarge the selected area by using the Select Extend command. In this example, the command increases the selected area to include all of the first back page.

```
Select Extend 0,0,1
```

4. End the Selection

Use the Select End command to terminate the selection. In this example, wIntegrate starts by selecting the top part of the screen, then the Select End command selects a portion of the next line.

```
Select Area 0,0,79,10
Select End 40,11
```

5. Clear the Screen

To clear the screen of the highlighted areas, use the Select Clear command.

```
Select Area 0,0,79,10
Select End 40,11
.
.
.
Select Clear
```

Select Commands

The following table briefly describes each Select Module command. For more information about each command, including a full description, syntax, and usage, see the "Reference - wIntegrate Script Commands" section in this manual.

Command	Description
Select Area	Sets the current screen selection.
Select Clear	Clears the current selection.
Select End	Moves the end position of the current selection.
Select Extend	Increases the size of the current selection.
SelectInfo	Returns the coordinates of the current selection.
Select Keyboard	Turns on and off screen selection from the keyboard.

Command	Description
Select Start	Moves the start position of the current selection.

Session Module

The Session Module contains a set of commands that enable you to manipulate wIntegrate sessions that are already running. You can change the focus, resize, move, and retrieve information about different sessions. Two commands, Session Execute and Session Script, enable you to run commands or scripts on any named session that is running.

Session Module Commands

The following table briefly describes each Session Module command. For more information about each command, including a full description, syntax, and usage, see the "Reference - wIntegrate Script Commands" section in this manual.

Command	Description
Session Enable	Enables/disables the session window.
Session Execute	Executes a script on session name.
SessionInfo	Returns information on a session.
Session Move	Moves a session on the desktop.
Sessions	Returns information about running sessions.
Session Script	Runs a script file on a specified session.
Session Show	Changes the current session display.
Session Size	Resizes the session window.

Session Variables Module

Session variables are those variables that can be manipulated with the Dialog, Set, Store, Configure, and Update commands. These variables are displayed within the built-in dialog boxes from the menus. You can display the dialogs themselves can be displayed with the Show command, or you can execute them with the Invoke command.

The follow table briefly describes each Session Variable command. For more information about each command, including a full description, syntax, and usage, see the "Reference - wIntegrate Script Commands" section in this manual.

Command	Description
Configure	Places the current script in configuration mode.
Dialog	Sets the menu command for Show, Invoke, Set, Store, and Configure commands and the Get function.
Get	Returns the current value of the session variable.
Invoke	Runs a menu command. You must set the local variables for the menu commands with the Set command.
Set	Sets a variable within the current session.
Show	Displays a dialog attached to a wIntegrate menu command.
Store	Stores a wIntegrate session variable.
Update	Updates session variables set since the script was in configuration mode.

String Manipulation Module

The String Manipulation module provides commands that enable you to directly modify the value of variables.

The following table briefly describes each command in the String Manipulation module. For more information about a command, including a full description, syntax, and usage, see the "Reference -wIntegrate Script Commands" section in this manual.

Command	Description
Asc	Returns the ASCII character of a specified value.
Change	Changes parts of a string.
Char	Returns the ASCII character code of a specified character.
Convert	Replaces characters with corresponding characters within a variable.
Count	Returns the number of occurrences of a single character in a string.
DCount	Returns the number of fields delimited by a single character in a string.
Del	Deletes the data at a specified line or tab position.
Extract	Returns a substring at a specified position.
Field	Returns a substring delimited by a specified character.
Index	Returns the position of a substring in a string.
Ins	Inserts a string into a variable at a specified location.
InString	Returns the character position of a substring in a string.

Command	Description
IsNumber	Checks to see if a string is numeric.
Left	Returns the left-most character(s) of a string.
Length	Returns the number of characters in a string.
Locate	Returns the position of a substring within a string.
Match	Compares a string with a pattern.
Mid	Returns a substring of a string.
Rep	Replaces data at a line or tab position.
Right	Returns the right-most character(s) of a string.
String	Returns copies of a string.
Strip	Returns a copy of a string that contains only ASCII characters from 32 through 126.
Trim	Removes spaces from a string.

Wait Module

The Wait module contains commands that enable you to pause the execution of a script until a specified action is complete. The pause can last a certain amount of time, until a task is performed, or until text is received. Use Wait commands to control the flow of your scripts.

Wait Commands

The following table briefly describes each command in the Wait module. For more information about each command, including a full description, syntax, and usage, see the "Reference -wIntegrate Script Commands" section in this manual.

Command	Description
Wait Delay	Pauses execution of a script for a specified period.
Wait DuringTask	Waits until the specified task is completed.
Wait ForText	Waits for text from the host.
Wait RxEmpty	Waits until the receive-from-host buffer is empty.
Wait TxEmpty	Waits until the transmit-to-host buffer is empty.

Appendix B

Using the Dialog Designer

wIntegrate provides the Dialog Designer to help you create dialog boxes. You can design the dialog box, add controls, such as buttons and list boxes, and assign commands to the controls you create.

The Dialog Designer should be used primarily for creating the look of a dialog box. You need to complete the script by using a text editor after you design the dialog box.

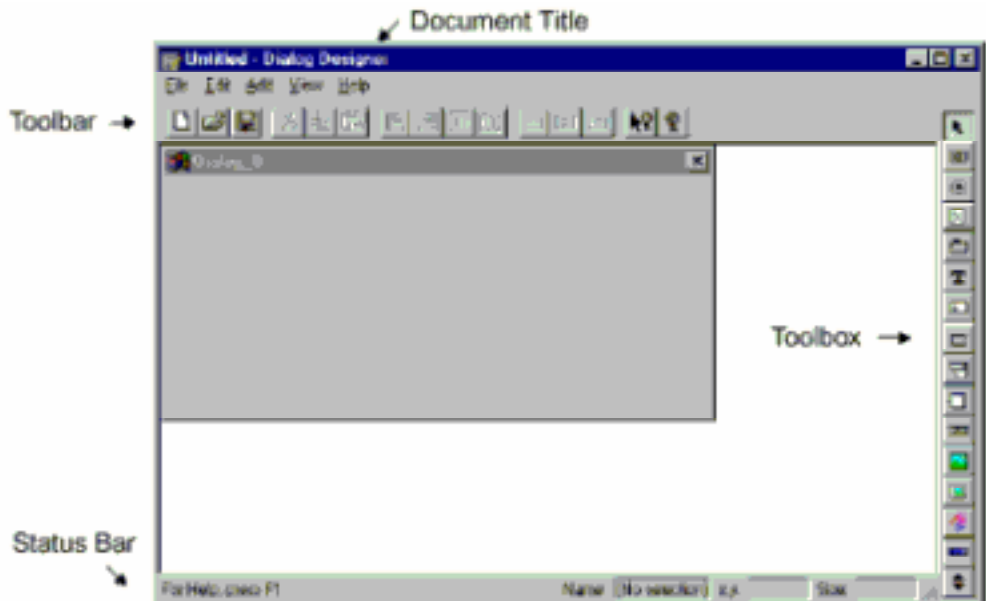
This appendix provides step-by-step instructions for creating a dialog box.

Dialog Designer

The default path for the Dialog Designer is "C:\Program Files\wIntegrate\Wintdlg.exe". To run the dialog designer program in Microsoft Windows 95, from the Start menu, select Programs, then select wIntegrate 98, and then click Dialog Designer:



To start a new dialog box after the Dialog Designer opens select File New from the menu or press the new toolbar button. This will start an untitled document, as shown in the following example:



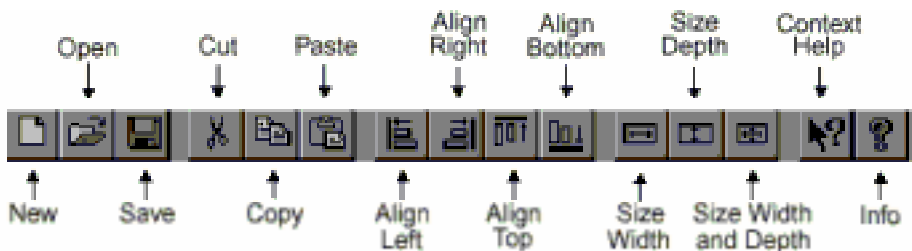
The toolbar, toolbox, and status bar are options in the Dialog Designer that you can turn on and off by using the View menu.

Understanding the Toolbar

The toolbar provides short cuts to accomplish the following tasks:

- Create a new file.
- Open an existing file.
- Save a file.
- Cut, copy, or paste text.
- Align boxes in relation to each other.
- Size boxes in relation to each other.
- Get context-sensitive help.
- Get help on the Dialog Designer.

The following example illustrates the Dialog Designer toolbar:



Note

The align and size buttons are activated only when you select two or more controls in the Dialog Designer.

The toolbar commands are also available from the Designer Dialog menu. For example, to choose the align commands, on the Edit menu, click Align.

Toolbar Commands

The following table describes the relationship between the toolbar icons and the menu commands:

Icon	Menu Command
New	From the File menu, click New.
Open	From the File menu, click Open.
Save	From the File menu, click Save.
Cut	From the Edit menu, click Cut.
Copy	From the Edit menu, click Copy.
Paste	From the Edit menu, click Paste.
Align left	From the Edit menu, click Align, and then click Left.
Align right	From the Edit menu, click Align, and then click Right.
Align top	From the Edit menu, click Align, and then click Top.
Align bottom	From the Edit menu, click Align, and then click Bottom.
Size width	From the Edit menu, click Size, and then click Width.
Size depth	From the Edit menu, click Size, and then click Depth.
Size width and depth	From the Edit menu, click Size, and then click Both.
Context help	From the Help menu, click Index.
Info	From the Help menu, click About WINTDLG.

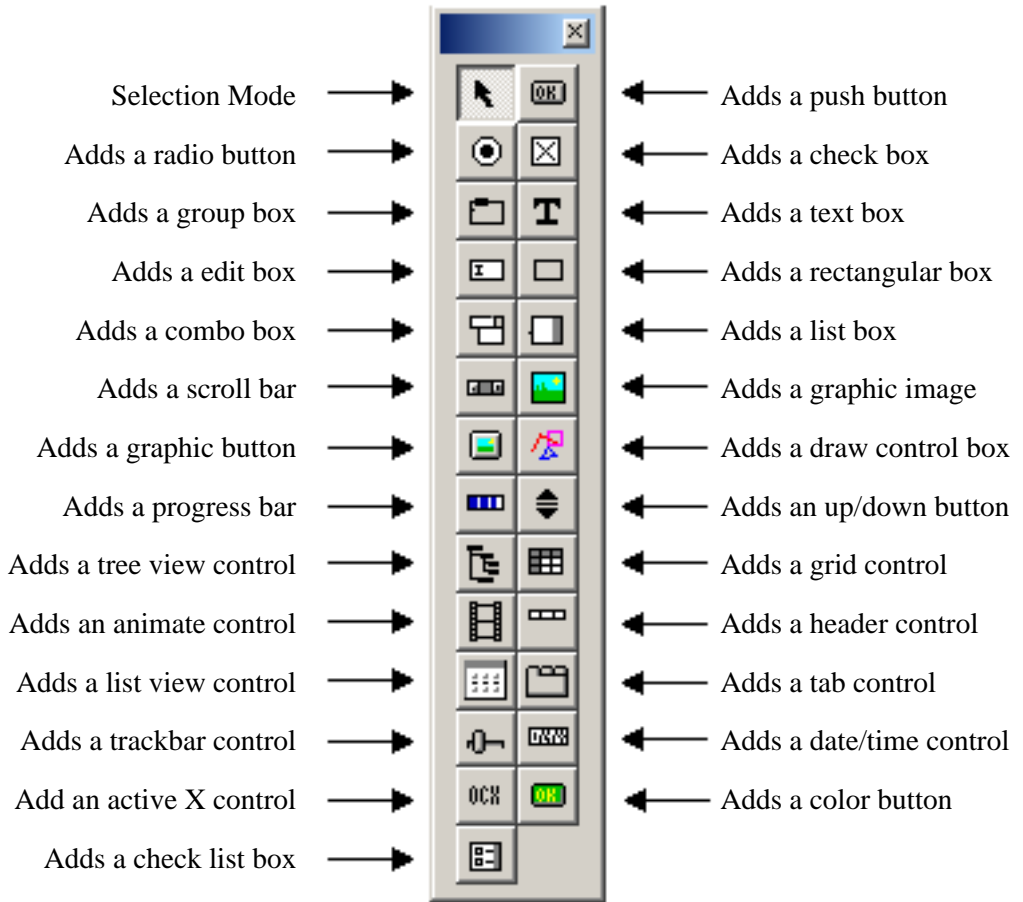
Understanding the Toolbox

The toolbox provides shortcuts for adding different types of controls and boxes to a dialog box. You can also execute each of the toolbox commands from the Dialog Designer Add menu. To use any of the options in the toolbox bar, click the icon, then click in the area of the Dialog Designer where you want that control to appear.

Note

You can drag the toolbox from the Dialog Designer window to size and reshape it as you desire.

The following example describes each toolbox icon:



Toolbox Commands

The following table describes the relationship between the toolbox icons and the menu commands.

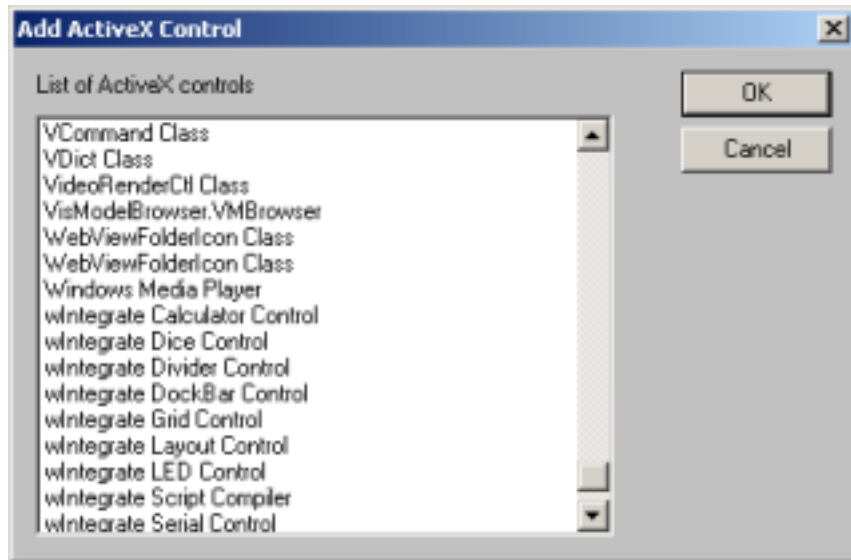
Icon	Menu Command
Select mode	From the Add menu, click Select mode
Push button	From the Add menu, click Standard Controls and then click Push Button
Radio button	From the Add menu, click Standard Controls and then click Radio Button

Icon	Menu Command
Check box	From the Add menu, click Standard Controls and then click Check Box
Group box	From the Add menu, click Standard Controls and then click Group Box
Text box	From the Add menu, click Standard Controls and then click Text Box
Edit box	From the Add menu, click Standard Controls and then click Edit
Rectangular box	From the Add menu, click Standard Controls and then click Rectangle
Combo box	From the Add menu, click Standard Controls and then click Combobox
List box	From the Add menu, click Standard Controls and then click Listbox
Scroll bar	From the Add menu, click Standard Controls and then click Scroll Bar
Graphic Image box	From the Add menu, click Other Controls and then click Graphic
Graphic button	From the Add menu, click Other Controls and then click Graphic Button
Draw Control box	From the Add menu, click Other Controls and then click Graphic Draw
Progress bar	From the Add menu, click Common Controls and then click Progress Bar
Up/Down button	From the Add menu, click Common Controls and then click Up Down Button
Tree view control	From the Add menu, click Common Controls and then click Tree view
Grid control	From the Add menu, click Other Controls and then click Grid

Icon	Menu Command
Animate control	From the Add menu, click Common Controls and then click Animate
Header control	From the Add menu, click Common Controls and then click Header
List View control	From the Add menu, click Common Controls and then click List View
Tab control	From the Add menu, click Common Controls and then click Tab Control
Trackbar control	From the Add menu, click Common Controls and then click Trackbar
Date/Time control	From the Add menu, click Other Controls and then click Date Time
Active X control	From the Add menu, click Active X Control and choose the control from the dialog.
Color button	From The Add menu, click Other Controls and then click Color Button
Check List box	From The Add menu, click Other Controls and then click Check List Box.

Adding an ActiveX Control

To add an ActiveX control in the Dialog Designer, click the Add menu, then click ActiveX Control. The following dialog box appears:



Choose the ActiveX control you want to add to the dialog box. The ActiveX controls used with the Add ActiveX Control box depend upon the software loaded on your PC.

Toolbox Script Commands

When you use the Dialog Designer to create the look of a dialog box, the Dialog Designer automatically creates a wIntegrate script containing the appropriate dialog box commands. For example, if you add a list box, wIntegrate adds the DialogBox ListBox command to the script with the appropriate coordinate, width, depth and style parameters.

Reminder

You must edit the script after you design the dialog box to add the controls and define the commands you want to associate with each control.

