Table of Contents

Foreword 0

Part I Introduction 48

Part II Overview (Part 1: Application Controls) 49

Part III Overview (Part 2: Advanced Visible Controls) 55

Part IV Versions History 60

Part V Installation instructions 77

Part VI Registration Information 80

Part VII License Agreement 81

Part VIII Contact 83

Part IX Components (Part I) 83

1 acAccurateTimer ... 83

acAccurateTimer ... 83
Properties ... 83
Active ... 83
Interval ... 84
Methods ... 84
Reset ... 84
Events ... 85
OnTimer ... 85

2 acAnimationEffect ... 85

TacAnimationEffect ... 85
Properties ... 85
AnimateOnShow ... 85
AnimateOnHide ... 86
Delay ... 86
EffectType ... 86
FireworkBlasts ... 87
Frames ... 87
Steps ... 87
Methods ... 87
Open ... 87
Close ... 88
Events ... 88
OnShowEffect ... 88
OnHideEffect ... 89

3 acAppAutoRun ... 89

TacAppAutoRun ... 89

© 1998-2006, UtilMind Solutions®

Properties ... **90**
AutoRun ... 90
AutoStarted ... 91
CaptionButton ... 91
BtnOrder ... 91
Cursor ... 92
CursorDown ... 92
Enabled ... 93
Hint ... 93
SeparatorWidth ... 93
ShowHint ... 94
Visible ... 94
LogonNotice ... 94
Caption ... 95
Enabled ... 95
Text ... 95
MenuItem ... 96
RunFor ... 96
RunOnce ... 96
SystemMenu ... 97
ApplyToMenu ... 98
Caption ... 98
Position ... 98
Separators ... 99
Events ... **99**
OnAutoRun ... 99
OnButtonClick ... 100
4 acAppBar ... 100
TacAppBar ... **100**
Properties ... **102**
AllowedEdges ... 102
AlwaysOnTop ... 103
AutoHide ... 103
Docking ... 104
Enabled ... 104
Width ... 105
Height ... 105
MaxWidth ... 105
MaxHeight ... 105
MinWidth ... 106
MinHeight ... 106
DontSlideOnTaskBarWhenAutoHide ... 106
FloatRestrictions ... 107
Enabled ... 107
MaxWidth ... 107
MaxHeight ... 108
MinWidth ... 108
MinHeight ... 108
LastDockingPlace ... 109
Placement ... 109
RegistrySaver ... 110
Enabled ... 110
IniFileName ... 111
IniSection ... 111

© 1998-2006, UtilMind Solutions®

Options ... 111
RegKey ... 112
RegLocation ... 112
RestoreProperties ... 113
Storage ... 113
Sizing ... 114
Enabled ... 114
HorizontalIncrement ... 114
VerticalIncrement ... 115
Sliding ... 115
Enabled ... 115
SlideTime ... 116
TaskIcon ... 116
TitleBar ... 116
Methods ... 117
IsMouseOverAppBar ... 117
ShowHiddenAppBar ... 117
UpdateBar ... 117
Events ... 118
OnActivate ... 118
OnClick ... 118
OnDblClick ... 118
OnDeactivate ... 119
OnDocked ... 119
OnDocking ... 119
OnDockMove ... 120
OnExitSizeMove ... 120
OnFullScreenApp ... 120
OnHiding ... 120
OnOtherAppBarPosChanged ... 121
OnUnableToAutoHide ... 121
OnUnhiding ... 121
OnWindowArrange ... 122
Frequently Asked Questions ... 122

5 acAppCursors ... 123

TacAppCursors ... 123
Properties ... 124
BorderBottom ... 124
BorderBottomLeft ... 124
BorderBottomRight ... 124
BorderLeft ... 125
BorderNonResize ... 125
BorderRight ... 125
BorderTop ... 126
BorderTopLeft ... 126
BorderTopRight ... 126
BtnClose ... 126
BtnHelp ... 127
BtnMaximize ... 127
BtnMinimize ... 127
Caption ... 127
Client ... 128
ClientNonVCL ... 128
MainMenu ... 128

© 1998-2006, UtilMind Solutions®

MenuItem ... 129
ScrollHorz ... 129
ScrollVert ... 129
SystemMenu ... 129
Transparent ... 130
Events ... 130
OnMouseOverObject ... 130
6 acAppEvents ... 130
TacAppEvents ... 130
Properties ... 131
AnimateIcon ... 131
AnimateInterval ... 131
AppPriorityClass ... 132
FixedMinimizeRestoreAnimation ... 132
HintColor ... 133
HintFont ... 133
HintHidePause ... 133
HintPause ... 133
HintShortPause ... 134
Icon ... 134
ImageIndex ... 134
ImageList ... 135
IsWindowsShutdown ... 135
MouseDragImediate ... 135
MouseDragThreshold ... 136
ShowHint ... 136
ShowMainForm ... 136
ShowTaskIcon ... 137
UpdateFormatSettings ... 137
UpdateMetricSettings ... 137
Events ... 138
OnActionExecute ... 138
OnActionUpdate ... 138
OnActivate ... 139
OnActiveControlChange ... 139
OnActiveFormChange ... 140
OnAppKeyDown ... 140
OnAppKeyUp ... 141
OnAppMouseDown ... 142
OnAppMouseMove ... 142
OnAppMouseUp ... 143
OnDeactivate ... 143
OnException ... 144
OnHelp ... 144
OnHint ... 145
OnIdle ... 145
OnMessage ... 146
OnMinimize ... 146
OnRestore ... 147
OnSettingsChanged ... 147
OnShortCut ... 147
OnShowHint ... 148
OnWindowsShutdown ... 148

© 1998-2006, UtilMind Solutions®

7 acAppGlobalAlert 149

TacAppGlobalAlert 149
Properties 149
GlobalAlert 149
Methods 150
Alert 150
Events 150
OnAlert 150

8 acAppGlobalSwitch 151

TacAppGlobalSwitch 151
Properties 151
GlobalSwitch 151
RegistrySaver 152
Enabled 152
RegKey 153
RegLocation 153
RegValue 153
Value 153
Events 154
OnSwitch 154

9 acAppStartCounter 154

TacAppStartCounter 154
Properties 155
Counter 155
InstallTime 155
LifeTime 156
RegKey 156
RegLocation 156
Methods 157
ResetCounter 157
Events 157
OnFirstStart 157

10 acAutoUpgrader 158

TacAutoUpgrader 158
Properties 159
AutoCheck 159
AutoCheckDelay 160
CacheOptions 160
HTTPPassword 161
HTTPUsername 161
InfoFile 162
Files 162
UpgradeMethod 162
UpgradeMsg 163
InfoFileURL 164
InternetOptions 165
MiscFilesUpgrade 165
Proxy 166
AccessType 167
ProxyBypass 167
ProxyPassword 168
ProxyPort 168
ProxyServer 168

© 1998-2006, UtilMind Solutions®

ProxyUsername ... 169
RestartParams ... 169
ShowMessages ... 170
ThreadPriority ... 171
TransferBufferSize ... 171
VersionControl ... 172
VersionDate ... 172
VersionDateAutoSet ... 173
VersionNumber ... 173
Wizard ... 174
Enabled ... 174
HideFileLocation ... 174
Language ... 175
Pic118x218 ... 175
StayOnTop ... 176
Methods ... 176
Abort ... 176
CheckUpdate ... 177
RestartApplication ... 177
Events ... 178
OnAborted ... 178
OnAfterRestart ... 178
OnBeginUpgrade ... 179
OnConnLost ... 180
OnDoOwnCloseAppMethod ... 181
OnEndUpgrade ... 181
OnFileDone ... 182
OnFileStart ... 183
OnHostUnreachable ... 183
OnLaterUpgrade ... 184
OnLostFile ... 185
OnNoInfoFile ... 186
OnNoUpdateAvailable ... 186
OnPasswordRequest ... 187
OnProgress ... 188
OnProxyAuthenticationRequest ... 190
Upgrade Information File ... 191
Self-upgrading mechansim ... 192
Calculation of upgrades ... 192
Screenshots ... 194
Application update wizard ... 194
Multi-language support ... 194
Access to protected area ... 195
Info-file designer ... 195
11 acAwayTimer ... 196
TacAwayTimer ... 196
Properties ... 197
Active ... 197
AwayTime ... 197
MonitorInterval ... 198
SleepInterval ... 198
Methods ... 198
ResetAway ... 198
Events ... 199