The following table describes the wIntegrate script command associated with each toolbox option:

Icon	Script Command
Radio button	DialogBox RadioButton
Check box	DialogBox CheckBox
Group box	DialogBox GroupBox
Text box	DialogBox TextBox
Edit box	DialogBox EditText
Rectangular box	DialogBox Control "Rect"
Combo box	DialogBox ComboBox
List box	DialogBox ListBox
Scroll bar	DialogBox ScrollBar
Graphic Image box	DialogBox Graphic
Graphic button	DialogBox GraphicButton
Draw Control box	DialogBox Control "Draw"
Progress bar	DialogBox ProgressBar
Up/Down button	DialogBox UpDownButton
Animate	DialogBox Animate
Header	DialogBox Header
List View	DialogBox ListView
Tab Control	DialogBox TabControl
Trackbar	DialogBox Trackbar
Tree view	DialogBox TreeView
ActiveX Control	DialogBox AxControl
Date/Time Control	DialogBox DateTime
Grid	DialogBox Grid
Color Button	DialogBox ColorButton
Check list box	DialogBox CheckListBox

Understanding the Status Bar

The Designer Dialog status bar provides the name, position, and size for the selected control, as shown in the following example:



The Name: field shows the name of the currently selected control.

The x,y: field shows the position of the currently selected control in dialog units.

The Size: field shows the width and depth of the currently selected control in dialog units.

When the dialog box is selected the above fields show the information for the dialog box.

As you move the position of the dialog box within the wIntegrate window, the x,y values change. When you size the dialog box by dragging the blue handles, the Size values change. These values are used as parameters with the appropriate commands in the wIntegrate script when you save the dialog box.

Creating a Dialog Box

The following procedure describes how to create a dialog box through the Dialog Designer, the options for different types of boxes and controls, and how to make the dialog box look consistent through aligning, sizing and spacing.

The dialog box in this example includes a group box, a list box, and push buttons. When you push one button, you can generate a full report from a file on the host computer. Another button chooses only certain information that displays in a list box.

Note

The dialog box example uses the UniData demo database.

Before you begin using the Dialog Designer to create your dialog box, you should have an idea of how large to make the dialog box, and where you want to place the controls within the box.

1. Position the Dialog Box

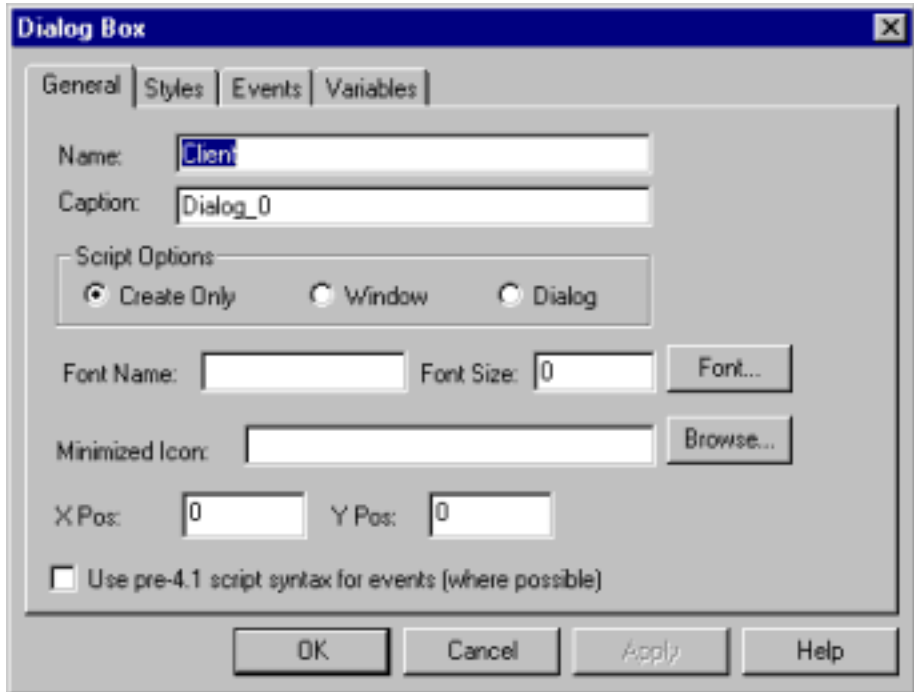
The position of the dialog box in the Dialog Designer is the same as it will appear in the wIntegrate window.

Click on the dialog box if you want to change its position in the window. Notice that blue handles appear when you click on the box. To change the size of the box, position the mouse over the red handle that you want to move, and drag the handle until the size of the box meets your needs. For example, choose the handle on the middle right side of the box and drag the handle to the right to widen the box. To move the entire box within the window, click anywhere inside the dialog box and move the box to the desired position with your mouse.

2. Choose Dialog Box Options

After you position and size the dialog box, choose the options to determine the appearance of the dialog box.

Double-click anywhere within the gray area of the dialog box. The Dialog Box window appears, as shown in the following example:



Name the Dialog Box

The Name field in the Dialog box defines the name for the completed dialog box. Enter CLIENT as the name of the dialog box. The name does not appear in the completed dialog box.

Select a Caption for the Dialog Box

The next field creates the Caption for the dialog box. This name appears in the completed dialog box. Tab to the Caption edit box and enter "Client Information."

Define Script Options

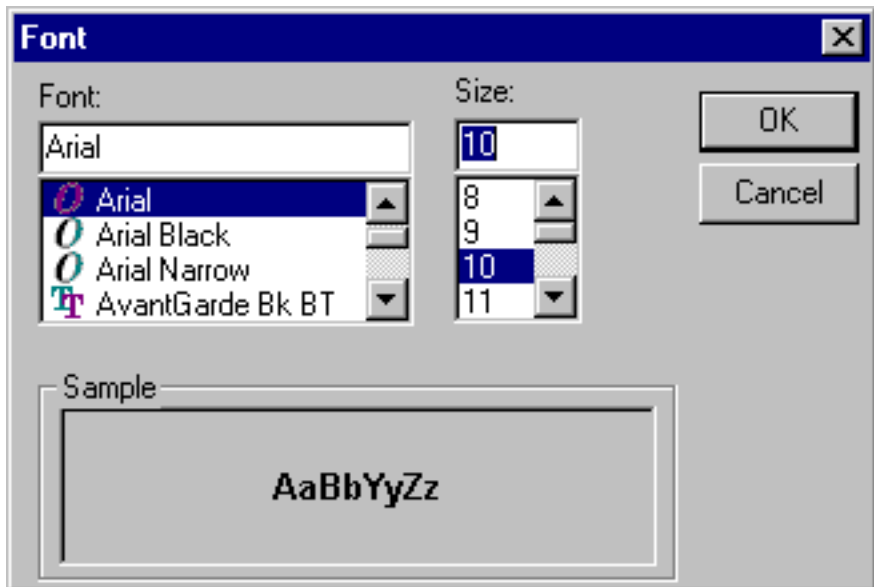
The Script Options specify how the dialog box displays by adding the appropriate DialogBox command to the script. The script options are:

- Create Only - Creates the dialog box, but does not add a command to display the dialog box in the script.
- Window - Displays the dialog box as a window or "modeless" dialog box. Adds the DialogBox Window command to the script.
- Dialog - Displays the dialog box as a modal dialog box, allowing input only to the current dialog box. Adds the DialogBox Show command to the script.

For this example, select Dialog.

Choose Font

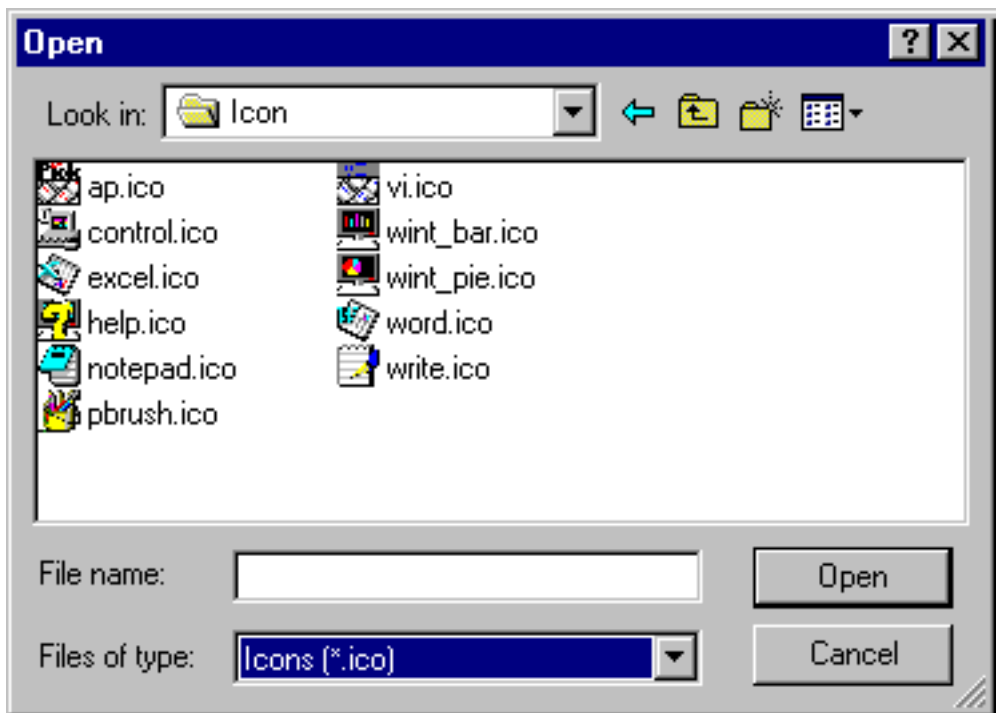
The Font Name and Font Size boxes define the font style and size. In the Font box, select a font style and size. The Sample box displays the style and size of the font you select, as shown in the following example:



When you have selected the font style and size you want to use in your dialog box, then click OK. In this example, the font is Arial and the size is 10 pt.

Choose Minimize Icon

If you want to display an icon in the title bar of the dialog box, click the Minimize Icon box, then click Browse. Change to the directory where you have wIntegrate installed, then double-click icon. Make sure that you select Icons in the Files of type box. The following dialog box appears:

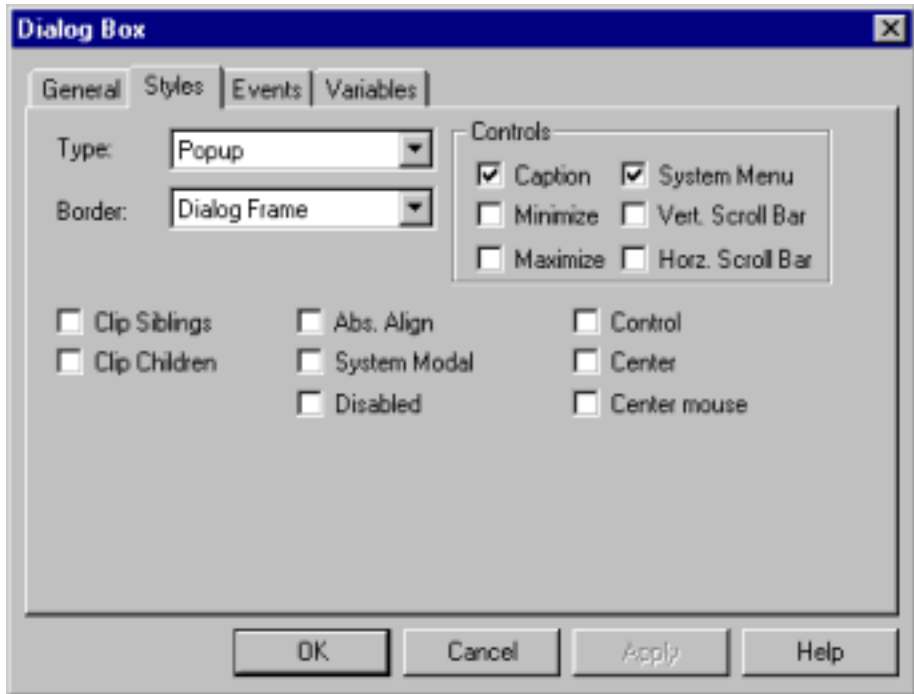


Double-click the Wint_bar.ico icon. The Dialog Box displays the path and display of the icon.

3. Choose Dialog Box Styles

Click the Styles tab to select style options for the dialog box. The following tabbed

page appears:



In the Type box, select available style types. The options are:

- Overlapped - Creates an overlapped window. An overlapped window usually contains a caption and a border.
- Popup - Makes the dialog box a popup window, independent of the current session. Adds the DialogBox style WS_POPUP to the script.
- Child - Creates a dialog box which will be used within another dialog box.

For more information about types of dialog boxes, see the DialogBox Create command in the "Reference - wIntegrate Script Commands" section of this manual.

In this example, select the Popup type.

Choose Border Style

In the Border box, select one of the following border styles from the Border list:

- None - No border appears around the dialog box.
- Thin - A thin line appears around the dialog box.
- Resizing - Creates a window frame that you can size. Adds the DialogBox style `WS_THICKFRAME` to the script.
- Dialog Frame - Makes the dialog box visible. Adds the DialogBox style `WS_VISIBLE` to the script.

Choose Dialog Box Style

You can choose one or a combination of the following dialog box styles:

- Abs. Align - Aligns the dialog box relative to the screen origin rather than the upper-left corner of the parent window. Adds the DialogBox style `DS_ABSALIGN` to the script.
- System Modal - Allows only this dialog box to run. Adds the DialogBox style `DS_SYSMODAL` to the script.
- Disabled - Does not allow input to the dialog box. Adds the DialogBox style `WS_DISABLED` to the script.
- Control - If you are using a dialog box as a child, allows you to tab within the controls in parent and child dialog boxes as if they were one dialog box.
- Center - Centers the dialog box on the screen.
- Center mouse - Centers the dialog box around the position of the mouse.

For more information about dialog box styles, see DialogBox Create in the "Reference -wIntegrate Script Commands" section of this manual.

In this example, no styles are selected.

Choose Controls

You can chose one or a combination of the following controls:

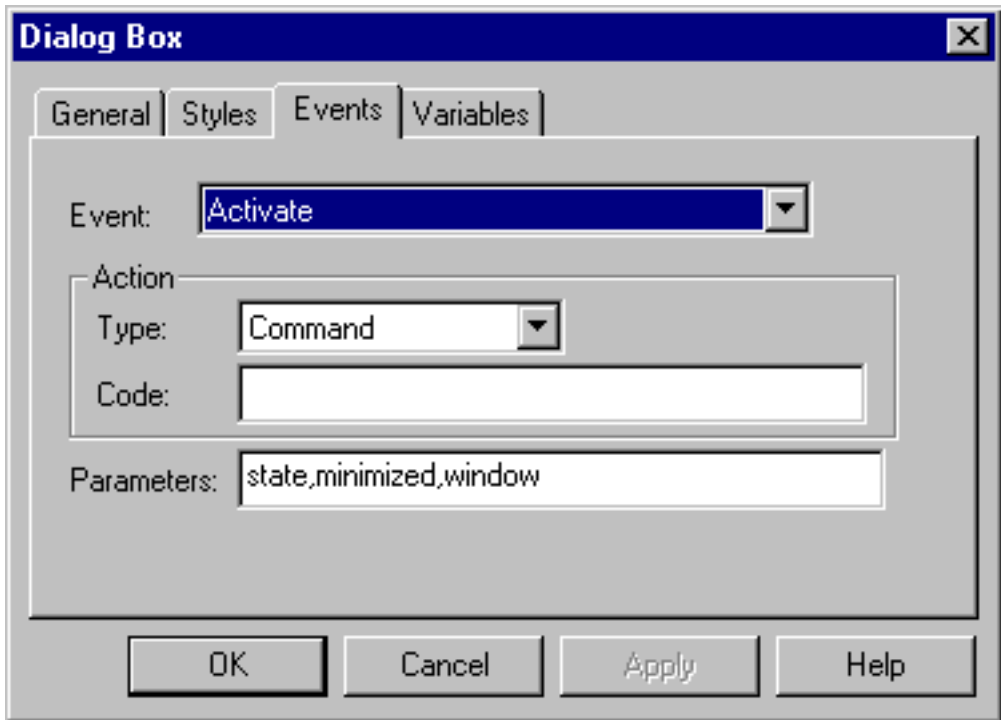
- Caption - Displays the defined caption in the dialog box. Adds the DialogBox style `WS_CAPTION` to the script.
- Minimize - Enables users to minimize the dialog box when it is not being used, by including the minimize button in the dialog box window. Adds the DialogBox styles `WS_MINIMIZEBOX` and `WS_GROUP` to the script.
- Maximize - Enables users to maximize the dialog box after minimizing it, by including the maximize button in the dialog box window. Adds the DialogBox styles `WS_MAXIMIZEBOX` and `WS_TABSTOP` to the script.
- System Menu - Includes the system menu. Adds the DialogBox style `WS_SYSMENU` to the script.
- Vertical Scroll Bar - Includes a vertical scroll bar in the dialog box window. Adds the `WS_VSCROLL` DialogBox style to the script.
- Horizontal Scroll Bar - Includes a horizontal scroll bar in the dialog box window. Adds the `WS_HSCROLL` DialogBox style to the script.

For more information about dialog box styles, see DialogBox Create in the "Reference -wIntegrate Script Commands" section of this manual.

Caption and System Menu are the default controls. For this example, choose minimize and maximize by selecting the box next to each control.

4. Choose Dialog Box Events

Click the Events tab if you want to define a wIntegrate command to execute each time you show the dialog box. The following screen appears:



Choose Event

In the Event list, choose the type of event you want.

For example, if you have a list box that displays the names Paul, Barry, and Susan, you can add an `InitCommand` to display Barry each time you show the dialog box. In this case, the Initialise field would contain a command similar to:

```
InitCommand 'Select MyDialog,names,"Susan"'
```

For information on the `DialogBox InitCommand`, see the "Reference - wIntegrate Script Commands" section of this manual.

Select Type of Event

In the Type list, select the type of event. The valid options are:

- **Command** - Specifies that the Event Code is a wIntegrate command. Quotation marks are automatically inserted in the text you enter in the Event Code when wIntegrate writes the code in the script.
- **Subroutine** - Indicates that the Event Code is a host subroutine. The Event code should contain the name of a subroutine to call within the script.
- **Expression** - Indicates that the Event Code is an expression. Any entry in the Event Code is written literally to the OnEvent command in the resulting script.

Define Event Code

wIntegrate executes the Event Code when an event occurs. The code is written to the script, but is modified when saved according to the type of event.

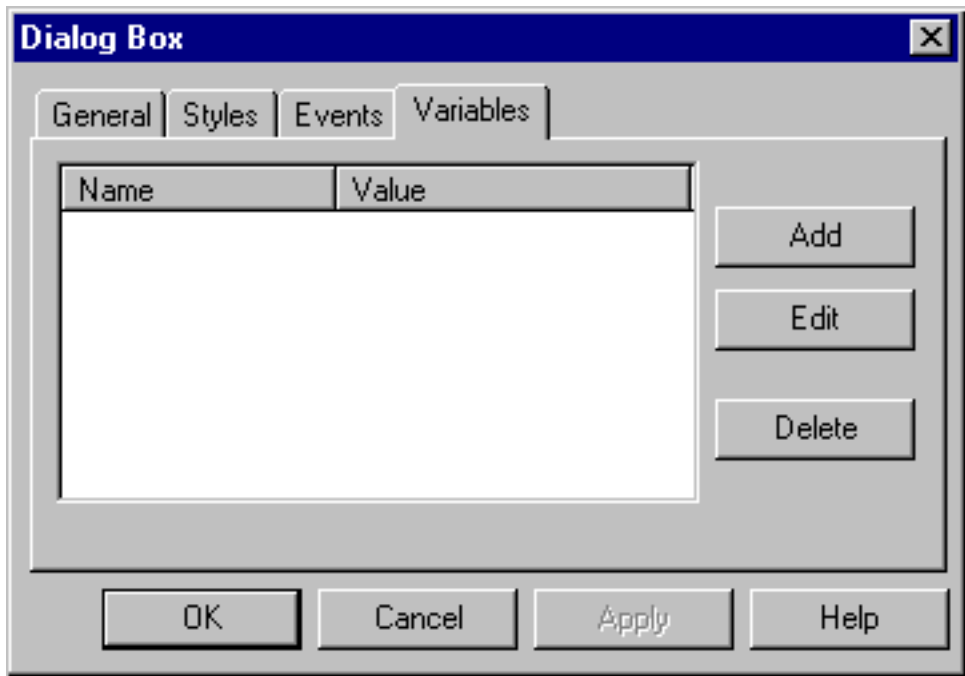
Define Event Parameters

Events can sometimes accept parameters. For example, the Grid control contains a validate cell event, which returns the row and column of the cell to be validated.

If the event accepts parameters, enter the name of the script variable that will be set with the parameter value. You can change the names of the variables or remove parameters that are of no use.

5. Define Variables

Click the Variables tab to define local variables for the dialog box. The following screen displays:



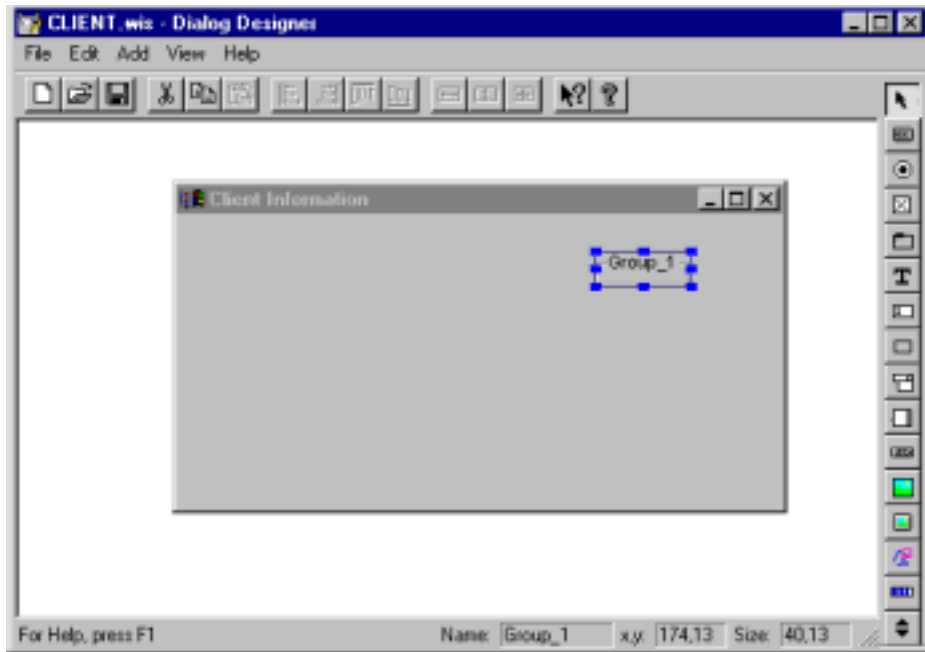
Variables enable you to create a variable associated with the dialog box. wIntegrate deletes these variables when the dialog box is deleted. Variables cannot conflict with other global variables you may be using.

Review Dialog Options

Review the dialog options. If you are satisfied with the options selected, click OK. If you want to exit without saving changes, click Cancel.

6. Add a Group Box

The CLIENT dialog box will contain two group boxes. To add the groups boxes, click the group box icon, or choose Group Box from the Add menu. Then click in the upper half of the Designer Dialog window. A box entitled Group_1 appears in the window, as shown in the following example:



Size the group box by dragging the blue handles until the box is the size you desire. In this example, the group box will occupy approximately the top one-third of the dialog box.

Add Group Box Options

Double-click the group box to access the Group Box Options window. The following window appears:



Add Name

The Group Box name defines the name of the group box in the wIntegrate script. Enter "Report" in the name field.

Add Caption

The Caption is the name that appears in the group box. Enter "Full Report" in the Caption field.

Choose Control Styles

The following group box control styles are available from the Group Box Options window:

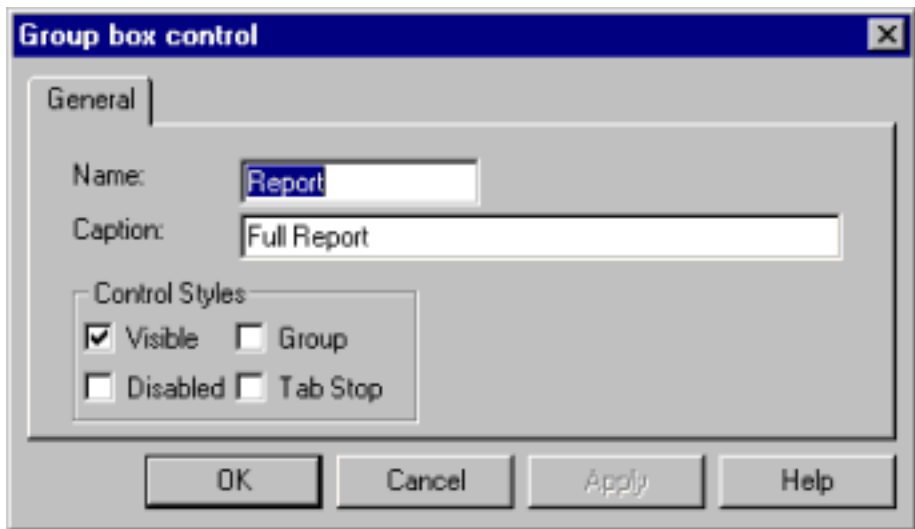
- Visible - Makes the group box visible.
- Disabled - Disables input to the group box. Adds the GroupBox style WS_DISABLED to the script.
- Group - Makes this control the first one in a group box. Up and down arrows move to the previous or next control in the group. Adds the GroupBox style WS_GROUP to the script.

- Tab Stop - Enables the tab or back tab to move to the previous or next control within the group box. Adds the GroupBox style WS_TABSTOP to the script.

The default style of Visible is used in this example.

Review Group Box Options

The Group Box Options window should now look like this:

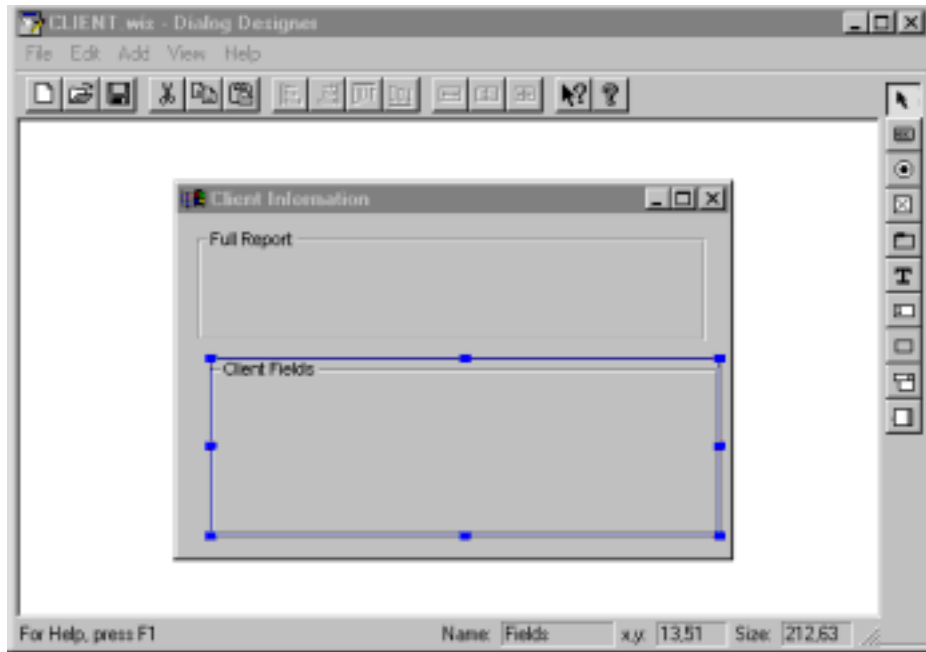


Click OK to return to the Dialog Designer window.

7. Add Another Group Box

Add a second group box occupying the lower two-thirds of the Dialog Designer window following the steps outlined previously. Leave enough room at the bottom of the window to add a push button. The name of this group box is Fields, and the Caption is Client Fields. Don't worry about perfectly aligning the second group box with the first group box. This is described later.

The Dialog Designer window should now look like this:



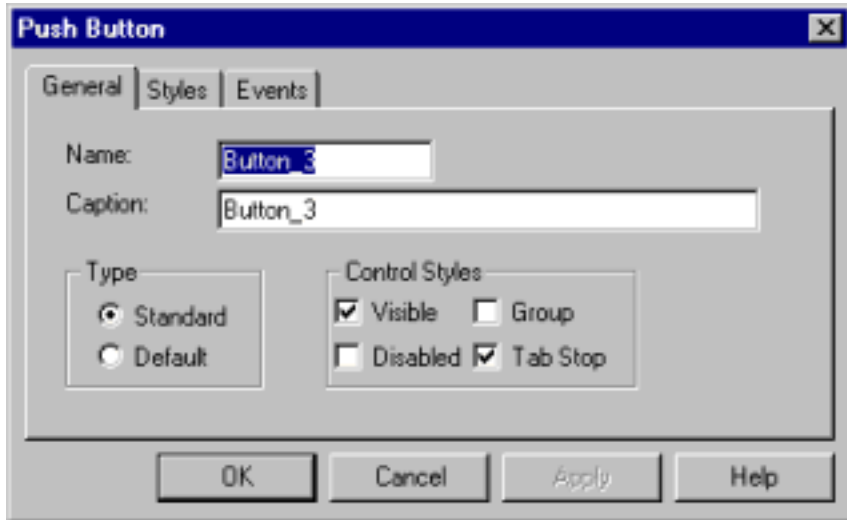
8. Add Controls to the First Group Box

The first group box contains a push button and a text box. The text box provides information about what the push button will execute, and the push button executes a report from the CLIENTS file on the host computer.

Add a Push Button

Click the push button icon from the toolbox, or click Push Button from the Add menu. Click in the right side of the group box with the caption Full Report. If desired, change the size of the push button using the blue handles, and move the push button using the mouse.

Double-click the push button to bring up the Push Button Options box, as shown in the following example:



Add Name

The Push Button name defines the name of the push button in the wIntegrate script. Enter "Report_button" in the name field.

Add Caption

The Caption is the name that appears in the push button. Enter "Process Report" in the Caption field.

Add Click Event

Select the Events tabs and choose the Click events from the Event drop down list and choose command from the action type drop down list. Adding code to the the event writes a DialogBox OnEvent command to the script.

In this example, the push button executes an ECL command on the UniData host demo database.

In the Code field, type:

Enter 'LIST CLIENTS BY NAME NAME ADDRESS CITY'

Choose Type

Choose one of the following push button types:

- **Standard** - Designates the push button as a standard push button. Adds the `DialogBox PushButton` command to the script.
- **Default** - Designates the push button as a default push button, indicating the default action for the dialog box. Adds the `DialogBox DefPushButton` command to the script.

In this example, use the Standard push button type.

Choose Control Style

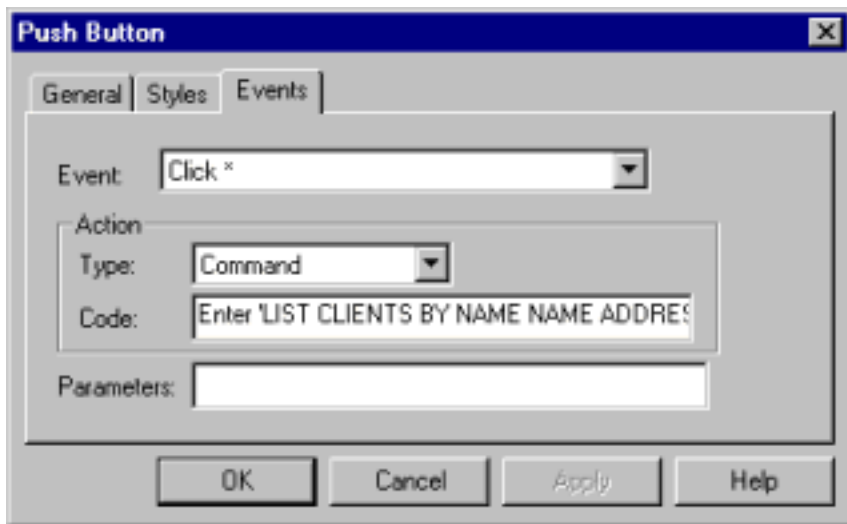
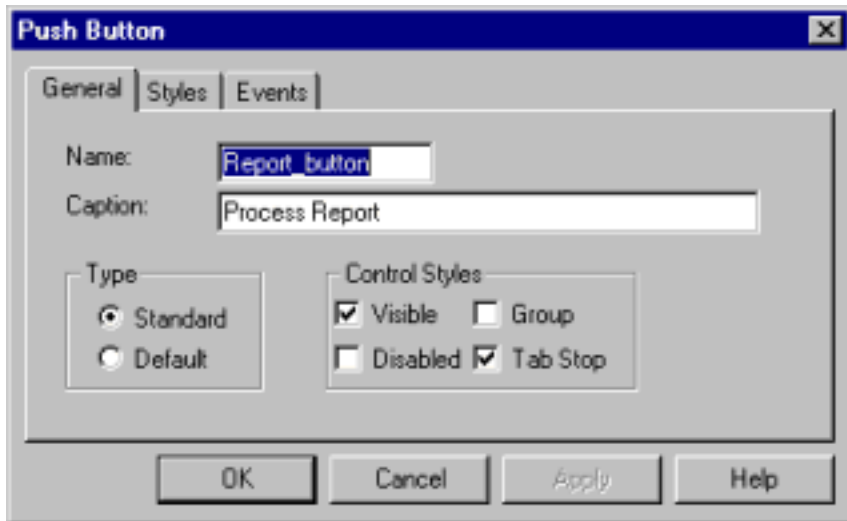
The following push button control styles are available from the Push Button Options window:

- **Visible** - Makes the push button visible.
- **Disabled** - Disables the push button. Adds the `DialogBox PushButton Style WS_DISABLED` to the script.
- **Group** - Makes this push button control the first in a group. Up and down arrows move to the previous or next control in the group.
- **Tab Stop** - The tab or back tab moves to the next or previous control. Adds the `DialogBox PushButton Style WS_TABSTOP` to the script.

The default styles of Visible and Tab Stop are used in this example.

Review Push Button Options

The Push Button Options window should now look like this:



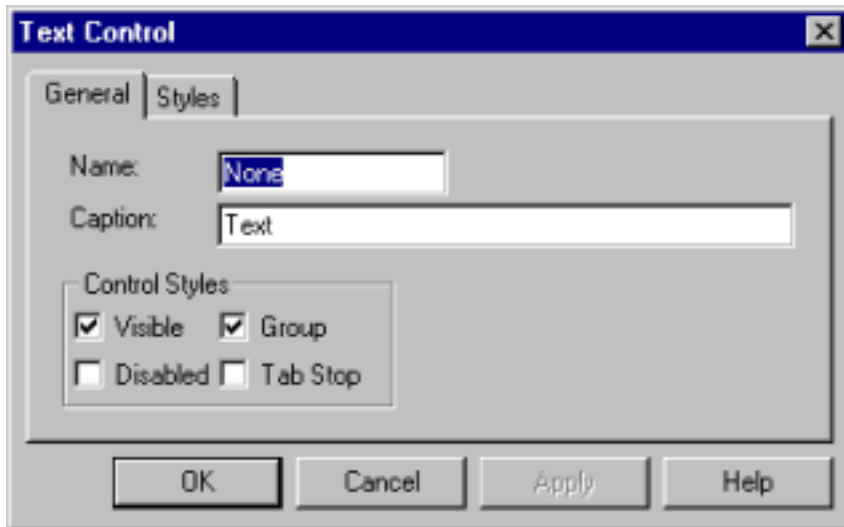
Click OK when the Push Button Options window is complete. The Dialog Designer window appears.