© 1998-2006, UtilMind Solutions®

OnActiveTimer ... 199
OnScreenSaverStart ... 199
OnScreenSaverEnd ... 199
OnSleep ... 200
OnAwake ... 200

12 acCaptionButton ... 200

TacCaptionButton ... 200
Properties ... 202
AllowPress ... 202
BtnOrder ... 202
Cursor ... 203
CursorDown ... 203
Down ... 203
Enabled ... 204
Glyph ... 204
GlyphDown ... 204
GlyphTransparent ... 205
Hint ... 205
PopupMenu ... 206
SeparatorWidth ... 206
ShowHint ... 206
Sign ... 207
SignFont ... 207
SignLeft ... 207
SignTop ... 208
SystemMenu ... 208
ApplyToMenu ... 209
Caption ... 209
Position ... 210
Separators ... 210
Visible ... 210
Methods ... 211
RefreshButton ... 211
RefreshNCArea ... 211
ShowParentForm ... 211
Events ... 212
OnCaptionDblClick ... 212
OnClick ... 212
OnButtonClick ... 212
OnMenuClick ... 213
OnPressed ... 213
OnReleased ... 213

13 acClipboard ... 214

acClipboard ... 214
Properties ... 214
Active ... 214
AvailableFormats ... 215
ClipboardBitmap ... 215
ClipboardFiles ... 216
ClipboardText ... 216
Methods ... 217
Clear ... 217
HasFormat ... 217

© 1998-2006, UtilMind Solutions®

GetFormatIndex ... 218
GetFormatName ... 219
GetStream ... 219
GetString ... 219
PutStream ... 220
PutString ... 220
Events ... 221
OnChange ... 221
14 acConnectionList ... 221
acConnectionList ... 221
Properties ... 222
AutoResolve ... 222
CacheSize ... 223
ConnectionList ... 223
Count ... 224
IsOnline ... 224
Items ... 225
LocalHostname ... 226
LocalIP ... 226
MonitorActive ... 227
MonitorInterval ... 227
Priority ... 227
RegistrySaver ... 228
RetrieveTCP ... 228
RetrieveUDP ... 229
Methods ... 229
Refresh ... 229
FindConnectionByLocalPort ... 229
FindConnectionByRemotePort ... 230
FindConnectionByLocalIPStr ... 230
FindConnectionByRemoteIPStr ... 231
FindLocalIPStrByLocalPort ... 231
FindLocalIPStrByRemotePort ... 231
FindRemoteIPStrByLocalPort ... 232
FindRemoteIPStrByRemotePort ... 232
IsPortListening ... 233
Events ... 233
OnHostnameRetrieved ... 233
OnOnlineStatusChanged ... 234
TacConnection ... 234
Data ... 235
LocalIP ... 235
LocalIPStr ... 236
LocalIPLong ... 236
LocalHostname ... 236
LocalPort ... 237
RemoteIP ... 237
RemoteIPStr ... 237
RemoteIPLong ... 237
RemoteHostname ... 238
RemotePort ... 238
State ... 238
StateStr ... 239
Proto ... 239

© 1998-2006, UtilMind Solutions®

ProtoName ... 239
IsIncoming ... 240
15 acCronJob ... 240
acCronJob ... 240
Properties ... 241
Active ... 241
AlertsCount ... 241
CronMask ... 242
LastTime ... 243
Events ... 243
OnAlert ... 243
16 acCursorLocker ... 243
TacCursorLocker ... 243
Properties ... 244
Active ... 244
Control ... 244
17 acEmbeddedForm ... 244
TacEmbeddedForm ... 244
Properties ... 245
FormName ... 245
HolderControl ... 246
18 acExtAssociation ... 246
TacExtAssociation ... 246
Properties ... 247
AccessMode ... 247
DefCommandName ... 247
ExecutableFile ... 248
ExtDescription ... 248
EXTENSION ... 248
FileDescription ... 249
IconIndex ... 249
IconLocation ... 249
LargeIcon ... 250
MIMEType ... 250
ParamString ... 250
SmallIcon ... 251
Methods ... 251
InstallExtension ... 251
UninstallExtension ... 252
ExtAssociation example ... 252
19 acFileStorage ... 253
TacFileStorage ... 253
Properties ... 253
AutoExtract ... 253
Enabled ... 254
ExtractTo ... 254
CreatePath ... 255
SubDirectory ... 255
ToplevelDir ... 256
Overwrite ... 257
RemoveOnTerminate ... 257
Count ... 258

© 1998-2006, UtilMind Solutions®

DataSize ... 258
Files ... 258
Methods ... 259
Extract ... 259
Events ... 260
OnExtract ... 260
OnExtracted ... 260
FileStorage designer ... 261
Playing .WAVs (example) ... 261
20 acFormBackground ... 262
TacFormBackground ... 262
Properties ... 263
Active ... 263
GradientColor1 ... 263
GradientColor2 ... 263
GradientDirection ... 264
Kind ... 264
Picture ... 265
PictureStyle ... 265
21 acFormHelp ... 266
TacFormHelp ... 266
Properties ... 267
Active ... 267
AdjustPopupWidth ... 268
CaptionButton ... 268
BtnOrder ... 269
Cursor ... 269
CursorDown ... 270
Enabled ... 270
Hint ... 270
SeparatorWidth ... 271
ShowHint ... 271
Visible ... 271
Color ... 272
Cursor ... 272
CursorHelp ... 272
DelayInterval ... 272
Font ... 273
Margins ... 273
Horizontal ... 273
Vertical ... 273
ParentFont ... 274
PopupWidth ... 274
ShadowColor ... 274
ShowShadow ... 275
SystemMenu ... 275
ApplyToMenu ... 276
Caption ... 276
Position ... 276
Separators ... 277
TextStyle ... 277
PlainText ... 278
TagClose ... 278

© 1998-2006, UtilMind Solutions®

TagOpen ... 279
UseF1 ... 279
WhatsThis ... 279
Caption ... 280
Enabled ... 280
MenuItem ... 281
ToolbarButton ... 281
Methods ... 282
InvokeHelpPointer ... 282
ShowHelp ... 282
ShowHelpFromControl ... 283
ShowHelpFromPoint ... 283
Events ... 284
OnButtonClick ... 284
OnHide ... 284
OnShow ... 284
OnWhatsThis ... 285
Rich-formatting tags ... 286
FormHelp designer / Hint property editor ... 287
Application.Hint problem ... 288
22 acFormHints ... 288
TacFormHints ... 288
Properties ... 289
Active ... 289
BiDiMode ... 289
ColorBackground ... 290
ColorBorder ... 290
ColorShadow ... 290
Direction ... 290
Font ... 291
MaxWidth ... 291
Rounded ... 291
RoundRadius ... 292
ShowArrow ... 292
ShowShadow ... 292
TextAlignment ... 293
TransparencyRate ... 293
Transparent ... 293
Events ... 294
OnShowHint ... 294
23 acFormHook ... 294
TacFormHook ... 294
Properties ... 295
AcceptFiles ... 295
ActivateFormOnAcceptFiles ... 295
Bottommost ... 295
CloseButton ... 296
DragByCaption ... 296
DragByClient ... 296
NoCaption ... 297
NoPaintBackground ... 297
Topmost ... 297
Events ... 298

© 1998-2006, UtilMind Solutions®

OnCaptionChanged ... 298
OnDropFiles ... 298
OnFormCloseButtonClick ... 298
OnFormRecreate ... 299
OnFormShown ... 299
OnMessageAfter ... 300
OnMessageBefore ... 300