If necessary, size the push button to accommodate the text by dragging the blue handles.

Add a Text Box

Add a text box to the group box by clicking the text box icon, or choose Text from the Add menu. Click to the left of the push button the Full Report group box. If desired, size the text box by dragging the blue handles, and move the text box using the mouse.

Double-click the text box. The Text Options window appears, as shown in the following example:



Add Name

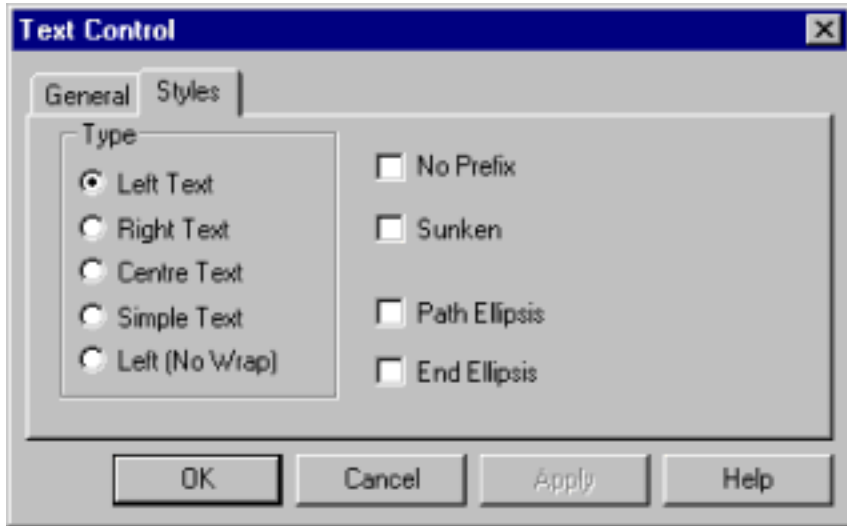
The Text Box name defines the name of the text box in the wIntegrate script. Enter "Text_box" in the name field.

Add Caption

The Caption is the name that appears in the text box. Type "Generate a Full Report" in the Caption field.

Set Text Styles

Click on the Styles Tab, and the following page appears:



Choose Text Type

The following text types are available:

- Left text - Left justifies the text, and automatically wraps the text. Adds the DialogBox LText command to the script.
- Right text - Right justifies the text within the box. Adds the DialogBox RText command to the script.
- Centre text - Centers the text with the text box. Adds the DialogBox CText command to the script.
- Simple text - Creates a simple text box with no formatting. Adds the DialogBox Control command to the script, with a static style of SS_SIMPLE.
- Left (no wrap) - Left-justifies the text, but truncates the text if it exceeds the size of the box. Adds the DialogBox Control command to the script, with a static style of SS_LEFTNOWORDWRAP.

Click on Centre Text to center the text.

Choose Text Styles

You can underline a letter in the Caption by placing the ampersand character (&) before the letter you want to underline. If you do not want a prefix, click the No Prefix check box.

Choose Control Style

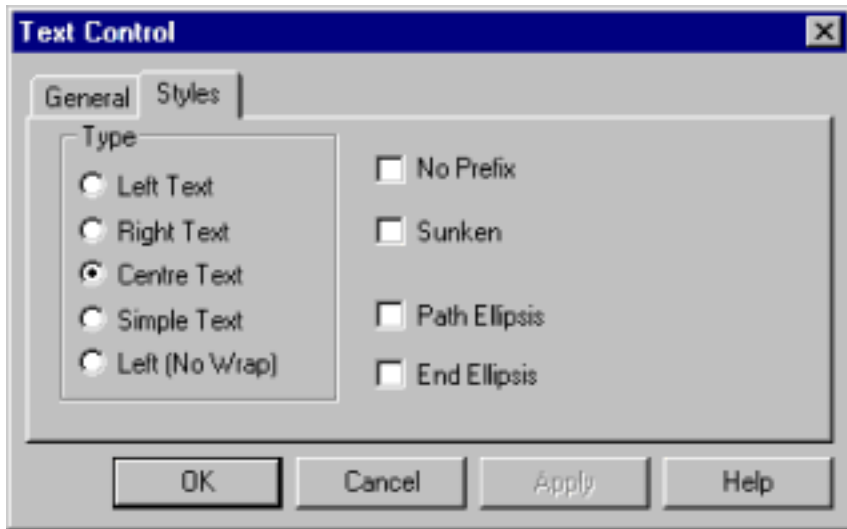
The following text box control styles are available from the Text Box Options window:

- Visible - Makes the text visible with the text box.
- Disabled - Disables the text within the box. Adds the `WS_DISABLED` style to text in the script.
- Group - Makes this text box the first control in a group. Up and down arrows move to the previous or next control in the group.
- Tab Stop - The tab or back tab moves to the next or previous control. Adds the `WS_TABSTOP` style to the text box in the script.

This example uses the default text box control styles of Visible and Group.

Review Text Control Styles

The Text Styles window should now look like this:



Click OK to return to the Dialog Designer window.

Enlarge Text Box

If the text box is not large enough to accommodate the text, size the window by dragging the blue handles.

9. Add Controls to the Second Group Box

The group box with the caption "Client Fields" will contain a push button, a text box, and a list box.

Add a List Box

Click the list box icon or choose List Box from the Add menu. Click the left side of the Client Fields list box. Then size the box by dragging the blue handles to occupy the left half of the group box.

Double-click the list box to bring up the List Box Options window, as shown in the following example:

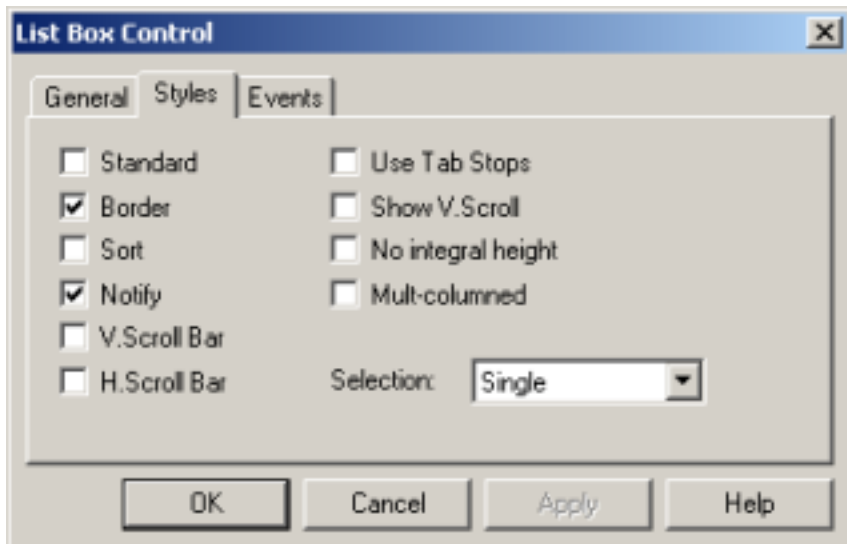


Add Name

The List Box name defines the name of the list box in the wIntegrate script. Enter "List_box" in the name field.

Choose List Box Styles

Click on the Styles Tab, and the following page appears:



The following list box styles are available:

- **Standard** - Creates a standard list box and adds the LBS_STANDARD style to the DialogBox ListBox command in the script. The LBS_STANDARD style is a combination of the border, sort, notify, and vertical scroll bar styles. Notice that all of these boxes are checked when you select the Standard style.
- **Border** - Places a border around the list box, and adds the WS_BORDER style to the DialogBox ListBox command in the script when you choose another style in addition to Border.
- **Sort** - Sorts the contents of the list box alphabetically. Adds the LBS_SORT style to the DialogBox ListBox command in the script.
- **Notify** - Notifies wIntegrate when you click the list box. Adds the LBS_NOTIFY style to the DialogBox ListBox command in the script when you choose another style in addition to Notify.
- **V.Scroll Bar** - Adds a vertical scroll bar to the list box. Adds the WS_VSCROLL style to the DialogBox ListBox command in the script.
- **H.Scroll Bar** - Adds a horizontal scroll bar to the list box. Adds the WS_HSCROLL style to the DialogBox ListBox command in the script.
- **Use Tab Stops** - Enables use of tab stops in the list box. Adds the LBS_USETABSTOPS style to the DialogBox ListBox command in the script.
- **Show V.Scroll** - Displays the vertical scroll bar, even if there is not enough information in the list box to warrant a scroll bar. Adds the LBS_DISABLENOSCROLL command to the DialogBox ListBox command in the script.
- **No integral height** - If this style is specified, the last line of the list may not completely display. Adds the LBS_NOINTEGRALHEIGHT style to the DialogBox ListBox command in the script.
- **Multi-columned** - Specifies to display multiple columns in the list box. Adds the LBS_MULTICOLUMN style to the DialogBox ListBox command in the script.

- Selection - The selection type for the list. None, Single, Multiple or Extended.

In this example, click the V.Scroll Bar and Show V.Scroll check boxes. These styles, along with the defaults of Border and Notify, are now selected.

Add Control Styles

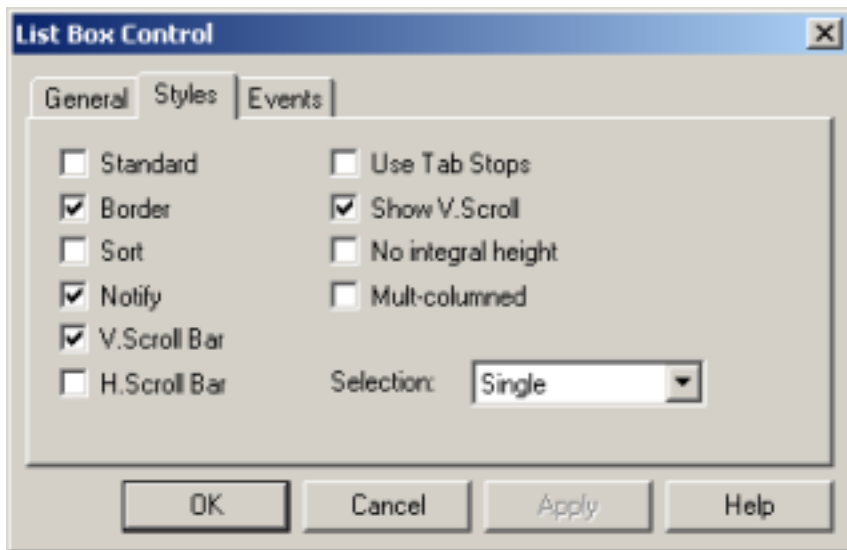
The following list box control styles are available from the List Box Options dialog box:

- Visible - Makes the list box visible in the dialog box.
- Disabled - Disables the list box. Adds the `WS_DISABLED` style to the `DialogBox ListBox` command in the script.
- Group - Makes this list box the first control in a group. Up and down arrows move to the previous or next control in the group. Adds the `WS_GROUP` style to the `DialogBox ListBox` command in the script.
- Tab Stop - The tab or back tab moves to the next or previous control. Adds the `WS_TABSTOP` style to the `DialogBox ListBox` command in the script.

This example uses the default list box style of Visible.

Review List Box Options

The List Box Options dialog box general and styles pages should now look like this:



Click OK to return the Dialog Designer dialog box.

Add a Push Button Control

Click on the Push Button icon, or choose Push Button from the Add menu, then click

on the right top side of the Client Fields box.

Double-click the push button to access the Push Button Options dialog box. Following the instructions outlined in Step 5, change the name of the push button to “Fields_button,” and change the Caption box to “Client Fields.”

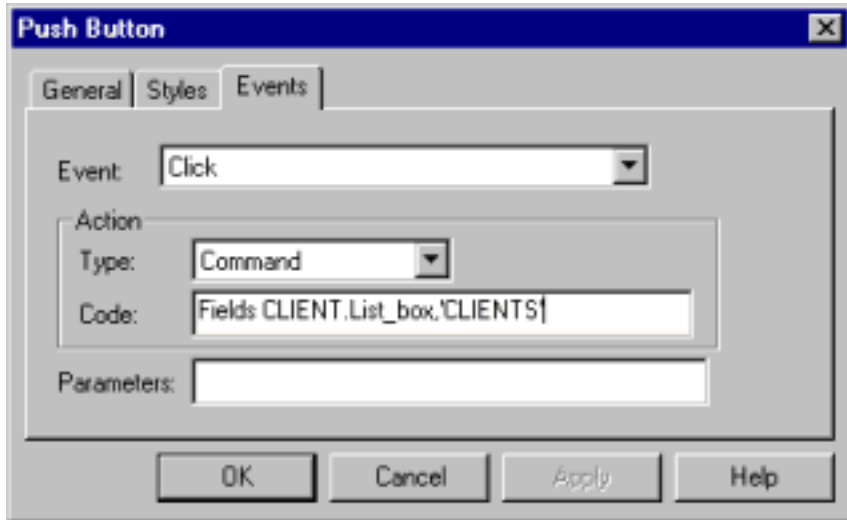
Add a Click Event

In this example, a portion of the click event is added through the Dialog Designer window. A later step illustrates how to complete the action by editing the script.

The click event in this example uses the Fields server library command. This command lists fields or records from the host, using the dialog box name and list box name parameters, along with the name of the file on the host. For more information about the Fields command, see the “Reference - wIntegrate Script Commands” section in this manual.

The name of the completed dialog box will be CLIENT. Combine this name with the name of the list box to pass to the Fields command, then pass the CLIENTS file name. Type "Fields CLIENT.List_box,'CLIENTS'" in the Click Event Action Code box. As the action type is Command you do not need to type in the double quotation marks around the command. If the action type had been specified as expression the double quotes would have been required.

For this example, use the default type of Standard and the default control styles of Visible and Tab Stop. The Push Button Events window should look like this:



Click OK to return to the Dialog Designer window.

Add a Text Box

Finally, add a text box to the Client Fields box. Click on the text box icon, or choose Text from the Add menu. Click in the Client Fields box below the Client Fields push button.

Double-click the text box to access the Text Options window. Using the instructions described in Step 5, change the Name box to “Text2_box.” In the Caption box, type "List Fields in CLIENTS File." Click Centre Text in the Type section of the Styles page to center the text.

Click OK to return to the Dialog Designer window. If necessary, size the text box to accommodate the text.

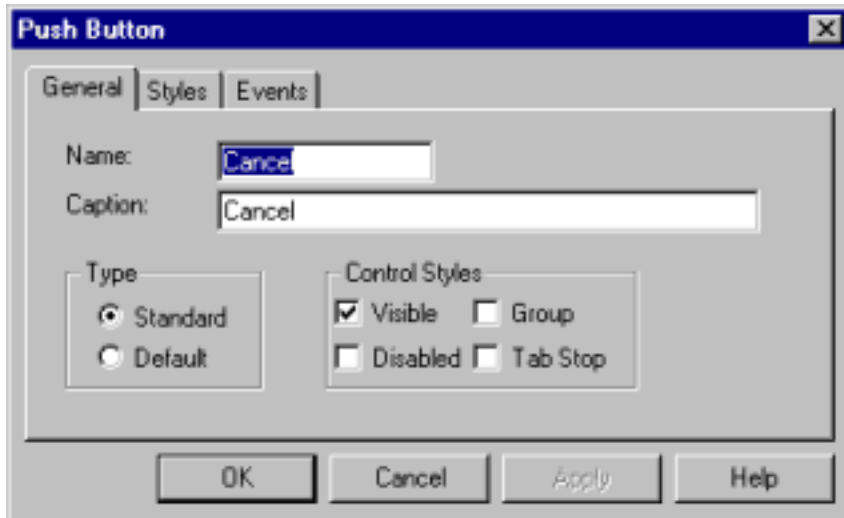
10. Add a Cancel Button

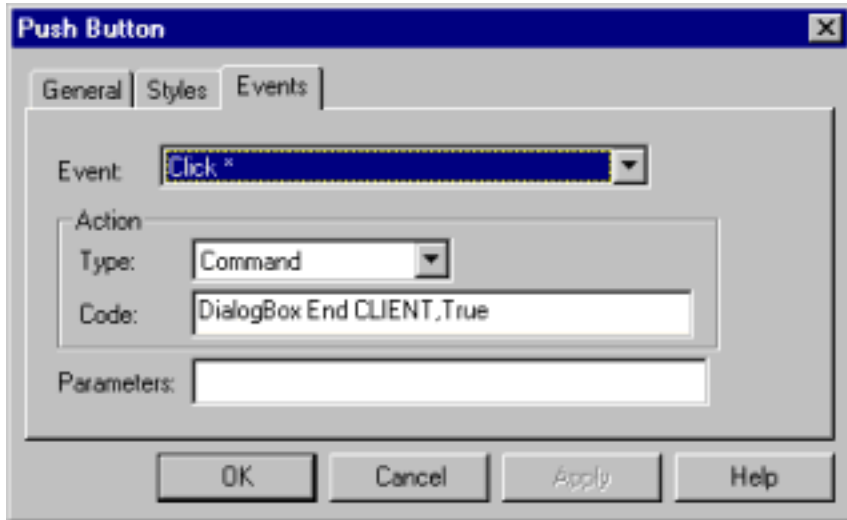
Next, add a Cancel button to the dialog box. Click the Push Button icon, or choose Push Button from the Add menu. Click in the lower right corner of the dialog box to place the Cancel button.