24 acFormMagnet ... 301

TacFormMagnet ... 301
Properties ... 301
Active ... 301
FormDragable ... 302
FormTopmost ... 302
Glue ... 302
Options ... 302
MagnetType ... 303
Power ... 303
ScreenMagnet ... 303
Area ... 304
Edges ... 304
ScreenPower ... 305
Methods ... 305
DoMagnet ... 305
Events ... 305
OnDragMove ... 305
OnGlue ... 305
OnMagnet ... 306

25 acFormPlacementSaver ... 306

TacFormPlacementSaver ... 306
Properties ... 307
Enabled ... 307
FlickerFreeMaximize ... 307
MaximizeIfNoInfoFound ... 307
Options ... 307
RegKey ... 308
RegLocation ... 308
RestoreFormSize ... 309
Methods ... 309
Load ... 309
Save ... 309
Events ... 310
OnLoadSettings ... 310
OnRegKeyNotFound ... 310
OnSaveSettings ... 311

26 acFormResizeGrip ... 312

TacFormResizeGrip ... 312
Properties ... 312
Visible ... 312

27 acFormRoller ... 312

TacFormRoller ... 312
Properties ... 313
Animate ... 313
AnimateSpeed ... 313

© 1998-2006, UtilMind Solutions®

CaptionButton ... 314
BtnOrder ... 314
Cursor ... 315
CursorDown ... 315
Enabled ... 316
Hint ... 316
SeparatorWidth ... 316
ShowHint ... 317
Visible ... 317
HintRollDown ... 317
HintRollUp ... 318
RemainHeight ... 318
RollUp ... 318
SystemMenu ... 319
ApplyToMenu ... 319
Caption ... 320
Position ... 320
Separators ... 321
Events ... 321
OnRollerChanged ... 321
28 acFormSizeRestrictions ... 321
TacFormSizeRestrictions ... 321
Properties ... 322
Enabled ... 322
MaxClientHeight ... 322
MaxClientWidth ... 322
MinClientHeight ... 323
MinClientWidth ... 323
MaxHeight ... 323
MaxWidth ... 324
MinHeight ... 324
MinWidth ... 324
29 acFormSystemMenu ... 325
TacFormSystemMenu ... 325
Properties ... 325
Active ... 325
InsertPosition ... 325
Items ... 326
SysItems ... 326
SystemMenuHandle ... 327
30 acFormTopmost ... 327
TacFormTopmost ... 327
Properties ... 328
CaptionButton ... 328
BtnOrder ... 329
Cursor ... 329
CursorDown ... 330
Enabled ... 330
Hint ... 330
SeparatorWidth ... 331
ShowHint ... 331
Visible ... 331
MenuItem ... 332

© 1998-2006, UtilMind Solutions®

SystemMenu ... 332
ApplyToMenu ... 333
Caption ... 333
Position ... 334
Separators ... 334
Topmost ... 334
Events ... 335
OnTopmostChanged ... 335
31 acHTTP ... 335
TacHTTP ... 335
Properties ... 337
AcceptTypes ... 337
AddHeaders ... 338
Agent ... 339
Busy ... 339
CacheOptions ... 339
FileName ... 340
HostName ... 341
InternetOptions ... 341
OutputFileAttributes ... 342
OutputFileName ... 343
Password ... 344
POSTData ... 344
Proxy ... 345
AccessType ... 346
ProxyBypass ... 346
ProxyPassword ... 346
ProxyPort ... 347
ProxyServer ... 347
ProxyUsername ... 347
Range ... 348
EndRange ... 348
StartRange ... 349
Referer ... 349
RequestMethod ... 349
ShowGoOnlineMessage ... 350
Suspended ... 351
Timeouts ... 351
ConnectTimeout ... 352
ReceiveTimeout ... 352
SendTimeout ... 352
Thread ... 353
ThreadPriority ... 353
TransferBufferSize ... 354
URL ... 354
Username ... 354
WaitThread ... 355
WaitTimeout ... 355
WorkOffline ... 356
Methods ... 356
Abort ... 356
IsGlobalOffline ... 357
Pause ... 357
Read ... 358

© 1998-2006, UtilMind Solutions®

ReadRange ... 358
Resume ... 359
Upload ... 360
UploadByFieldNames ... 360
Events ... 361
OnAborted ... 361
OnAnyError ... 362
OnBeforeSendRequest ... 362
OnConnLost ... 362
OnDone ... 363
OnDoneInterrupted ... 365
OnHeaderInfo ... 365
OnHostUnreachable ... 367
OnHTTPError ... 367
OnOutputFileError ... 369
OnPasswordRequest ... 370
OnProgress ... 371
OnProxyAuthenticationRequest ... 373
OnRedirected ... 373
OnUploadCGITimeoutFailed ... 374
OnUploadFieldRequest ... 374
OnUploadProgress ... 377
OnWaitTimeoutExpired ... 379
Appendix: HTTP Status Codes ... 379
HTTPReadString ... 382
32 acHTTPDownloadQueue ... 383
TacHTTPDownloadQueue ... 383
33 acIESniffer ... 383
TacIESniffer ... 383
Properties ... 385
Active ... 385
FormAutoFill ... 386
IEList ... 386
MonitorInterval ... 387
SearchBar ... 388
SearchRedirect ... 388
SniffWithHTTPPrefixOnly ... 389
URLs ... 389
Methods ... 390
AddBrowser ... 390
ClearMarks ... 390
CloseBrowsers ... 391
MarkText ... 391
Refresh ... 392
ReplaceText ... 393
Events ... 393
OnURLChange ... 393
OnWBBeforeNavigate2 ... 394
OnWBCommandStateChange ... 395
OnWBDocumentComplete ... 396
OnWBDownloadBegin ... 397
OnWBDownloadComplete ... 397
OnWBFileDownload ... 399

© 1998-2006, UtilMind Solutions®

OnWBFullScreen ... 399
OnWBMenuBar ... 400
OnWBNavigateComplete2 ... 400
OnWBNewWindow2 ... 401
OnWBProgressChange ... 402
OnWBPropertyChange ... 402
OnWBQuit ... 402
OnWBStatusBar ... 403
OnWBStatusTextChange ... 403
OnWBTheaterMode ... 404
OnWBTitleChange ... 404
OnWBToolBar ... 405
OnWBVisible ... 405
OnWindowLoad ... 406
OnWindowUnload ... 407

34 acIESnifferAutoFillUserInfo ... 407

TacIESnifferAutoFillUserInfo ... 407
Properties ... 408
AutoFill ... 408
AutoHighlight ... 408
CustomFields ... 409
Fields ... 409
FillTokens ... 410
HighlightColor ... 411
HighlightTextColor ... 411
RegistrySaver ... 411
Methods ... 412
Fill ... 412
Save ... 413

35 acInfoBox ... 413

TacInfoBox ... 413
Properties ... 414
ButtonCaptions ... 414
CheckBoxCaption ... 414
Font ... 415
Icon ... 415
InfoBoxButtons ... 415
Picture ... 415
PictureResID ... 415
Position ... 416
RegKey ... 417
RegLocation ... 417
RegValue ... 417
ShowCheckBox ... 417
Text ... 418
TextStyle ... 418
PlainText ... 418
TagClose ... 419
TagOpen ... 419
Title ... 420
Methods ... 420
Execute ... 420
Events ... 420