Double-click the Push Button to access the Push Button Options window. Using the instructions outlined in Step 5, change both the Name and Caption boxes to Cancel. Cancel is a special control name in wIntegrate that is activated by a mouse click on the button, by pressing the Escape key, or by choosing the Close option from the system menu.

Click Action uses the DialogBox End command to close the dialog box. Type "DialogBox End CLIENT,True". For more information on the DialogBox End command, see the "Reference -wIntegrate Script Commands" section in this manual.

The Push Button Options box General and Events pages should now look like the following example:





Click OK to return to the Dialog Designer window.

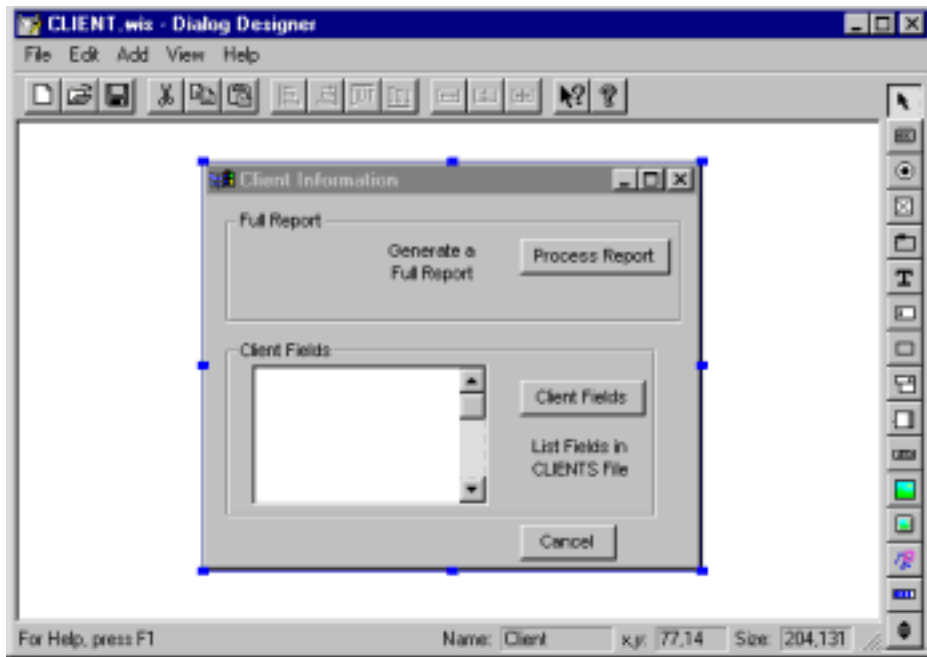
11. Align the Controls

Next, align the two group boxes to the left. First, click the Client Fields box. Blue handles appear around the box. Then, click the Full Report box with the right mouse button. The blue handles in the Client Field box are dimmed, and the blue handles appear around the Full Report box.

Choose the Align Left toolbar icon, or choose Align from the Edit menu, then click Left. The two group boxes are now aligned with each other based on the left position of the Full Report box. The box you want to align other boxes with must be selected last.

Next, align each push button based on the Process Report button. Select the Cancel button with left mouse button, then the Client Fields button with the right mouse button, and finally the Process Report button. Choose the Align Left toolbar icon, or choose Align from the Edit menu, then click Left.

The Dialog Designer should now look like the following example:



12. Save the Dialog Box

Once the dialog box appears the way you want it to, save the dialog box by choosing Save As from the File menu. Use CLIENT as the name of the dialog box with .wis as the file type, indicating a wIntegrate script.

13. Edit the Script

Most of the functionality for this dialog box is now complete. To complete this dialog box, you must edit the script created by Dialog Designer.

Warning!

If you alter the script between the DialogBox Create and the DialogBox End Create commands, then open the script in Dialog Designer, you may lose lines of your altered script. This is because Dialog Designer will remove any lines it does not recognize.

This example uses Microsoft Notepad as the text editor, but you can use any editor or word processing application to edit the script.

The Dialog Designer creates the following script:

```
* Dialog box CLIENT
* Saved On Wed 07 Feb 2001, 15:10

DialogBox Create CLIENT 77,14,204,131
Caption "Client Information"
Style WS_CAPTION | WS_POPUP | WS_VISIBLE | WS_SYSMENU | WS_MINIMIZEBOX
| WS_MAXIMIZEBOX | DS_MODALFRAME | WS_TABSTOP | WS_GROUP
Font 8,"Arial"
MinIcon 'C:\Program Files\wIntegrate\Icon\wint_bar.ico'
GroupBox "Full Report",Report,8,5,190,41
GroupBox "Client Fields",Fields,8,51,179,63
Pushbutton "Process Report",Report_button,131,15,63,13
OnEvent Report_button,"Click","Enter 'LIST CLIENTS BY NAME NAME
ADDRESS CITY'"
CText "Generate a Full Report",Text_box,71,15,46,19
ListBox List_box,19,60,99,53,LBS_NOTIFY | LBS_DISABLENOSCROLL |
WS_BORDER | WS_VSCROLL
Pushbutton "Client Fields",Fields_button,131,66,52,13
OnEvent Fields_button,"Click","Fields CLIENT.List_box,'CLIENTS'"
CText "List Fields in CLIENTS File",Text2,131,85,51,17
Pushbutton "Cancel",Cancel,131,117,40,13
OnEvent Cancel,"Click","DialogBox End CLIENT,True"

EndCreate
DialogBox Show CLIENT
```

You may want to review this script against the DialogBox commands described in the “Reference - wIntegrate Script Commands” section of this manual. The Dialog Designer creates the script using commands and parameters based on the options you choose.

This script will run now in wIntegrate, but the Client Fields button will not produce any results because it uses a server library command, and the server library has not been loaded in the script. In addition, the dialog box will remain in memory, since there is no command in the script to delete it from memory when it is closed.

Add the Server Library

To add the server library, place the cursor after the second line in the script and press ENTER to create a new line. Then insert the following line:

```
Library 'wintsys\lib\server', S
```

The S option makes the caches the subroutines and functions within the server library, so they may be called from any script within the current session.

```
* Dialog box CLIENT
* Saved On Wed 07 Feb 2001, 15:10

Library 'wintsys\lib\server', S

DialogBox Create CLIENT 77,14,204,131
Caption "Client Information"
...
```

Close the Server

After the list box displays the requested information from the CLIENTS file, close the server. Change the line:

```
OnEvent Fields_button, "Click", "Fields CLIENT.List_box,'CLIENTS'"
```

To:

```
OnEvent Fields_button, "Click", "Notify 'CloseServer'; Fields
CLIENT.List_box,'CLIENTS'"
```

There are two new commands prefixed to the original script statement:

Notify - A server library command that causes the script command past as its argument to be run when the next action on the server completes.

CloseServer - A server library command that closes the server. You must close the server before any subsequent action or input from the server commands will work.

Delete the Dialog from Memory

Finally add a script command to delete the dialog box from memory when you close it. If you do not delete the script from memory, wIntegrate displays an error message indicating that the dialog box is already in place when you run the script.

To delete the dialog box from memory, add the following line to the end of the script:

DialogBox Delete CLIENT

The completed script should now look like the following example:

```
* Dialog box CLIENT
* Saved On Wed 07 Feb 2001, 15:10

Library 'wintsys\lib\server', S

DialogBox Create CLIENT 77,14,204,131
Caption "Client Information"
Style WS_CAPTION | WS_POPUP | WS_VISIBLE | WS_SYSMENU | WS_MINIMIZEBOX
| WS_MAXIMIZEBOX | DS_MODALFRAME | WS_TABSTOP | WS_GROUP
Font 8,"Arial"
MinIcon 'D:\Dev\wIntegrate\Icon\wint_bar.ico'
GroupBox "Full Report",Report,8,5,190,41
GroupBox "Client Fields",Fields,8,51,179,63
Pushbutton "Process Report",Report_button,131,15,63,13
OnEvent Report_button,"Click","Enter 'LIST CLIENTS BY NAME NAME
ADDRESS CITY'"
CText "Generate a Full Report",Text_box,71,15,46,19
ListBox List_box,19,60,99,53,LBS_NOTIFY | LBS_DISABLENOSCROLL |
WS_BORDER | WS_VSCROLL
Pushbutton "Client Fields",Fields_button,131,66,52,13
OnEvent Fields_button,"Click","Notify 'CloseServer';Fields
CLIENT.List_box,'CLIENTS'"
CText "List Fields in CLIENTS File",Text2,131,85,51,17
Pushbutton "Cancel",Cancel,131,117,40,13
OnEvent Cancel,"Click","DialogBox End CLIENT,True"

EndCreate
DialogBox Show CLIENT
DialogBox Delete CLIENT
```

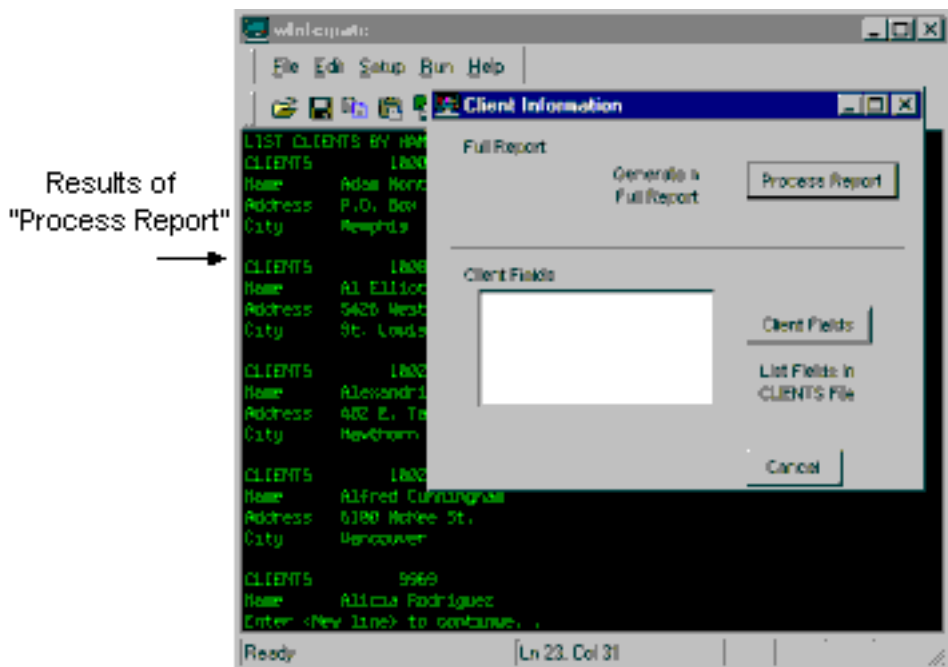
After you have made the changes, save the script.

14. Run the Script

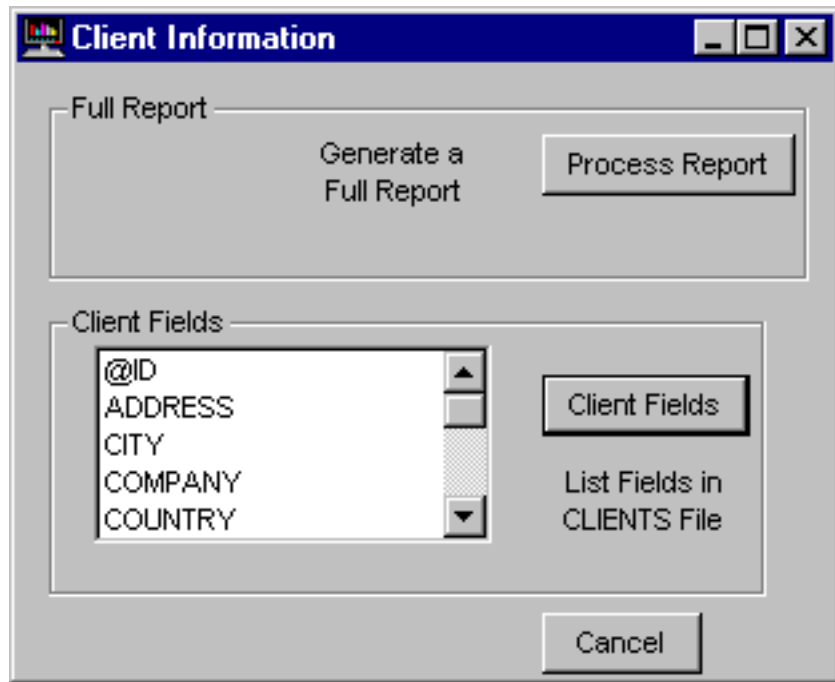
You can now run the script. You must be in a wIntegrate session in the account from which you can access the report data. In this example, you should be at the UniData ECL prompt.

From the wIntegrate Run menu, and change directories to the directory where you saved the script.

The Client Information box should appear in the wIntegrate window. The push buttons should each run the commands assigned to them, as shown in the following examples:



The above example shows the results of pressing the process fields report.



The above example shows the results of pressing the Client Fields button.

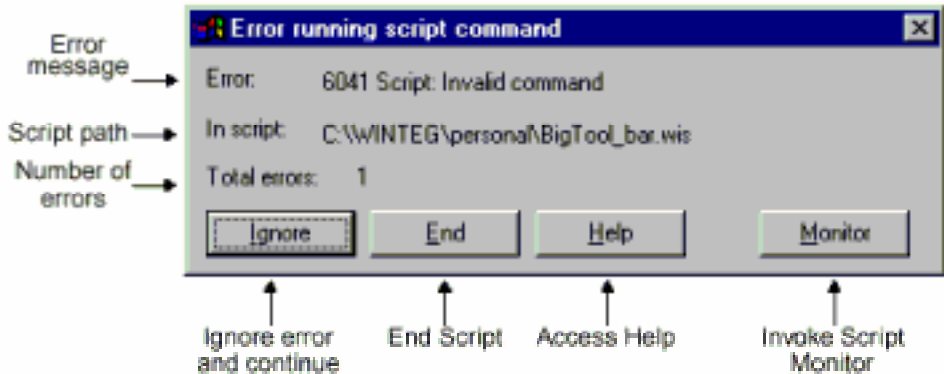
Appendix C

The Script Monitor

When wIntegrate encounters an error in a script, it displays an error message box with a brief description of the error. This box includes an option to display the script monitor, which provides detailed information about the error. You can also display the script monitor when you terminate a script using the break key, or by using the Debug command within a script.

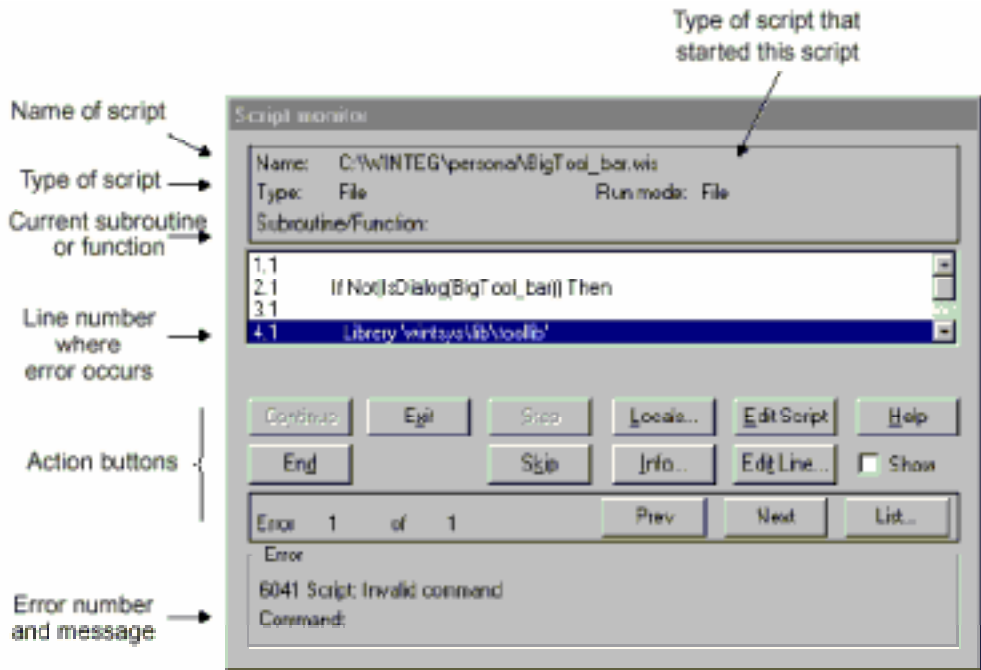
The Script Error Box

The following example illustrates the error message box that wIntegrate displays when an error is found in a script:



The Script Monitor

The script monitor appears when you click Monitor in the Script Error box. The script monitor provides detailed information about errors in a script. The following example illustrates the Script Monitor dialog box when an error appears in the script:



The following table describes each action button:

Action Button	Description
Continue	Continue running the current script. If this action button is disabled, use Skip to skip over the current line, then click Continue again.
End	Stop executing the current script and any script that may have called it.

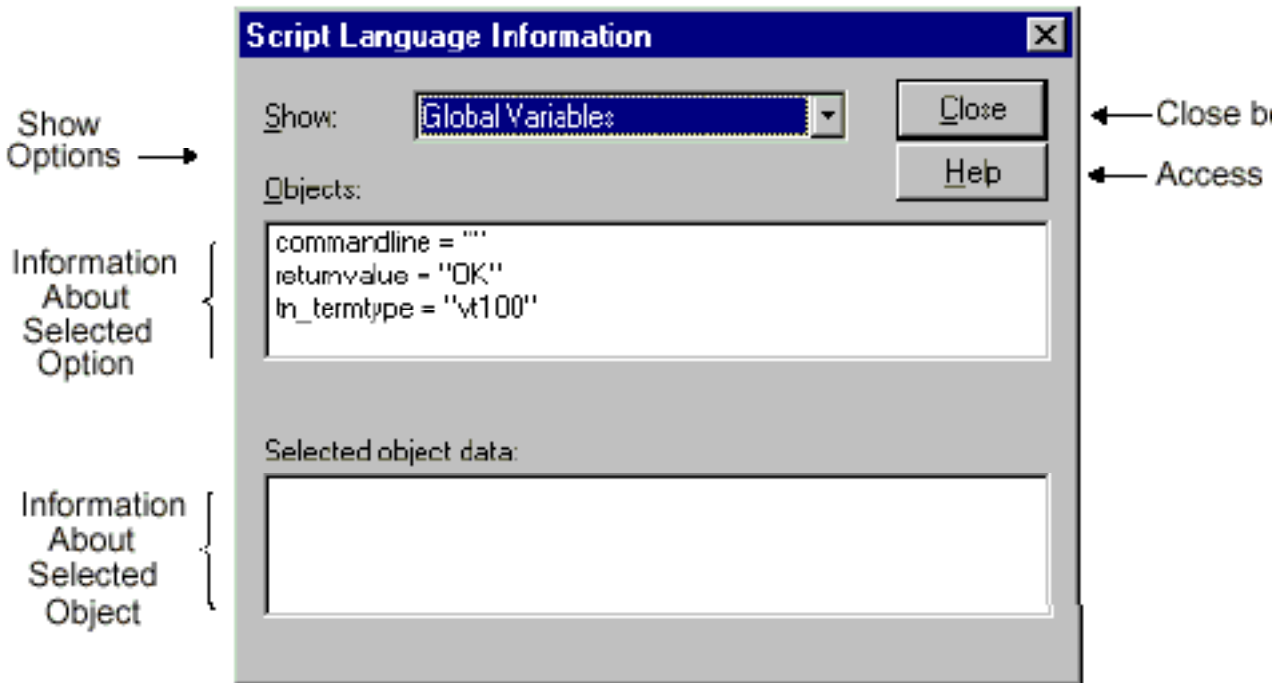
Action Button	Description
Exit	Stop executing the current script and return control to the calling script (if any).
Step	Execute the current line in the script without closing the script monitor.
Skip	Skip the current line in the script.
Locals	Display local variables.
Info	Display the Script Information box to show more detailed information.
Edit Script	Edit the current script file. The script must already have been saved to disk.
Edit Line	Edit the current line of the script. You can then execute the line again. Any changes made using this option are not saved to the original version of the script.
Help	Access help about the script monitor.
Show	If you select this box, the Script Monitor dialog box rather than the Script Error box appears automatically when wIntegrate encounters an error in a script, rather than the Script Error box.
Prev	If more than one script is stopped at the same time, displays the previous script in the stopped list.
Next	If more than one script is stopped at the same time, displays the next script in the stopped list.
List	If more than one script is stopped at the same time, displays summary information of all stopped scripts.

The Script Information Box

The Script Information box displays information about global variables, scripts, soft

menus, user dialog boxes, cached libraries, extension subroutines, extension functions, and triggers. The Script Information box is produced by the ScriptInfo menu command and can be shown at any time by executing the Show ScriptInfo command.

The following example illustrates information displayed about global variables in the Script Info dialog box:



The following table describes the available Show options:

Show Option	Description
Global Variables	Displays all global variables defined in the wIntegrate session. If you select one of the global variables displayed, the Select object data portion of the dialog box displays the value for that variable.

Show Option	Description
Scripts	Displays the active script names in the Objects box. If you select a script, the Select object box displays detailed information about that script.
Soft Menus	Lists the soft menus in the Objects portion of the dialog box. If you select a soft menu, wIntegrate displays detailed information about the menu in the Select object data portion of the box.
User Dialog Boxes	Displays all currently defined dialog boxes. If you select one of the dialog boxes, wIntegrate displays the control types and names.
Cached Libraries	Displays the full path of cached libraries.
Extension Subroutines	If you cache a library using the ‘S’ option within a script, displays subroutines contained in the library.
Extension Functions	If you cache a library using the ‘S’ option within a script, displays functions contained in the library.

Appendix D

Script Constants

Constants are script variables that are built in to the script language to improve readability of the script or return environment or script information.

Flag values in the Script commands and control styles are usually defined using constants.

You can find the actual value used by a constant by running "example\script\wc.ws" and typing the command:

MessageBox constant_name

This value can be used directly in the script instead of the constant name except for Application, Environment and Script constants as these can change.

The different constant types are:

Constant Type	Description
Application	Constants that return information about the application, session.
Boolean	True or false constants
Character	Single characters
Control Style	Control styles used with dialog box commands
Environment	Information from Windows
Multiple Commands	General values used with multiple script commands functions
Script	Information about the currently running script
Specific Command	Values for a specific command

Constant Type	Description
Window Style	Windows dialog box styles and general styles for dialog box controls

Application Constants

Application constants may give different values when run by a different user.

These constants are:

Constant Name	Description
AppDir	The name of the folder this application is installed in
AppName	The name of the executable of this application
AppTitle	The name of this application
FullName	The full path name of the current session file.
Serial	The serial number for this user
SettingsDir	The default folder where the user settings will be stored. This is the same as the UserDir for the Local version and the location of the Users files on the Server for the server version.
SharedDir	A folder shared by all users. (This folder may not have been created or be accessible).
UserDir	The default local folder where user created files will be stored
Version	The version of the application

Run the example script "example\script\const.wis" to see the current values of selected constants.

Boolean Constants

Boolean constants are value that can be used instead of 1 or 0 to indicate a true or false value.

The following constant names evaluate to 1 (true):

On

True

Yes

The following constant names evaluate to 0 (false):

False

No

Off

Character Constants

Character constants return values that are a single character.

The same character can also be returned using the Char function with the ASCII code from the following table:

Constant Name	ASCII code
cr	13
etb	17
etx	3
lf	10
stx	2
tab	9

Control Styles

Control Style constants are used to define the styles for the dialog box controls created using the DialogBox commands.

The control style constants have a unique prefix as follows:

Prefix	Control
ACS_	Animate
BS_	Button
CBS_	Combo box
DS_	Dialog box
DTS_	Date Time
ES_	Edit control
GBS_	Graphic button
HDS_	Header
LBS_	List box
LVS_	List view
SBS_	Scroll bar
SS_	Text
TBS_	Track bar
TCS_	Tab control
TVS_	Tree view
UDS_	UpDown control

See the "Reference - Script Controls" section in this manual for details on the styles names and usage.

Environment Constants

Environment Constants return information about the Windows Desktop environment.

These constants are:

Constant Name	Description
ClientUserName	The name the user logged on to Windows
ClientIpAddress	The IP address where the client is running. In the local version this only returns a valid value if Windows Sockets is the current connection and the connection is open.
ComputerName	The name of this computer
ServerComputerName	The name of the server computer when running the thin client
DesktopType	The desktop type
ThinClient	True if running on a thin client version
Language	The language code of the desktop
OSVersion	The Windows OS version
WindowsDir	The path to the "Windows" folder.

Run the example script "example\script\const.wis" to see the current values of selected constants.

Multiple Command Constants

These constants are used to represent the same concept in different script commands.

Screen Colors

Screen colors are from 0 to 15 and represent the index into the screen palette setup under the Colour_<number> entries in the Setup Colors menu option.

The following name were chosen to represent the default palette used:

Constant Name	Value
COL_Black	0
COL_Blue	1
COL_Brown	6
COL_Cyan	3
COL_Gray	8
COL_Green	2
COL_Grey	8
COL_LightBlue	9
COL_LightCyan	11
COL_LightGray	7
COL_LightGreen	10
COL_LightGrey	7
COL_LightMagenta	13
COL_LightRed	12
COL_Magenta	5
COL_Red	4
COL_White	15

Constant Name	Value
COL_Yellow	14

The constant COL_None (value -1) is used to turn off color processing in the Color script command.

Data Formats

Data formats are used when reading/writing or retrieving captured data.

Constant Name	Value
DF_BS	1
DF_FT	3
DF_Hex	2
DF_NoZero	4
DF_UTF16	5
DF_UTF8	6

Font Styles

Font styles are used to specify the weight and style of a font.

Constant Name	Value
FONT_Bold	7
FONT_Heavy	9
FONT_Italic	16
FONT_Light	3
FONT_Normal	4
FONT_StrikeOut	64
FONT_Underline	32

Screen position

The screen position constants used by function to specify a location on the screen.

Constant Name	Value
Get_Page	2
Get_X	0
Get_Y	1

RGB Colors

The RGB color constants define a color in 24 bits where the red, green and blue intensities can range from 0 to 255. An alternative to using these constants is the RGB function directly.

The intensity of each of the components of the color can be found with the Red, Green and Blue functions.

Commands that use the RGB colors, select an appropriate color similar to the required colour when the Windows desktop is set to a lower color resolution.

Constant Name	Value
RGB_Black	0
RGB_Blue	8388608
RGB_Brown	32896
RGB_Cyan	8421376
RGB_Gray	8421504
RGB_Green	32768
RGB_Grey	8421504
RGB_LightBlue	16711680

Constant Name	Value
RGB_LightCyan	16776960
RGB_LightGray	12632256
RGB_LightGreen	65280
RGB_LightGrey	12632256
RGB_LightMagenta	16711935
RGB_LightRed	255
RGB_Magenta	8388736
RGB_Red	128
RGB_White	16777215
RGB_Yellow	65535

Script Constants

Script Constants return values that are specific to the currently running script.

Constant Name	Description
ScriptDir	The folder the script was run from.
ScriptFile	The file name of the current script.

Both of the above constants will be "" if the current script has not been run directly from a file.

Run the example script "example\script\const.wis" to see the current values of selected constants.

Specific Commands

Specific Commands constants return values that are only of use to the command they are designed to support.

Each has a unique prefix. See the specific command descriptions in the "Reference - Script Commands" section of this manual.

Prefix	Command
BRUSH_	Draw Brush
CAP_	Capture On
CF_	ChooseFont function options.
DE_	DialogBox Reply (Drag/Drop events)
ERASE_	Draw Erase
MB_	MessageBox
PEN_	Draw Pen

Windows Styles

The Windows Styles constants return values to be used by dialog boxes or controls.

The following styles apply to dialog boxes and controls equally. See the individual Control description and the DialogBox Style command for details.

WS_BORDER, WS_CHILD, WS_DISABLED, WS_GROUP, WS_TABSTOP,
WS_HSCROLL, WS_VISIBLE, WS_VSCROLL

The following styles apply to dialog boxes only. See the DialogBox Style command for details.

WS_CAPTION, WS_CHILDWINDOW, WS_CLIPCHILDREN,
WS_CLIPSIBLINGS, WS_DLGFRAME, WS_ICONIC, WS_MAXIMIZE,
WS_MAXIMIZEBOX, WS_MINIMIZE, WS_MINIMIZEBOX,
WS_OVERLAPPED, WS_OVERLAPPEDWINDOW, WS_POPUP,
WS_POPUPWINDOW, WS_SIZEBOX, WS_SYSMENU, WS_THICKFRAME,
WS_TILED, WS_TILEDWINDOW

The following extended windows styles are used with the DialogBox ExStyle command.

WS_EX_APPWINDOW, WS_EX_TOOLWINDOW

Appendix E

Backslash format strings

Backslash format strings are Unicode strings with character codes less than 32 and code 127 "escaped" out

The backslash format also allows specific character codes to be included in the string.

Certain commands (e.g Type, Display Text) convert these strings to their character equivalent as part of their functionality.

The Backslash format for a "special" character consists of the back slash character (\) followed by from 1 to 3 characters as in the following table:

Sequence	8-bit ASCII	Description
\r	13	Carriage Return
\n	10	Line feed
\t	9	tab
\f	12	Form feed
\e	27	Escape
\z	0	Character zero
\nnn	nnn	Decimal character number nnn. e.g. \013 = \r = Carriage return
\xnn	Hex nn	Hexidecimal number nn gives the character with the hex value nn. e.g. \x0D = \r = Carriage return
\unnnn	Unicode nnnn	Unicode character nnnnn gives the character with the hexadecimal value nnnn. e.g. \u000D = \r = Carriage return

Appendix F

The Script Compiler

Scripts can be compiled to a more efficient pseudo code format.

Scripts can be compiled individually in the wIntegrate Editor or a selection from a folder can be compiled using the wIntSys\Script\Compile.wis script.

Compiled scripts are saved with the .WIX file extension.

Scripts, called scripts and libraries work properly together regardless of whether any or all have been compiled.

Compiled scripts are stored in a format that consists of printable ASCII characters designed to facilitate storage and execution from either PC or host.

Compiled scripts are only available with version 4.2.1 or later of this application.

Index

#

operator, [34](#)

\$

\$, [35](#)

&

& operator, [37](#)

&= assignment, [38](#)

*

* comment, [39](#)

* operator, [39](#)

*= assignment, [40](#)

+

+ operator, [41](#)

+= assignment, [42](#)

-

- operator, [43](#)

-= assignment, [44](#)

/

/ operator, [45](#)

/= assignment, [46](#)

:

: operator, [47](#)

:= assignment, [48](#)

<

< operator, [49](#)

<= operator, [50](#)

<> operator, [34](#)

=

= assignment, [51](#)

= operator, [51](#)

>

> operator, [52](#)

>= operator, [53](#)

A

Abs, [54](#)

Activate, [55](#)

Activate Grid event, [1412](#)

AddCheckBox, [57](#)

AddComboBox, [58](#)

AddGap, [59](#)

AddIcon, [60](#)

AddLargeIcon, [62](#)

AddLargeSeparator, [64](#)

AddLargeStyleIcon, [65](#)

AddLine ComboBox method, [1159](#)

AddLine ListBox method, [1160](#)

AddSeparator, [67](#)

AddStyleIcon, [68](#)

AddTool, [70](#)

AddToolTip, [72](#)

AddWideIcon, [73](#)

AddWideStyleIcon, [74](#)

AdjustRect TabControl method, [1161](#)

Alias command, [35](#)

AlwaysOnTop, [76](#)

AnchorIndex ListBox property, [1045](#)

And, [77](#)

Animate control, [951](#)

AppFolder, [78](#)

Arrange ListView method, [1162](#)

Asc, [80](#)

AsciiToBS, [82](#)

AsciiToFT, [83](#)

AsciiToHex, [84](#)

assignments

&=, [38](#)

*=, [40](#)

+=, [42](#)

-=, [44](#)

/=, [46](#)

:=, [48](#)

=, [51](#)

|=, [949](#)

Attribute, [85](#)

AutoWrapOn, [88](#)

B

BackColor property, [1046](#)

BackStyle property, [1047](#)

Base UpDownButton property, [1048](#)

BaseBar, [89](#)

BeginLabelEdit ListView event, [1413](#)

BeginLabelEdit TreeView event, [1414](#)

BeginTrack Header event, [1415](#)

Binary And, [37](#)

Binary Or, [948](#)

Blue, [90](#)

BorderStyle Divider property, [1049](#)

BSToAscii, [92](#)

Buddy UpDownButton property, [1050](#)

ButtonClick Grid event, [1416](#)

C

Calculator control, [954](#)

Caption property, [1051](#)

Capture AddData, [93](#)
Capture Continue, [94](#)
Capture On, [99](#)
Capture Pause, [103](#)
CaptureText, [107](#)
CaretIndex ListBox property, [1052](#)
CellClick Grid event, [1417](#)
CellRightClick Grid event, [1418](#)
CellValue Grid property, [1053](#)
Chain, [109](#)
Change, [111](#)
Change EditText event, [1419](#)
Changed DateTime event, [1420](#)
ChangeUserSession, [113](#)
Char, [116](#)
CharFromPos EditText method, [1164](#)
Check CheckBox property, [1054](#)
Check RadioButton property, [1055](#)
CheckAll CheckListBox method, [1165](#)
CheckBox control, [956](#)
CheckIndex CheckListBox property, [1056](#)
CheckList CheckListBox property, [1057](#)
CheckListBox control, [959](#)
CheckServer, [118](#)
CheckText CheckListBox property, [1058](#)
ChooseColor, [119](#)
ChooseColour, [119](#)
ChooseFont, [121](#)
Clear method, [1166](#)
ClearRange Grid method, [1167](#)
ClearSel ListBox method, [1168](#)
ClearSel TrackBar method, [1169](#)
ClearTicks TrackBar method, [1170](#)
Click event, [1421](#)
Click ListView event, [1422](#)
Clipboard Read, [124](#)
Clipboard Write, [126](#)
Close Animate method, [1171](#)
CloseServer, [128](#)
CloseUp ComboBox event, [1423](#)
ColHeaders Grid property, [1059](#)
Color, [129](#)
ColorButton control, [964](#)
ColText Grid property, [1060](#)
Column Grid property, [1062](#)
ColumnClick ListView event, [1424](#)
ColumnInfo property, [1063](#)
Columns Grid property, [1065](#)
ColumnWidth ListBox property, [1064](#)
ColWidths Grid property, [1061](#)
ComboBox control, [967](#)
CommandBar Add, [131](#)
CommandBar OnRightClick, [133](#)
CommandBar Remove, [134](#)
CommandBar Replace, [135](#)
CommandBar SetScript, [136](#)
CommandBar Show, [137](#)
CommandBarInfo, [138](#)
CommandBarList, [140](#)
comments, [39](#)
concatenating strings, [47](#)
ConfigFile Read, [142](#)
ConfigFile ReadInteger, [144](#)
ConfigFile ReadSection, [146](#)
ConfigFile Remove, [148](#)
ConfigFile Write, [149](#)
Configure, [151](#)
Convert, [153](#)
Copy method, [1172](#)
Count, [154](#)
Count ComboBox method, [1173](#)
Count Header method, [1174](#)
Count ListBox method, [1175](#)
Count ListView method, [1176](#)
Count TabControl method, [1177](#)
Count TreeView method, [1178](#)
CreateIconBar, [155](#)
CreateToolBar, [157](#)