© 1998-2006, UtilMind Solutions®

OnCheckBoxChecked ... 420
36 acOnlyOne ... 421
TacOnlyOne ... 421
Properties ... 422
CloseFirst ... 422
CloseSecond ... 422
SwitchToFirst ... 422
Methods ... 423
CloseFirstInstance ... 423
Events ... 424
OnAlreadyExist ... 424
37 acProcessList ... 424
acProcessList ... 424
Properties ... 425
Count ... 425
Items ... 425
ProcessList ... 427
Methods ... 427
Refresh ... 427
FindProcessByPID ... 428
FindProcessByWnd ... 428
FindExeNameByPID ... 428
FindExeNameByWnd ... 429
FindDescriptionByPID ... 429
FindDescriptionByWnd ... 429
FindWindowTitleByPID ... 430
FindWindowTitleByWnd ... 430
TacProcess object ... 430
Properties ... 431
AppType ... 431
AppTypeName ... 432
Data ... 432
Description ... 433
ExeName ... 433
ProcID ... 434
SystemIcon16Handle ... 434
SystemIcon32Handle ... 434
SystemIconIndex ... 435
Windows ... 436
WindowTitle ... 436
Methods ... 437
Kill ... 437
38 acRSSFeed ... 437
TacRSSFeed ... 437
39 acRSSFeedContainer ... 438
TacRSSFeedContainer ... 438
40 acSendMail ... 438
TacSendMail ... 438
Properties ... 439
AddHeaders ... 439
Agent ... 439
Attach ... 440

© 1998-2006, UtilMind Solutions®

AttachType ... 440
Authentication ... 441
Enabled ... 441
Password ... 442
Username ... 442
Busy ... 442
FromAddress ... 443
FromName ... 443
FromOrganization ... 443
MsgBody ... 443
MsgContentCharset ... 444
MsgContentType ... 444
MsgDateTime ... 445
MsgPriority ... 445
MsgSubject ... 445
ReplyTo ... 446
SMTPHost ... 446
SMTPPort ... 446
Suspended ... 447
TemplateConverter ... 447
Thread ... 447
Threaded ... 448
ThreadPriority ... 448
ToAddr ... 449
ToBCC ... 450
ToCC ... 451
WaitThread ... 451
WaitTimeout ... 452
Methods ... 452
SaveToFile ... 452
Send ... 453
Abort ... 453
Events ... 453
OnAborted ... 453
OnAnyError ... 454
OnCantAttach ... 454
OnConnLost ... 454
OnHostUnreachable ... 455
OnProgress ... 455
OnResponse ... 456
OnSMTPError ... 457
OnSuccess ... 457
OnWaitTimeoutExpired ... 457
QuickSendMail function ... 458
41 acShortcut ... 459
TacShortcut ... 459
Properties ... 460
FILENAME ... 460
HotKey ... 460
IconIndex ... 461
IconLocation ... 461
LargeIcon ... 461
LinkTarget ... 461
Parameters ... 462

© 1998-2006, UtilMind Solutions®

RunAs ... 462
ShortDescription ... 462
SmallIcon ... 463
WorkingDirectory ... 463
Methods ... 463
LoadFromFile ... 463
SaveToFile ... 464
42 acThread ... 464
TacThread ... 464
Properties ... 465
Handle ... 465
HandleExceptions ... 465
Priority ... 465
ReturnValue ... 466
Running ... 466
Suspended ... 467
Terminated ... 467
ThreadID ... 467
WaitThread ... 468
WaitTimeout ... 468
Methods ... 469
Execute ... 469
Resume ... 469
Suspend ... 469
Synchronize ... 470
SynchronizeEx ... 470
Terminate ... 471
WaitFor ... 472
Events ... 472
OnException ... 472
OnExecute ... 472
OnTerminate ... 474
OnWaitTimeoutExpired ... 474
43 acThreadSafeLog ... 474
TacThreadSafeLog ... 474
Properties ... 475
DateTimeFormat ... 475
FileName ... 477
FlushBuffers ... 477
LogToFile ... 478
LogToMemo ... 478
MaxSize ... 479
Memo ... 479
Options ... 479
SendMail ... 480
Limit ... 481
MsgContentCharset ... 481
MsgFromAddress ... 481
MsgFromName ... 482
MsgPriority ... 482
MsgSubject ... 482
MsgToAddress ... 483
MsgToName ... 483

© 1998-2006, UtilMind Solutions®

SMTPHost ... 483
SMTPPort ... 483
Threaded ... 484
Methods ... 484
ClearLog ... 484
Flush ... 485
Log ... 485
GetLastRecords ... 486
EmailLastRecords ... 486
Events ... 486
OnCustomPrefix ... 486
OnFileWriteError ... 487
OnMaxSizeReached ... 487
44 acTrayIcon ... 488
TacTrayIcon ... 488
Properties ... 489
Animate ... 489
AnimateInterval ... 490
CaptionButton ... 490
BtnOrder ... 490
Cursor ... 491
CursorDown ... 491
Enabled ... 492
Hint ... 492
SeparatorWidth ... 492
ShowHint ... 493
Visible ... 493
Hint ... 493
Icon ... 494
Icon16 ... 494
ImageIndex ... 495
ImageList ... 495
IsMinimized ... 496
Options ... 496
PopupMenu ... 497
PopupMenuRestoreItem ... 497
PopupMenuTrayMinimizeItem ... 498
PopupMenuWhatsThisItem ... 498
ShowHint ... 499
StartMinimized ... 499
SystemMenu ... 499
ApplyToMenu ... 500
Caption ... 500
Position ... 501
Separators ... 501
Visible ... 502
WhatsThisHint ... 502
Win2kInfoBalloon ... 503
Enabled ... 503
Icon ... 504
NoSound ... 504
Text ... 504
Timeout ... 505
Title ... 505

© 1998-2006, UtilMind Solutions®

Methods ... **505**
MinimizeToTray ... 505
RestoreFromTray ... 506
ShowInfoBalloon ... 506
HideInfoBalloon ... 507
Events ... **507**
OnAnimate ... 507
OnCaptionDblClick ... 507
OnClick ... 508
OnBalloonHide ... 508
OnBalloonShow ... 508
OnBalloonTimeOut ... 508
OnBalloonUserClick ... 509
OnDblClick ... 509
OnHide ... 509
OnMinimize ... 510
OnMouseDown ... 510
OnMouseMove ... 510
OnMouseUp ... 511
OnPopupMenu ... 511
OnRestore ... 512
OnShow ... 512

45 acWavPlayer ... 512

TacWavPlayer ... **512**
Properties ... **513**
Asynchronous ... 513
Looped ... 513
SoundType ... 513
WaveSound ... 514
Methods ... **514**
Play ... 514
Stop ... 514
Events ... **515**
OnAfterPlay ... 515
OnBeforePlay ... 515

46 acWin2kEffects ... 515

TacWin2kEffects ... **515**
Properties ... **516**
AnimateOnHide ... 516
Enabled ... 516
Flags ... 517
Time ... 517
AnimateOnShow ... 518
Enabled ... 518
Flags ... 518
Time ... 519
FlashCount ... 520
FlashTaskIcon ... 520
Translucent ... 520
TranslucentRate ... 521
Methods ... **521**
Flash ... 521
FlashBackground ... 522

© 1998-2006, UtilMind Solutions®

Events ... **522**
OnShowEffect ... 522
OnHideEffect ... 522

Part X Components (Part II) 523

1 acApartPopupMenu ... 523

TacApartPopupMenu ... **523**
Methods ... **523**
PopupFromCursorPos ... 523
Events ... **523**
OnCloseUp ... 523

2 acBitBtn ... 523

TacBitBtn ... **523**
Properties ... **524**
Align ... 524
Caption ... 525
Color ... 525
ParentColor ... 526
WordWrap ... 526
Events ... **527**
OnMouseEnter ... 527
OnMouseLeave ... 527

3 acButton ... 527

TacButton ... **527**
Properties ... **528**
Align ... 528
Caption ... 529
WordWrap ... 529
Events ... **529**
OnMouseEnter ... 529
OnMouseLeave ... 530

4 acButtonsPanel ... 530

TacButtonsPanel ... **530**
Properties ... **531**
Align ... 531
Alignment ... 531
ApplyEnabled ... 532
BtnApply ... 532
BtnCancel ... 533
BtnCustom ... 534
BtnHelp ... 535
BtnOK ... 536
ButtonHeight ... 537
ButtonSpacing ... 537
ButtonWidth ... 537
CustomButtonPosition ... 538
Images ... 538
MarginHorz ... 539
MarginVert ... 539
ShowAccelChar ... 539
ShowDivider ... 539
ShowSizeGrip ... 540

© 1998-2006, UtilMind Solutions®

Events ... **540**
OnAnyBtnClick ... 540
OnApplyEnabledChanged ... 541
OnApplySettings ... 541
OnBtnApplyClick ... 541
OnBtnCancelClick ... 542
OnBtnCustomClick ... 542
OnBtnHelpClick ... 542
OnBtnOKClick ... 542
5 acCheckBox ... 543
TacCheckBox ... **543**
Properties ... **544**
Align ... 544
AlignmentHorz ... 545
AlignmentVert ... 545
BoxAlignment ... 546
Ctl3D ... 546
ParentCtl3D ... 547
WordWrap ... 547
Events ... **547**
OnMouseEnter ... 547
OnMouseLeave ... 548
6 acCheckGroupBox ... 548
TacCheckGroupBox ... **548**
Properties ... **549**
BoxAlignment ... 549
Caption ... 549
Checked ... 550
Ctl3D ... 550
State ... 550
Events ... **551**
OnCheckBoxClick ... 551
7 acComboDatePicker ... 551
TacComboDatePicker ... **551**
Properties ... **552**
Date ... 552
DateFormat ... 552
Day ... 553
Labels ... 553
Position ... 554
Spacing ... 554
StrDay ... 554
StrMonth ... 555
StrYear ... 555
Visible ... 555
MaxYear ... 555
MinYear ... 556
Month ... 556
SpacingBetweenCombos ... 556
WidthDayMonthBoxes ... 557
WidthYearBox ... 557
Year ... 557
Methods ... **557**

© 1998-2006, UtilMind Solutions®

Clear ... 557
Events ... **558**
OnChange ... 558
8 acControlKeyState ... 558
TacControlKeyState ... **558**
Properties ... **559**
CapsLock ... 559
Control ... 559
Insert ... 559
NumLock ... 559
ScrollLock ... 559
Shift ... 560
Events ... **560**
OnCapsLock ... 560
OnControl ... 560
OnInsert ... 560
OnNumLock ... 560
OnScrollLock ... 561
OnShift ... 561
9 acCurrencyEdit ... 561
TacCurrencyEdit ... **561**
Properties ... **561**
Value ... 561
Events ... **561**
OnChange ... 561
10 acDateTimeCalc ... 561
TacDateTimeCalc ... **561**
Properties ... **562**
BeginTime ... 562
EndTime ... 562
Years ... 562
Months ... 563
Days ... 563
Hours ... 563
Minutes ... 564
Seconds ... 564
MSeconds ... 564
11 acDBListView ... 564
TacDBListView ... **564**
Properties ... **565**
DataSource ... 565
12 acDesktopWallpaper ... 565
TacDesktopWallpaper ... **565**
Properties ... **566**
AutoFormat ... 566
Stretch ... 566
Tile ... 566
Wallpaper ... 567
Methods ... **567**
SetDesktopWallpaper ... 567
RemoveWallpaper ... 567
13 acDragPanel ... 568

© 1998-2006, UtilMind Solutions®

TacDragPanel **568**
Properties **568**
DragCursor 568
DragObject 568
14 acEdit **569**
TacEdit **569**
Properties **569**
Align 569
Alignment 570
AttachedLabel 571
Caption 572
Color 572
Cursor 573
Enabled 573
FocusOnClick 574
Font 574
Hint 574
ParentColor 575
ParentFont 575
ParentShowHint 576
PopupMenu 576
Position 576
ShowAccelChar 577
ShowHint 577
Spacing 578
SpacingKind 578
Transparent 579
Visible 579
WordWrap 580
AutoSelect 580
Button 580
Cursor 581
Flat 581
Glyph 581
NumGlyphs 582
Hint 583
Kind 583
Width 583
Visible 583
Color 584
ColorDisabled 584
CursorBorder 584
ValidateChars 585
ValidChars 585
Methods **585**
SetSelection 585
Events **586**
OnButtonClick 586
OnButtonLeftMouseDown 586
OnLabelClick 586
OnLabelDblClick 587
OnMouseEnter 587
OnMouseLeave 587

© 1998-2006, UtilMind Solutions®

15 acHeader 588
TacHeader 588
Properties 588
AutoSize 588
Background 588
Active 589
GradientColor1 590
GradientColor2 590
GradientDirection 590
Kind 591
Picture 591
PictureStyle 592
CaptionMenu 592
Image 593
ImageAlignment 593
ImageIndex 594
ImageList 594
ImageMenu 594
ImageTransparent 594
RightButton 595
AutoRoll 595
Cursor 595
Enabled 596
Glyph 596
Hint 596
Kind 596
ShowEdges 597
ShowHint 597
UseCaptionColor 597
Visible 597
RightButton2 598
AutoRoll 598
Cursor 598
Enabled 598
Glyph 598
Hint 598
Kind 598
UseCaptionColor 598
Visible 598
RightCaption 598
RightCaptionFont 598
TextStyle 599
PlainText 599
TagClose 600
TagOpen 600
Events 601
OnCaptionClick 601
OnImageClick 601
OnMouseEnter 601
OnMouseLeave 602
OnPaint 602
OnRightButtonClick 602
OnRightButton2Click 603

© 1998-2006, UtilMind Solutions®

16 acHeaderPanel 603

TacHeaderPanel 603
Properties 603
Background 603
Active 604
GradientColor1 604
GradientColor2 605
GradientDirection 605
Kind 606
Picture 606
PictureStyle 606
BackgroundHeader 607
Active 608
GradientColor1 608
GradientColor2 609
GradientDirection 609
Kind 609
Picture 610
PictureStyle 610
CaptionMenu 611
Flat 611
FlatBorderColor 612
HeaderHeight 612
Image 612
ImageAlignment 612
ImageIndex 612
ImageList 612
ImageMenu 612
ImageTransparent 612
PopupMenuHeader 612
RightButton 613
AutoRoll 613
Enabled 613
Glyph 613
Kind 613
UseCaptionColor 613
Visible 613
RightButton2 613
AutoRoll 613
Enabled 613
Glyph 613
Kind 613
UseCaptionColor 613
Visible 613
RightCaption 613
RightCaptionFont 614
RollOnDblClick 614
RollRegistrySaver 614
Enabled 614
IniFileName 614
IniSection 614
Options 614
RegKey 614
RegLocation 614

© 1998-2006, UtilMind Solutions®

Storage ... 614
ShowHeader ... 614
ShowSizeGrip ... 614
Text ... 614
TextAlignment ... 614
TextFont ... 615
TextStyle ... 615
PlainText ... 615
TagClose ... 616
TagOpen ... 616
TextLeft ... 617
TextTop ... 617
WordWrap ... 617
Events ... 617
OnHeaderCaptionClick ... 617
OnHeaderClick ... 617
OnHeaderImageClick ... 617
OnHeaderMouseEnter ... 618
OnHeaderMouseLeave ... 618
OnMouseEnter ... 618
OnMouseLeave ... 618
OnPaint ... 619
OnPaintHeader ... 619
OnRightButtonClick ... 620
OnRightButton2Click ... 620
OnRolled ... 620

17 acHostnameResolver ... 620

TacHostnameResolver ... 620
Properties ... 621
AutoResolve ... 621
CacheSize ... 621
IPList ... 621
IsOnline ... 622
LocalHostname ... 623
LocalIP ... 623
MonitorActive ... 623
MonitorInterval ... 624
Priority ... 624
RegistrySaver ... 625
Enabled ... 625
RegKey ... 626
RegLocation ... 626
Methods ... 627
Execute ... 627
Abort ... 627
ResolveCachedIP ... 628
Events ... 628
OnHostnameRetrieved ... 628
OnOnlineStatusChanged ... 629

18 acHTMLViewer ... 629

TacHTMLViewer ... 629
Properties ... 630
BorderColor ... 630

© 1998-2006, UtilMind Solutions®

BorderStyle ... 630
Color ... 631
HTML ... 631
Margins ... 631
Left ... 632
Top ... 632
Right ... 632
Bottom ... 632
NoScroll ... 633
NoStdPopupMenu ... 633
PopupMenu ... 633
WebBrowser ... 634