CurrentCell Grid event, [1425](#)
Cursor, [159](#)
CursorShape, [161](#)
CustomFormat DateTime property, [1066](#)
Cut method, [1179](#)

D

Data Grid property, [1068](#)
Date, [163](#)
DateMax DateTime property, [1069](#)
DateMin DateTime property, [1070](#)
DateTime control, [972](#)
DbClick event, [1426](#)
DbClick ListView event, [1427](#)
DCount, [164](#)
DDE Advise, [166](#)
DDE Execute, [169](#)
DDE HostLink, [171](#)
DDE Initiate, [173](#)
DDE Terminate, [177](#)
DDE Timeout, [179](#)
DDEChanged, [183](#)
DDERequest, [184](#)
Deactivate Grid event, [1428](#)
Debug, [185](#)
DecimalPlaces Calculator property, [1071](#)
Default PushButton property, [1072](#)
Define, [187](#)
Del, [189](#)
DeleteAllItems ListView method, [1180](#)
DeleteAllItems TabControl method, [1181](#)
DeleteAllItems TreeView method, [1182](#)
DeleteCols Grid method, [1183](#)
DeleteColumn ListView method, [1184](#)
DeleteItem Header method, [1185](#)
DeleteItem ListView method, [1186](#)

DeleteItem TabControl method, [1187](#)
DeleteItem TreeView method, [1188](#)
DeleteLine ComboBox method, [1189](#)
DeleteLine ListBox method, [1190](#)
DeleteRows Grid method, [1191](#)
DeltaPos ProgressBar method, [1192](#)
Destroy, [191](#)
Dialog, [193](#)
DialogBox AddToList, [195](#)
DialogBox Animate, [196](#)
DialogBox AxControl, [198](#)
DialogBox Background, [200](#)
DialogBox BaseBar, [201](#)
DialogBox Caption, [202](#)
DialogBox CheckBox, [203](#)
DialogBox CheckListBox, [205](#)
DialogBox ColorButton, [207](#)
DialogBox ComboBox, [209](#)
DialogBox Control, [211](#)
DialogBox ControlCommand, [214](#)
DialogBox Create, [216](#)
DialogBox CText, [218](#)
DialogBox DateTime, [220](#)
DialogBox Default, [222](#)
DialogBox DefPushButton, [223](#)
DialogBox Delete, [225](#)
DialogBox DeleteFromList, [227](#)
DialogBox EditText, [228](#)
DialogBox Enable, [230](#)
DialogBox EnableAll, [232](#)
DialogBox End, [234](#)
DialogBox EndCreate, [236](#)
DialogBox EventLibrary, [238](#)
DialogBox ExStyle, [239](#)
DialogBox Font, [241](#)
DialogBox Graphic, [242](#)
DialogBox GraphicButton, [244](#)
DialogBox Grid, [246](#)
DialogBox GroupBox, [247](#)

DialogBox Header, [249](#)
DialogBox Icon, [250](#)
DialogBox IconBar, [252](#)
DialogBox InitCommand, [253](#)
DialogBox InputOK, [254](#)
DialogBox LimitText, [256](#)
DialogBox ListBox, [258](#)
DialogBox ListView, [260](#)
DialogBox LoseFocus, [261](#)
DialogBox LText, [263](#)
DialogBox MaxSize, [265](#)
DialogBox MinIcon, [267](#)
DialogBox MinSize, [269](#)
DialogBox Move, [271](#)
DialogBox MoveControl, [272](#)
DialogBox New, [274](#)
DialogBox NextControl, [276](#)
DialogBox OnEvent, [280](#)
DialogBox Option, [281](#)
DialogBox Panel, [284](#)
DialogBox PopupMenu, [286](#)
DialogBox PopupWindow, [289](#)
DialogBox ProgressBar, [291](#)
DialogBox PushButton, [292](#)
DialogBox RadioButton, [294](#)
DialogBox Reply, [296](#)
DialogBox ResetControls, [297](#)
DialogBox RText, [299](#)
DialogBox ScrollBar, [301](#)
DialogBox ScrollStep, [303](#)
DialogBox Select, [305](#)
DialogBox SetColor, [306](#)
DialogBox SetColour, [306](#)
DialogBox SetFont, [308](#)
DialogBox SetKey, [310](#)
DialogBox SetMenu, [312](#)
DialogBox SetRange, [313](#)
DialogBox SetTabs, [315](#)
DialogBox SetText, [317](#)
DialogBox Show, [319](#)
DialogBox ShowControl, [321](#)
DialogBox Size, [322](#)
DialogBox Style, [324](#)
DialogBox TabControl, [326](#)
DialogBox ToolTip, [327](#)
DialogBox Trackbar, [328](#)
DialogBox TreeView, [330](#)
DialogBox UpDownButton, [332](#)
DialogBox Validate, [333](#)
DialogBox Variable, [335](#)
DialogBox Window, [338](#)
DialogBox WindowState, [339](#)
DialogBox*OnDialogEvent, [277](#)
DialogIndex, [341](#)
DialogList, [342](#)
DialogListFind, [343](#)
DialogNextEvent, [345](#)
DialogRect, [347](#)
DialogUnit, [349](#)
DialogWindowState, [352](#)
DirExist, [354](#)
Display, [363](#)
Display At, [355](#)
Display Attribute, [357](#)
Display CharFill, [361](#)
Display Command, [363](#)
Display Direct, [366](#)
Display Effect, [367](#)
Display EffectFill, [369](#)
Display Graphic property, [1073](#)
Display Text, [372](#)
Display UpdateOff, [373](#)
Display UpdateOn, [375](#)
DisplayBox, [359](#)
DisplayOn, [376](#)
Divider control, [975](#)
DividerDbClick Header event, [1429](#)
Drag Divider event, [1430](#)

DragData property, [1074](#)
DragEnd event, [1431](#)
DragEnded Divider event, [1433](#)
DragEnter event, [1434](#)
DragLeave event, [1436](#)
DragMode property, [1075](#)
DragOver event, [1437](#)
DragStart event, [1439](#)
DragStarted Divider event, [1441](#)
DragTarget property, [1076](#)
Draw Arc, [377](#)
Draw Brush, [379](#)
Draw Chord, [381](#)
Draw control, [977](#)
Draw CopyTo, [383](#)
Draw Ellipse, [385](#)
Draw Erase, [387](#)
Draw Font, [390](#)
Draw Image, [392](#)
Draw Line, [394](#)
Draw Move, [396](#)
Draw Pen, [397](#)
Draw Pie, [400](#)
Draw Polygon, [402](#)
Draw Rect, [404](#)
Draw Size, [406](#)
Draw Text, [407](#)
Draw To, [409](#)
Drop event, [1442](#)
DropDown ComboBox event, [1444](#)
Dropped ComboBox property, [1077](#)
Dropped DateTime property, [1078](#)

E

Edit Copy, [1497](#)
Edit Copy Special, [1498](#)
Edit Copy Special To, [1499](#)
Edit CopyTo, [1499](#)
Edit Paste, [1501](#)
Edit Play, [1502](#)
Edit Record, [1503](#)
Edit Search, [1505](#)
Edit Select All, [1506](#)
Edit Select Screen, [1507](#)
Edit Select Window, [1508](#)
EditChange ComboBox event, [1445](#)
EditInput, [411](#)
EditInput2, [415](#)
EditInputInfo, [419](#)
EditLabel ListView method, [1193](#)
EditLabel TreeView method, [1194](#)
EditSelect ComboBox method, [1195](#)
EditSelectInfo ComboBox method, [1196](#)
EditText control, [978](#)
EditUpdate ComboBox event, [1446](#)
Effect, [421](#)
Else, [423](#)
Empty ComboBox method, [1197](#)
Empty ListBox method, [1198](#)
Enabled property, [1079](#)
EndCreateIconBar, [424](#)
EndEditing Grid event, [1447](#)
EndIf, [425](#)
EndLabelEdit ListView event, [1448](#)
EndLabelEdit TreeView event, [1449](#)
EndOfFile, [426](#)
EndScript, [428](#)
EndSub, [429](#)
EndTrack Header event, [1450](#)
EndWith, [430](#)
EnsureVisible ListView method, [1199](#)
EnsureVisible TreeView method, [1200](#)
Enter, [431](#)
EnterAction Grid property, [1080](#)
Enum, [720](#)

EqualsPressed Calculator event, [1451](#)
ErrSpace ComboBox event, [1452](#)
ErrSpace EditText event, [1453](#)
Event Delete, [432](#)
Event OnExit, [433](#)
Event OnHostText, [435](#)
Event OnPortClose, [437](#)
Event OnPortOpen, [438](#)
Event OnRestore, [439](#)
Events, [1411](#)
ExecAccess, [440](#)
Execute, [441](#)
Expand TreeView method, [1201](#)
Extract, [443](#)

F

FastEntry DateTime property, [1081](#)
Field, [445](#)
Fields, [447](#)
File Animate property, [1082](#)
File Another, [1509](#), [1517](#)
File Close, [448](#)
File Copy, [450](#)
File Create, [452](#)
File CreateDir, [454](#)
File Delete, [456](#), [1510](#)
File DeleteDir, [458](#)
File Edit, [1511](#)
File Exit, [1512](#)
File ExitAll, [1513](#)
File Move, [460](#)
File Open, [462](#), [1514](#)
File Pointer, [464](#)
File Print, [1515](#)
File Printer Setup, [1516](#)
File Read, [465](#)

File ReadLine, [468](#)
File Write, [470](#)
FileBrowse, [473](#)
FileExist, [475](#)
FileInfo, [476](#)
FileList, [478](#)
Files, [480](#)
FileTempName, [481](#)
FillRange Grid method, [1202](#)
Find ComboBox method, [1203](#)
Find Grid method, [1204](#)
Find ListBox method, [1206](#)
FindExact ComboBox method, [1207](#)
FindExact ListBox method, [1208](#)
FindHWnd, [482](#)
FindItem ListView method, [1209](#)
FirstVisibleLine EditText method, [1211](#)
FloatingCells Grid property, [1083](#)
FocusItem TabControl property, [1084](#)
FolderBrowse, [483](#)
Font property, [1085](#)
FontInfo, [485](#)
ForeColor property, [1086](#)
FormatType DateTime property, [1087](#)
Ftp Close, [487](#)
Ftp Connect, [488](#)
Ftp CreateDir, [490](#)
Ftp Delete, [491](#)
Ftp DeleteDir, [492](#)
Ftp Disconnect, [493](#)
Ftp Get, [494](#)
Ftp Move, [496](#)
Ftp Open, [498](#)
Ftp Pointer, [500](#)
Ftp Put, [502](#)
Ftp Read, [504](#)
Ftp SetDir, [506](#)
Ftp Write, [507](#)
FtpDirExist, [509](#)

FtpExist, [511](#)
FtpGetDir, [513](#)
FtpInfo, [514](#)
FtpList, [517](#)
FTToAscii, [519](#)

G

Get, [520](#), [722](#)
GetCellAttribute Grid method, [1212](#)
GetCellAutoSize Grid method, [1213](#)
GetCellBackColor Grid method, [1214](#)
GetCellBorder Grid method, [1215](#)
GetCellControl Grid method, [1216](#)
GetCellEnabled Grid method, [1217](#)
GetCellFloat Grid method, [1218](#)
GetCellFont Grid method, [1219](#)
GetCellForeColor Grid method, [1220](#)
GetCellJustify Grid method, [1221](#)
GetCellList Grid method, [1222](#)
GetCellReadOnly Grid method, [1223](#)
GetCellValue Grid method, [1224](#)
GetCheck CheckListBox method, [1225](#)
GetColumn ListView method, [1227](#)
GetColumnWidth ListView method, [1229](#)
GetColWidth Grid method, [1226](#)
GetCountPerPage ListView method, [1230](#)
GetCurrentCell Grid method, [1231](#)
GetCurrentText Grid method, [1232](#)
GetFocus, [521](#)
GetFocusHWnd, [523](#)
GetFromHost Next, [524](#)
GetFromHost RawData, [526](#)
GetIconColumn, [528](#)
GetItem Header method, [1233](#)
GetItem ListView method, [1235](#)
GetItem TreeView method, [1237](#)

GetItemData ComboBox method, [1240](#)
GetItemData ListBox method, [1241](#)
GetItemData TreeView method, [1242](#)
GetItemImage TreeView method, [1243](#)
GetItemPosition ListView method, [1244](#)
GetItemState ListView method, [1245](#)
GetItemState TreeView method, [1247](#)
GetItemText Header method, [1249](#)
GetItemText ListView method, [1250](#)
GetItemText TabControl method, [1251](#)
GetItemText TreeView method, [1252](#)
GetItemWidth Header method, [1253](#)
GetMachineType, [529](#)
GetNextItem ListView method, [1254](#)
GetNextItem TreeView method, [1256](#)
GetOrigin ListView method, [1258](#)
GetRangeAttribute Grid method, [1259](#)
GetRangeAutoSize Grid method, [1260](#)
GetRangeBackColor Grid method, [1261](#)
GetRangeBorder Grid method, [1262](#)
GetRangeControl Grid method, [1263](#)
GetRangeEnabled Grid method, [1264](#)
GetRangeFloat Grid method, [1265](#)
GetRangeForeColor Grid method, [1266](#)
GetRangeJustify Grid method, [1267](#)
GetRangeList Grid method, [1268](#)
GetRangeReadOnly Grid method, [1269](#)
GetRangeValue Grid method, [1270](#)
GetRowHeight Grid method, [1271](#)
GetSel ListBox method, [1272](#)
GetSelectedCount ListView method, [1273](#)
GetTick TrackBar method, [1274](#)
GetTopIndex ListView method, [1275](#)
Global, [530](#)
Graphic control, [983](#)
GraphicButton control, [985](#)
Green, [532](#)
Grid control, [987](#)
GroupBox control, [996](#)

H

HCall, [533](#)
HDelete, [534](#)
Header control, [998](#)
Height property, [1088](#)
Help About, [1518](#)
Help Contents, [1519](#)
Help Index, [1519](#)
Help On Help, [1520](#)
Help Status, [1521](#)
Help Support Information, [1522](#)
Help Support Log, [1522](#)
Help TopicId, [535](#)
Help TopicName, [536](#)
HexToAscii, [537](#)
HideCols Grid method, [1276](#)
HideRows Grid method, [1277](#)
Horizontal Divider property, [1089](#)
HorizontalExtent property, [1090](#)
Host Break, [538](#)
Host Get, [539](#)
Host Send, [540](#)
HotSpot Add, [541](#)
HotSpot AutoDelete, [544](#)
HotSpot Delete, [545](#)
HotSpot DeleteAll, [546](#)
Hotspot Display, [547](#)
HotSpot Off, [549](#)
HotSpot On, [550](#)
HourGlass Off, [551](#)
HourGlass On, [552](#)
HourGlass Smash, [553](#)
HRead, [554](#)
HReadU, [556](#)
HReadV, [557](#)

HReadVU, [559](#)
HRelease, [560](#)
HScroll EditText event, [1454](#)
HWnd, [561](#)
HWrite, [562](#)
HWriteU, [564](#)
HWriteV, [565](#)
HWriteVU, [566](#)

I

Icon control, [1001](#)
IconBar, [567](#)
IConv, [568](#)
If, [571](#)
Image Graphic property, [1092](#)
Image GraphicButton property, [1093](#)
Image property, [1091](#)
ImageHeight Graphic property, [1094](#)
ImageList Add, [573](#)
ImageList AddIcon, [575](#)
ImageList Delete, [576](#)
ImageList GetIconSize, [577](#)
ImageList GetImageCount, [578](#)
ImageList GetImageInfo, [579](#)
ImageList Load, [580](#)
ImageList New, [582](#)
ImageList Remove, [583](#)
ImageWidth Graphic property, [1095](#)
Indent TreeView property, [1096](#)
Index, [584](#)
InfoBox, [586](#)
InitStorage ComboBox method, [1278](#)
InitStorage ListBox method, [1279](#)
Ins, [588](#)
InsertCols Grid method, [1281](#)
InsertColumn ListView method, [1282](#)