19 acImagesComboBox ... 634

TacImagesComboBox ... 634
Properties ... 635
AttachedLabel ... 635
Caption ... 636
Color ... 636
Cursor ... 637
Enabled ... 637
FocusOnClick ... 638
Font ... 638
Hint ... 638
ParentColor ... 639
ParentFont ... 639
ParentShowHint ... 640
PopupMenu ... 640
Position ... 640
ShowAccelChar ... 641
ShowHint ... 641
Spacing ... 642
SpacingKind ... 642
Transparent ... 643
Visible ... 643
WordWrap ... 644
Images ... 644
Items ... 644

20 acIPEdit ... 645

TacIPEdit ... 645
Properties ... 646
Align ... 646
Alignment ... 647
AttachedLabel ... 647
Caption ... 648
Color ... 649
Cursor ... 649
Enabled ... 649
FocusOnClick ... 650
Font ... 650
Hint ... 650
ParentColor ... 651
ParentFont ... 651
ParentShowHint ... 651

© 1998-2006, UtilMind Solutions®

PopupMenu ... 651
Position ... 651
ShowAccelChar ... 651
ShowHint ... 651
Spacing ... 651
SpacingKind ... 652
Transparent ... 652
Visible ... 652
WordWrap ... 652
AutoSelect ... 652
AutoSize ... 653
Color ... 653
ColorDisabled ... 653
IP1 ... 654
IP2 ... 654
IP3 ... 654
IP4 ... 654
IPLong ... 655
Text ... 655
Events ... 655
OnChange ... 655
21 acLabel ... 656
TacLabel ... 656
Properties ... 656
Blinking ... 656
DelayInvisible ... 657
DelayVisible ... 657
Enabled ... 657
CursorClick ... 657
FontClick ... 658
Color ... 658
Style ... 659
Used ... 660
FontHover ... 660
Color ... 660
Style ... 661
Used ... 662
Highlight ... 662
Color ... 662
ColorClick ... 662
ColorHover ... 663
Depth ... 663
Direction ... 663
Enabled ... 663
Shadow ... 664
Color ... 664
ColorClick ... 664
ColorHover ... 665
Depth ... 665
Direction ... 665
Enabled ... 665
Options ... 666
WordWrap ... 666
Events ... 666

© 1998-2006, UtilMind Solutions®

OnMouseEnter ... 666
OnMouseLeave ... 667
22 acLabeledComboBox ... 667
TacLabeledComboBox ... 667
Properties ... 667
AttachedLabel ... 667
Caption ... 668
Color ... 669
Cursor ... 669
Enabled ... 670
FocusOnClick ... 670
Font ... 670
Hint ... 671
ParentColor ... 671
ParentFont ... 672
ParentShowHint ... 672
PopupMenu ... 673
Position ... 673
ShowAccelChar ... 673
ShowHint ... 674
Spacing ... 674
SpacingKind ... 675
Transparent ... 675
Visible ... 676
WordWrap ... 676
Methods ... 677
SetSelection ... 677
Events ... 677
OnLabelClick ... 677
OnLabelDblClick ... 677
OnMouseEnter ... 678
OnMouseLeave ... 678
23 acLinksPanel ... 678
TacLinksPanel ... 678
Properties ... 678
FontClick ... 678
FontHover ... 679
ImageList ... 679
ImageSpacing ... 679
Items ... 679
LinksProperties ... 679
MarginLeft ... 680
MarginTop ... 680
Spacing ... 680
Events ... 680
OnLinkClick ... 680
24 acListView ... 681
TacListView ... 681
Properties ... 684
CheckOnClick ... 684
Cursors ... 684
Border ... 684
Divider ... 685

© 1998-2006, UtilMind Solutions®

Header ... 685
HotCursor ... 685
ScrollHorz ... 685
ScrollVert ... 686
EmptyListMessage ... 686
EmptyListMessageColor ... 686
HeaderColor ... 687
HeaderFont ... 687
HeaderMenu ... 687
AlignLeftItem ... 688
AlignRightItem ... 688
AlignCenterItem ... 689
AscendingItem ... 689
DescendingItem ... 689
BestFitItem ... 690
PopupMenu ... 690
HeaderStyle ... 690
ItemPopupMenu ... 691
RegistrySaver ... 691
Enabled ... 691
RegKey ... 692
RegLocation ... 692
ShowEmptyListMessage ... 693
ShowHeaderMenu ... 693
ShowScrollTips ... 693
ShowSortMark ... 694
ShowToolTips ... 694
SortColumn ... 695
SortDirection ... 695
SortMarkAlign ... 696
SortMarkImageAsc ... 696
SortMarkImageDesc ... 697
SortMarkKind ... 697
ToolTipOptions ... 698
Methods ... 698
BestFit ... 698
Clear ... 699
CopySelectionToClipboard ... 699
CountPerPage ... 699
DeleteSelected ... 700
ExportToHTML ... 700
ExportToTXT ... 701
FindCaption ... 701
GetColumnAt ... 702
GetItemAt ... 702
ImportFromTXT ... 703
IsScrollBarsVisible ... 703
SelectAll ... 704
SetColumnOrders ... 704
Events ... 704
OnAlignmentChanged ... 704
OnCheckBoxClick ... 705
OnColumnDragged ... 705
OnColumnResizeBegin ... 706

© 1998-2006, UtilMind Solutions®

OnColumnResizeEnd ... 706
OnCustomDrawSubItemEx ... 707
OnDrawHeaderSection ... 707
OnMouseEnter ... 708
OnMouseLeave ... 708
OnSettingsChanged ... 708
OnToolTip ... 709

25 acMRUComboBox ... 710

TacMRUComboBox ... 710
Properties ... 711
Align ... 711
AttachedLabel ... 712
Caption ... 713
Color ... 713
Cursor ... 714
Enabled ... 714
FocusOnClick ... 714
Font ... 715
Hint ... 715
ParentColor ... 715
ParentFont ... 716
ParentShowHint ... 716
PopupMenu ... 717
Position ... 717
ShowAccelChar ... 717
ShowHint ... 718
Spacing ... 719
SpacingKind ... 719
Transparent ... 720
Visible ... 720
WordWrap ... 720
AutoFill ... 721
AutoFillCaseSensitive ... 721
AutoFillDelay ... 722
MRUEnabled ... 722
MRUMax ... 722
MRURegKey ... 722
MRURegLocation ... 723
MRUSaveOnExit ... 723
SubmitButton ... 724
Methods ... 724
ClearMRU ... 724
Save ... 724
SetSelection ... 725
Events ... 725
OnMouseEnter ... 725
OnMouseLeave ... 725
OnSubmit ... 726

26 acNumberEdit ... 726

TacNumberEdit ... 726
Properties ... 727
Align ... 727
Alignment ... 728

© 1998-2006, UtilMind Solutions®

AttachedLabel ... 728
Caption ... 729
Color ... 730
Cursor ... 730
Enabled ... 730
FocusOnClick ... 731
Font ... 731
Hint ... 731
ParentColor ... 732
ParentFont ... 732
ParentShowHint ... 733
PopupMenu ... 733
Position ... 734
ShowAccelChar ... 734
ShowHint ... 735
Spacing ... 735
SpacingKind ... 736
Transparent ... 736
Visible ... 737
WordWrap ... 737
AutoSelect ... 737
Button ... 738
Cursor ... 738
Flat ... 738
Glyph ... 739
NumGlyphs ... 740
Hint ... 740
Kind ... 740
Width ... 740
Visible ... 741
Color ... 741
ColorDisabled ... 741
CursorBorder ... 742
Kind ... 742
MaxValue ... 742
Value ... 743
Methods ... 743
SetSelection ... 743
Events ... 743
OnButtonClick ... 743
OnButtonLeftMouseDown ... 744
OnChange ... 744
OnLabelClick ... 744
OnLabelDblClick ... 745
OnMouseEnter ... 745
OnMouseLeave ... 745
27 acPicturePanel ... 746
TacPicturePanel ... 746
Properties ... 746
Background ... 746
Active ... 747
GradientColor1 ... 747
GradientColor2 ... 748
GradientDirection ... 748