InsertItem Header method, [1284](#)
InsertItem ListView method, [1286](#)
InsertItem TreeView method, [1289](#)
InsertItemText Header method, [1292](#)
InsertItemText TabControl method, [1293](#)
InsertLine ComboBox method, [1294](#)
InsertLine ListBox method, [1295](#)
InsertLine TreeView method, [1296](#)
InsertRows Grid method, [1298](#)
InsPosFromPoint ListBox method, [1280](#)
InString, [590](#)
Invoke, [592](#)
IsApp, [594](#)
IsCommandBar, [595](#)
IsCommandBarShown, [596](#)
IsControl, [597](#)
IsDialog, [598](#)
IsEvent, [599](#)
IsFile, [600](#)
IsFullPath, [601](#)
IsMenu, [602](#)
IsMenuItem, [603](#)
IsNumber, [605](#)
IsObject, [606](#)
IsOnMenu, [607](#)
IsRemotePath, [608](#)
IsShown, [609](#)
IsTask, [610](#)
IsTrigger, [612](#)
IsVScreen, [613](#)
ItemChanged ListView event, [1455](#)
ItemChanging ListView event, [1458](#)
ItemClick Header event, [1461](#)
ItemDbClick Header event, [1462](#)
ItemExpanded TreeView event, [1463](#)
ItemExpanding TreeView event, [1465](#)
ItemFromPoint ListBox method, [1299](#)
ItemFromPoint TreeView method, [1300](#)
Items, [614](#)

K

Key ClearAll, [615](#)
Key ClearSet, [616](#)
Key LoadEmulation, [618](#)
Key Run, [619](#)
KeyboardLocked, [620](#)
KeyDown event, [1467](#)
KeyIn, [621](#)
KeyUp Calculator event, [1468](#)
KillFocus event, [1469](#)

L

Labels Header property, [1098](#)
Labels TabControl property, [1099](#)
LabelWidths Header property, [1097](#)
LastErrorText, [623](#)
Left, [625](#)
Left property, [1100](#)
Length, [626](#)
Library, [627](#)
LicenseInfo, [629](#)
LimitText property, [1101](#)
Line ComboBox method, [1301](#)
Line EditText method, [1302](#)
Line ListBox method, [1303](#)
LineCount EditText method, [1304](#)
LineFromChar EditText method, [1305](#)
LineIndex EditText method, [1306](#)
LineLength EditText method, [1307](#)
LineScroll EditText method, [1308](#)
LineSize TrackBar property, [1102](#)
List property, [1103](#)
ListBox control, [1002](#)

ListView control, [1007](#)
Local, [630](#)
LocalMode, [632](#)
LocalPrint, [634](#)
Locate, [635](#)
Loop, [638](#)

M

Mail Address, [640](#)
Mail Delete, [642](#)
Mail LookUp, [644](#)
Mail Read, [646](#)
Mail Send, [649](#)
MailAvailable, [651](#)
MailFindAll, [652](#)
MailFindNext, [655](#)
MakeCommandBar, [658](#)
MapChars, [659](#)
Match, [660](#)
MaximumPosition Divider property, [1104](#)
MaxText EditText event, [1470](#)
Menu AddItem, [662](#)
Menu Attach, [664](#)
Menu Command, [666](#)
Menu Create, [667](#)
Menu Delete, [669](#)
Menu DeleteItem, [671](#)
Menu Detach, [672](#)
Menu Enable, [673](#)
Menu EndCreate, [674](#)
Menu Help, [676](#)
Menu Main, [677](#)
Menu New, [679](#)
Menu OnClose, [680](#)
Menu OnOpen, [681](#)
Menu PopUp, [682](#)

Menu Separator, [684](#)
Menu SetCheck, [685](#)
MenuChecked, [686](#)
MenuEnabled, [688](#)
MenuGui Activate, [689](#)
MenuGUI Command, [690](#)
MenuGUI DefaultMap, [691](#)
MenuGUI GetScript, [693](#)
MenuGUI Initialise, [694](#)
MenuGUI Load, [695](#)
MenuGUI LoadDefault, [696](#)
MenuGUI LoadEmulationKeys, [697](#)
MenuGUI MapVariable, [699](#)
MenuGUI Save, [701](#)
MenuGUI SetupComms, [702](#)
MenuGUI Update, [703](#)
MenuList, [704](#)
MessageBox, [706](#)
Methods, [1158](#)
Mid, [709](#)
MinimumPosition Divider property, [1105](#)
Modified property, [1106](#)
ModifyCell Grid event, [1471](#)
Mouse, [711](#)
MultiExec, [713](#)
MultiScript, [715](#)

N

NextRow, [717](#)
NoEdit DateTime property, [1107](#)
Not, [718](#)
Notify, [719](#)

O

Object New, [724](#)
Object Release, [726](#)
Object Set, [727](#)
OConv, [730](#)
Open Animate method, [1309](#)
OpenServer, [735](#)
operators
 #, [34](#)
 &, [37](#)
 *, [39](#)
 +, [41](#)
 -, [43](#)
 /, [45](#)
 :, [47](#)
 <, [49](#)
 <=, [50](#)
 <>, [34](#)
 =, [51](#)
 >, [52](#)
 >=, [53](#)
 |, [948](#)
Optional DateTime property, [1108](#)
Or, [736](#)
OutOfMemory event, [1472](#)

P

PageSize TrackBar property, [1109](#)
PasswordChar EditText property, [1110](#)
Paste method, [1310](#)
PathAddExt, [737](#)
PathAppend, [739](#)
PathComponent, [741](#)
Play Abort, [743](#)
Play Animate method, [1311](#)
Play Start, [744](#)
Play Stop, [745](#)

PlayInfo, [746](#)
PopupCalculator, [747](#)
PortInfo, [749](#)
Position ProgressBar property, [1111](#)
Position ScrollBar property, [1112](#)
Position TrackBar property, [1113](#)
Position UpDownButton property, [1114](#)
Print, [751](#)
PrinterInfo, [752](#)
ProgressBar control, [1014](#)
Properties, [1044](#)
PushButton control, [1016](#)

R

Radio GroupBox property, [1115](#)
RadioButton control, [1019](#)
Random, [754](#)
Range ProgressBar property, [1116](#)
Range ScrollBar property, [1117](#)
Range UpDownButton property, [1118](#)
RangeMax TrackBar property, [1119](#)
RangeMin TrackBar property, [1120](#)
ReadOnly property, [1121](#)
Receive, [756](#)
Rectangle control, [1023](#)
Red, [757](#)
Redraw property, [1122](#)
Rep, [758](#)
Repeat, [760](#)
ReplaceSel EditText method, [1312](#)
ResizeColToFit Grid method, [1313](#)
ResizeRowToFit Grid method, [1314](#)
ResolveEffect, [761](#)
Return, [763](#)
Return event, [1473](#)
RGB, [764](#)

Right, [765](#)
RightClick event, [1474](#)
RightClick ListView event, [1475](#)
RightDbClick event, [1476](#)
RightDbClick ListView event, [1477](#)
Row Grid property, [1123](#)
RowCount TabControl method, [1315](#)
RowHeaders Grid property, [1124](#)
RowHeights Grid property, [1125](#)
Rows Grid property, [1127](#)
RowText Grid property, [1126](#)
Run, [766](#)
Run Bridge Copy File, [1523](#)
Run Clear Back Pages, [1525](#)
Run Export File, [1526](#)
Run Import File, [1528](#)
Run Kermit Server, [1534](#)
Run Kermit Server Command Get File, [1532](#)
Run Kermit Server Command Send File, [1533](#)
Run Program, [1535](#)
Run Recieve Binary File, [1536](#)
Run Resize Window, [1538](#)
Run Restart Port, [1539](#)
Run Script, [1540](#)
Run Scroll Across Width, [1546](#)
Run Scroll Back Width, [1547](#)
Run Scroll Down Line, [1548](#)
Run Scroll End Page, [1549](#)
Run Scroll Left Column, [1550](#)
Run Scroll Next Page, [1551](#)
Run Scroll Prev Page, [1552](#)
Run Scroll Right Column, [1553](#)
Run Scroll Top Page, [1554](#)
Run Scroll Up Line, [1555](#)
Run Send Binary File, [1541](#)
Run Server Kermit Command Finish, [1531](#)
Run Spool File, [1543](#)

S

Screen CopyPage, [769](#)
Screen Load, [770](#)
Screen Off, [772](#)
Screen On, [774](#)
Screen Remove, [776](#)
Screen Restore, [778](#)
Screen Save, [780](#)
Screen ScrollRegion, [782](#)
Screen Store, [784](#)
ScreenOn, [786](#)
ScreenText, [787](#)
ScreenWord, [789](#)
Script, [791](#)
Script Information, [1545](#)
ScriptError End, [792](#)
ScriptError EndAll, [793](#)
ScriptError Exit, [794](#)
ScriptError Ignore, [795](#)
ScriptError ReplaceLine, [796](#)
ScriptError RestartScript, [797](#)
ScriptError Skip, [798](#)
ScriptError Step, [799](#)
ScriptErrorInfo, [800](#)
ScriptGlobalInfo, [802](#)
ScriptInfo, [803](#)
ScriptVersion, [807](#)
Scroll EditText method, [1316](#)
Scroll TrackBar event, [1478](#)
ScrollBar control, [1025](#)
ScrollCaret EditText method, [1317](#)
ScrollTo Graphic method, [1318](#)
ScrollToCell Grid method, [1319](#)
Seek Animate method, [1320](#)
SelCancel ListBox event, [1479](#)

- SelChange ComboBox event, [1480](#)
- SelChange ListBox event, [1481](#)
- SelChanged TabControl event, [1482](#)
- SelChanged TreeView event, [1483](#)
- SelChanging TabControl event, [1484](#)
- SelChanging TreeView event, [1485](#)
- SelCount ListBox method, [1321](#)
- Select Area, [808](#)
- Select Clear, [810](#)
- Select EditText method, [1325](#)
- Select End, [811](#)
- Select Extend, [813](#)
- Select Keyboard, [815](#)
- Select Start, [816](#)
- Select TrackBar method, [1326](#)
- SelectAll Grid method, [1327](#)
- SelectCell Grid method, [1328](#)
- SelectedItem TabControl property, [1134](#)
- SelectInfo, [818](#)
- SelectInfo EditText method, [1329](#)
- SelectInfo Grid method, [1330](#)
- SelectItem TreeView method, [1331](#)
- SelectMode Grid property, [1133](#)
- SelectRange Grid method, [1332](#)
- SelEnd TrackBar property, [1128](#)
- SelEndCancel ComboBox event, [1486](#)
- SelEndOK ComboBox event, [1487](#)
- SelIndex ListBox property, [1130](#)
- SelIndex property, [1129](#)
- SelLine ComboBox method, [1322](#)
- SelLine ListBox method, [1323](#)
- SelRange ListBox method, [1324](#)
- SelStart TrackBar property, [1131](#)
- SelText ListBox property, [1132](#)
- SendKeys, [820](#)
- SendMessage, [823](#)
- ServerErrMsg, [826](#)
- Session Enable, [827](#)
- Session Execute, [828](#)
- Session LockSize, [829](#)
- Session Move, [830](#)
- Session Script, [832](#)
- Session Show, [833](#)
- Session Size, [834](#)
- SessionInfo, [835](#)
- SessionName, [837](#)
- Sessions, [838](#)
- Set, [840](#)
- SetCellAttribute Grid method, [1333](#)
- SetCellAutoSize Grid method, [1335](#)
- SetCellBackColor Grid method, [1336](#)
- SetCellBorder Grid method, [1337](#)
- SetCellControl Grid method, [1338](#)
- SetCellEnabled Grid method, [1340](#)
- SetCellFloat Grid method, [1341](#)
- SetCellFont Grid method, [1342](#)
- SetCellForeColor Grid method, [1344](#)
- SetCellJustify Grid method, [1345](#)
- SetCellList Grid method, [1346](#)
- SetCellReadOnly Grid method, [1347](#)
- SetCellValue Grid method, [1348](#)
- SetCheck CheckListBox method, [1349](#)
- SetColumn ListView method, [1351](#)
- SetColumnWidth ListView method, [1353](#)
- SetColWidth Grid method, [1350](#)
- SetCurrentCell Grid method, [1354](#)
- SetCurrentText Grid method, [1355](#)
- SetFocus, [842](#)
- SetFocus event, [1488](#)
- SetIconColumn, [844](#)
- SetImageList ListView method, [1356](#)
- SetImageList TreeView method, [1357](#)
- SetItem Header method, [1358](#)
- SetItem ListView method, [1360](#)
- SetItem TreeView method, [1362](#)
- SetItemCount ListView method, [1365](#)
- SetItemData ComboBox method, [1366](#)
- SetItemData ListBox method, [1367](#)

SetItemData TreeView method, [1368](#)
SetItemImage TreeView method, [1369](#)
SetItemPosition ListView method, [1370](#)
SetItemSize TabControl method, [1371](#)
SetItemState ListView method, [1372](#)
SetItemState TreeView method, [1374](#)
SetItemText Header method, [1376](#)
SetItemText ListView method, [1377](#)
SetItemText TabControl method, [1378](#)
SetItemText TreeView method, [1379](#)
SetItemWidth Header method, [1380](#)
SetPadding TabControl method, [1381](#)
SetRangeAttribute Grid method, [1382](#)
SetRangeAutoSize Grid method, [1383](#)
SetRangeBackColor Grid method, [1384](#)
SetRangeBorder Grid method, [1385](#)
SetRangeControl Grid method, [1386](#)
SetRangeEnabled Grid method, [1388](#)
SetRangeFloat Grid method, [1389](#)
SetRangeFont Grid method, [1390](#)
SetRangeForeColor Grid method, [1392](#)
SetRangeJustify Grid method, [1393](#)
SetRangeList Grid method, [1394](#)
SetRangeReadOnly Grid method, [1395](#)
SetRangeValue Grid method, [1396](#)
SetRowHeight Grid method, [1397](#)
SetSel ListBox method, [1398](#)
SetTick TrackBar method, [1399](#)
Setup Application, [1556](#)
Setup Character, [1558](#)
Setup Colors, [1559](#)
Setup Colours, [1559](#)
Setup Communications
 PicLan, [1563](#)
 Serial, [1564](#)
 Windows Sockets, [1566](#)
Setup EditKeys, [1568](#)
Setup FunctionKeys, [1569](#)
Setup Keyboard, [1569](#)

Setup Mouse, [1573](#)
Setup Preferences, [1574](#)
Setup Terminal, [1577](#)
Show, [845](#)
ShowURL, [847](#)
SortChildren TreeView method, [1400](#)
SortColOnDbClick Grid property, [1135](#)
SortCols Grid method, [1401](#)
SortRowOnDbClick Grid property, [1136](#)
SortRows Grid method, [1403](#)
Sound Loop, [849](#)
Sound Play, [851](#)
Sound PlaySystem, [853](#)
Sound Stop, [855](#)
Stamp, [856](#)
Start Animate event, [1489](#)
StartEditing Grid event, [1490](#)
Step ProgressBar property, [1137](#)
StepIt ProgressBar method, [1405](#)
Stop Animate event, [1491](#)
Stop Animate method, [1406](#)
StopList, [857](#)
Store, [858](#)
StoreAccess, [860](#)
String, [861](#)
Strip, [862](#)
Sub, [863](#)
SystemColor, [865](#)
SystemFolder, [868](#)
SystemFont, [869](#)
SystemMetrics, [871](#)

T

T, [873](#)
TabControl control, [1027](#)
TabStops EditText property, [1138](#)

TabStops ListBox property, [1139](#)
Text control, [1031](#)
Text property, [1140](#)
TextBackColor ListView property, [1141](#)
ThumbLength TrackBar property, [1142](#)
TickCount TrackBar method, [1407](#)
TickFrequency TrackBar property, [1143](#)
Time, [874](#)
ToolSpace, [875](#)
Top property, [1144](#)
TopIndex property, [1145](#)
Track Header event, [1492](#)
TrackBar control, [1033](#)
TrackHeight Grid property, [1146](#)
TrackWidth Grid property, [1147](#)
Translation Load, [877](#)
Translation Unload, [879](#)
Translation Use, [880](#)
TreeView control, [1037](#)
Trigger Add, [881](#)
Trigger Delete, [884](#)
Trigger DeleteAll, [885](#)
Trim, [886](#)
Type, [888](#)

U

Undo method, [1408](#)
Update, [891](#)
Update EditText event, [1493](#)
Update ListView method, [1409](#)
UpdateMode CheckListBox property, [1148](#)
UpdateMode ComboBox property, [1149](#)
UpdateMode ListBox property, [1150](#)
UpDownButton control, [1042](#)
UserGlobal, [893](#)

V

ValidateCell Grid event, [1495](#)
ValidMode DateTime property, [1151](#)
Value Calculator property, [1152](#)
Value DateTime property, [1153](#)
VarType, [895](#)
View ListView property, [1154](#)
VirtualScreen Background, [897](#)
VirtualScreen Delete, [899](#)
VirtualScreen DeleteAll, [900](#)
VirtualScreen MoveWindow, [901](#)
VirtualScreen New, [903](#)
VirtualScreen Pan, [905](#)
VirtualScreen PrintOff, [906](#)
VirtualScreen PrintOn, [907](#)
VirtualScreen Resize, [909](#)
VirtualScreen ResizeWindow, [910](#)
VirtualScreen Restore, [912](#)
VirtualScreen Scroll, [914](#)
VirtualScreen Store, [916](#)
VirtualScreen Window, [918](#)
Visible property, [1155](#)
VisibleCount TreeView method, [1410](#)
VSAttribute, [920](#)
VScroll EditText event, [1494](#)
VSCursor, [922](#)
VSInfo, [923](#)
VSText, [925](#)

W

Wait Delay, [926](#)
Wait During Task, [927](#)
Wait ForText, [928](#)
Wait RxEmpty, [930](#)

Wait TxEmpty, [932](#)
WebStyle property, [1156](#)
While, [940](#)
Width property, [1157](#)
WindowPos, [941](#)
With, [943](#)
WorkArea, [945](#)

Y

Yield, [947](#)

|

| operator, [948](#)
|= assignment, [949](#)