© 1998-2006, UtilMind Solutions®

Kind ... 749
Picture ... 749
PictureStyle ... 750
BevelStyle ... 750
Image ... 750
ImageAlignment ... 751
ImageIndex ... 751
ImageList ... 751
ImageMargin ... 751
ImageTransparent ... 751
Margin ... 751
ShowBevel ... 752
Text ... 752
TextAlignment ... 752
TextStyle ... 752
PlainText ... 753
TagClose ... 753
TagOpen ... 754
Events ... 754
OnMouseEnter ... 754
OnMouseLeave ... 755
OnPaint ... 755
28 acProportionalImage ... 755
TacProportionalImage ... 755
Properties ... 755
Alignment ... 755
ControlCanvas ... 756
DrawRect ... 756
FixedStretching ... 757
JPEGProperties ... 757
CompressionQuality ... 758
Grayscale ... 758
Performance ... 758
PixelFormat ... 759
ProgressiveDisplay ... 759
ProgressiveEncoding ... 760
Scale ... 760
Smoothing ... 761
Proportional ... 761
29 acQuickAboutBox ... 762
TacQuickAboutBox ... 762
Properties ... 762
AnimationDelay ... 762
AnimationEffect ... 762
Caption ... 763
Copyright ... 763
Cursors ... 763
Caption ... 763
Client ... 764
Border ... 764
BtnClose ... 764
Buttons ... 764
Links ... 764

© 1998-2006, UtilMind Solutions®

Description ... 765
Email ... 765
GradientBackground ... 765
Color1 ... 765
Color2 ... 766
Direction ... 766
Enabled ... 766
Icon ... 766
Localization ... 767
stAbout ... 767
stOK ... 767
stRegister ... 767
stRegisteredTo ... 768
stUnregistered ... 768
Registered ... 768
RegisteredTo ... 768
RegistrationURL ... 769
Shareware ... 769
ShowBevels ... 769
ShowHints ... 769
WebSite ... 770
Width ... 770
Methods ... 770
Execute ... 770
30 acRadioButton ... 770
TacRadioButton ... 770
Properties ... 771
Align ... 771
AlignmentHorz ... 772
AlignmentVert ... 773
BoxAlignment ... 773
Ctl3D ... 774
ParentCtl3D ... 774
WordWrap ... 775
Events ... 775
OnMouseEnter ... 775
OnMouseLeave ... 775
31 acRichLabel ... 776
TacRichLabel ... 776
Properties ... 777
Caption ... 777
MarginHorz ... 777
MarginVert ... 777
ShadowColor ... 777
ShowFrame ... 778
ShowShadow ... 778
TextStyle ... 778
PlainText ... 779
TagClose ... 779
TagOpen ... 779
Transparent ... 780
WordWrap ... 780
Events ... 780

© 1998-2006, UtilMind Solutions®

OnMouseEnter ... 780
OnMouseLeave ... 781
32 acSimpleTCPClient ... 781
TacSimpleTCPClient ... 781
Properties ... 781
AllowChangeHostAndPortOnConnection ... 781
AutoReconnect ... 781
AutoReconnectInterval ... 781
Connected ... 782
Host ... 782
Port ... 782
Methods ... 782
Send ... 782
SendStream ... 783
Receive ... 783
ReceiveStream ... 783
Events ... 783
OnConnected ... 783
OnDataAvailable ... 783
OnDisconnected ... 783
OnError ... 784
OnRead ... 784
OnSelfDisconnected ... 784
33 acSimpleTCPServer ... 784
TacSimpleTCPServer ... 784
Properties ... 785
AllowChangeHostAndPortOnConnection ... 785
Connections ... 785
Listen ... 785
LocalHostName ... 785
LocalIP ... 786
Port ... 786
Methods ... 786
Send ... 786
SendStream ... 786
Broadcast ... 787
BroadcastStream ... 787
Receive ... 787
ReceiveStream ... 787
Events ... 787
OnAccept ... 787
OnClientConnected ... 788
OnClientDataAvailable ... 788
OnClientDisconnected ... 788
OnClientRead ... 789
OnError ... 789
34 acSplitterPanel ... 789
TacSplitterPanel ... 789
Properties ... 790
Align ... 790
Background ... 790
Active ... 790
GradientColor1 ... 791

© 1998-2006, UtilMind Solutions®

GradientColor2 ... 791
GradientDirection ... 791
Kind ... 792
Picture ... 793
PictureStyle ... 793
ClipSplitRect ... 794
CursorHorz ... 794
CursorVert ... 794
MinSize ... 794
ProportionalPanes ... 795
RegistrySaver ... 795
Enabled ... 795
IniFileName ... 795
IniSection ... 795
Options ... 795
RegKey ... 795
RegLocation ... 795
Storage ... 795
SplitterStyle ... 795
SplitterPos ... 796
SplitterWidth ... 796
Events ... 796
OnPaint ... 796
OnSplitterPosChanged ... 797

35 acSQLQueryThread ... 797

TacSQLQueryThread ... 797
Properties ... 797
Priority ... 797
Query ... 798
WaitThread ... 798
WaitTimeout ... 799
Methods ... 799
RunSQL ... 799
Abort ... 800
Events ... 800
OnDone ... 800
OnException ... 801
OnWaitTimeoutExpired ... 801

36 acStatusControl ... 801

TacStatusControl ... 801
Properties ... 802
Control ... 802
PanelIndex ... 802
StatusBar ... 802

37 acStringListContainer ... 803

acStringListContainer ... 803
Properties ... 803
Count ... 803
Duplicates ... 803
List ... 804
MaxSize ... 804
NVSeparator ... 804
NVSeparatorMainSide ... 805

© 1998-2006, UtilMind Solutions®

Sorted ... 805
Names ... 806
Strings ... 807
Values ... 807
Methods ... 808
Add ... 808
AddStrings ... 809
Clear ... 810
Delete ... 810
DeleteFirstItem ... 810
Exchange ... 810
GetAndDeleteItem ... 811
GetFIFO ... 811
IndexOf ... 811
IndexOfName ... 812
IndexOfValue ... 812
Insert ... 813
LoadFromFile ... 813
LoadFromRegistry ... 814
SaveToFile ... 815
Sort ... 815
StoreToRegistry ... 816
Events ... 817
OnChange ... 817
OnChanging ... 817
38 acSystemImageList ... 818
TacSystemImageList ... 818
Properties ... 818
ImageSize ... 818
39 acSystemInfo ... 819
TacSystemInfo ... 819
Properties ... 820
Allow_Edit ... 820
ComputerDescription ... 820
ComputerName ... 820
ComputerWorkgroup ... 821
CPUArchitecture ... 821
CPUFrequency ... 821
CPUIdentifier ... 822
CPUsNumber ... 822
CPUType ... 822
CPUVendor ... 822
CurrentUserName ... 823
DefaultBrowser ... 823
DirAppData ... 823
DirAppDataAll ... 823
DirCache ... 824
DirCommonFiles ... 824
DirCookies ... 824
DirDesktop ... 824
DirFavorites ... 825
DirFonts ... 825
DirHistory ... 825

© 1998-2006, UtilMind Solutions®

DirMedia ... 825
DirMyPictures ... 826
DirNetHood ... 826
DirPersonal ... 826
DirPrintHood ... 826
DirProgramFiles ... 827
DirPrograms ... 827
DirRecent ... 827
DirSendTo ... 827
DirStartMenu ... 828
DirStartup ... 828
DirStartupAll ... 828
DirSystem ... 829
DirTemp ... 829
DirTemplates ... 829
DirWindows ... 829
DiskCSerialNumber ... 830
DiskCSerialNumberStr ... 830
InternetStatus ... 830
MemAvailPhisical ... 831
MemAvailVirtual ... 831
MemLoad ... 831
MemTotalPhisical ... 831
MemTotalVirtual ... 832
OSVersion ... 832
OwnerCompany ... 833
OwnerName ... 833
ScreenColorMode ... 833
ScreenResolution ... 834
ScreenSaver ... 834
SystemPathes ... 834
SystemUpTime ... 835
WinDirectXVersion ... 835
WinNT ... 835
WinProductID ... 835
WinProductKey ... 836
WinProductName ... 836
WinVersion ... 836
WinVersionBuild ... 837
WinVersionMajor ... 837
WinVersionMinor ... 837
WinVersionNumber ... 837
WinVersionSubNumber ... 837

40 acTextTemplateConverter ... 838

TacTextTemplateConverter ... 838
Properties ... 839
IgnoreCaseOfParams ... 839
Params ... 840
Template ... 841
Methods ... 841
Convert ... 841
ConvertStream ... 842
ConvertStrings ... 842
ConvertToFile ... 843

© 1998-2006, UtilMind Solutions®

ParamByName ... 843
ParamByValue ... 844
Events ... 845
OnParamChanged ... 845
OnTemplateChanged ... 845
41 acTrackBar ... 845
TacTrackBar ... 845
42 acTreeComboBox ... 845
TacTreeComboBox ... 845
Properties ... 846
AcceptOnDblClick ... 846
AttachedLabel ... 847
Caption ... 848
Color ... 848
Cursor ... 849
Enabled ... 849
FocusOnClick ... 849
Font ... 850
Hint ... 850
ParentColor ... 850
ParentFont ... 851
ParentShowHint ... 851
PopupMenu ... 852
Position ... 852
ShowAccelChar ... 853
ShowHint ... 853
Spacing ... 854
SpacingKind ... 854
Transparent ... 855
Visible ... 855
WordWrap ... 856
AutoComplete ... 856
AutoCompleteDelay ... 856
AutoCompleteIgnoreCase ... 857
Button ... 857
Cursor ... 858
Flat ... 858
Glyph ... 858
NumGlyphs ... 859
Hint ... 859
Kind ... 860
Width ... 860
Visible ... 860
ColorDisabled ... 860
CursorBorder ... 861
DropDownCount ... 861
DropDownWidth ... 861
EmptyItemImageIndex ... 862
EmptyItemText ... 862
Images ... 862
ItemIndex ... 863
Items ... 863
SelectedNode ... 864

© 1998-2006, UtilMind Solutions®

ShowEditor ... 864
ShowTreePathInEdit ... 864
StateImages ... 865
TreeOptions ... 865
TreePathSeparator ... 866
TreeView ... 867
Methods ... 867
AddPath ... 867
ExportToTreeNodes ... 868
FindAbsoluteIndexByNode ... 868
FindNode ... 868
ImportFromTreeNodes ... 869
ImportFromTreeView ... 869
SelectNodeByText ... 869
SetSelection ... 870
Events ... 870
OnAfterDropDown ... 870
OnButtonClick ... 870
OnButtonLeftMouseDown ... 871
OnCanSelectNode ... 871
OnCloseUp ... 871
OnDropDown ... 872
OnLabelClick ... 872
OnLabelDblClick ... 873
OnMouseEnter ... 873
OnMouseLeave ... 873
43 acTreeView ... 874
TacTreeView ... 874
Properties ... 875
AlwaysExpandRoot ... 875
AlwaysFullExpand ... 875
AutoExpand ... 875
BoldItem ... 876
BoldNode ... 877
ColorBack ... 877
ColorFrame ... 878
ColorText ... 878
ImageIndex ... 878
ImageList ... 879
ImageSpacing ... 879
ItemIndex ... 880
ItemIndexIs ... 880
ShowFrame ... 881
SwitchSelectionOnBold ... 881
SwitchToBoldOnDblClick ... 881
Text ... 882
Cursors ... 882
Border ... 882
ScrollHorz ... 883
ScrollVert ... 883
HotTrack ... 883
ItemPopupMenu ... 883
RowSelect ... 884
ToolTips ... 884

© 1998-2006, UtilMind Solutions®

Methods ... **884**
AddPath ... 884
FindNode ... 885
FindAbsoluteIndexByNode ... 885
ExpandTopLevelNodes ... 885
Events ... **886**
OnBoldItemBeforeSwitch ... 886
OnBoldItemSwitched ... 887
OnMouseEnter ... 887
OnMouseLeave ... 887

44 acUnixCrypt ... 888

TacUnixCrypt ... **888**
Properties ... **889**
Key ... 889
Salt ... 889
Result ... 889

45 acURLLabel ... 889

TacURLLabel ... **889**
Properties ... **890**
Email ... 890
Address ... 891
BlindCarbonCopy ... 891
CarbonCopy ... 891
Message ... 892
Name ... 892
Subject ... 892
Kind ... 893
URL ... 893
URLInNewWindow ... 893

46 acWebImage ... 894

TacWebImage ... **894**
Properties ... **894**
Alignment ... 894
CacheOptions ... 894
ControlCanvas ... 895
DrawRect ... 895
FixedStretching ... 895
JPEGProperties ... 895
HTTPAgent ... 895
HTTPPassword ... 895
HTTPUsername ... 896
HTTPProxy ... 896
AccessType ... 897
ProxyBypass ... 897
ProxyPassword ... 898
ProxyPort ... 898
ProxyServer ... 898
ProxyUsername ... 899
InternetOptions ... 899
ThreadPriority ... 900
URL ... 900
Methods ... **900**
Retry ... 900

© 1998-2006, UtilMind Solutions®

Abort ... 901
Events ... 901
OnAborted ... 901
OnDone ... 901
OnDownloadProgress ... 901
OnError ... 901
OnPasswordRequest ... 901
OnProxyAuthenticationRequest ... 902
OnUnsupportedFormat ... 903

Part XI Additions to the Delphi/BCB IDE 903

1 Additional Colors ... 903

2 Additional Cursors ... 904

Part XII Bonus Routines 906

1 Advanced GDI routines ... 909

acMakeTranslucentWindow ... 910
acMakeOpaqueWindow ... 911
acAnimateWindow ... 911
acFlashWindow ... 912
AnimateWindow ... 912
FlashWindowEx ... 914
SetLayeredWindowAttributes ... 914

2 Numerical conversion routines ... 915

DecToHex ... 916
HexToDec ... 916
DecToBin ... 916
BinToDec ... 917
DecToRoman ... 917
Int2x32ToInt64 ... 917

3 Shell utilities routines ... 918

ComposeEmail ... 918
GetExecutableByExtension ... 919
GetSystemImageList ... 919
MinimizeAll ... 920
ExploreFolder ... 920
OpenURL ... 920
OpenWithDlg ... 921
RepaintScreen ... 921
RestoreAll ... 921
RunControlPanel ... 922
RunCPL ... 923

4 String/Filename routines ... 924

SplitStr ... 924
LeftPart ... 926
RightPart ... 926
LeftPartR ... 927
RightPartR ... 928
PosR ... 928
SplitFileNameAndParams ... 929
IsValidEmail ... 930

© 1998-2006, UtilMind Solutions®

IncludeTrailingBackslash ... 930
ExcludeTrailingBackslash ... 930
IncludeLeadingChar ... 931
ExcludeLeadingChar ... 931
StringReplace ... 931
HasParamStr ... 932
FullParamStr ... 933
CaseAnsiCompareText ... 933
URLEncode ... 933
URLDecode ... 934
StateNameToStateCode ... 934
StateCodeToStateName ... 934
StrToStrList ... 934
FileReadString ... 935
FileWriteString ... 935

5 MySQL String routines ... 935

ESC ... 936
SQLDateTimeToDateTime ... 936
DateTimeToSQLDateTime ... 937
DateTimeToSQLDate ... 937
DateTimeToSQLTime ... 937

6 Date/Time routines ... 937

StrToDateTimeDef ... 938
IsLeapYear ... 938
DaysPerMonth ... 938
GMTToLocalTime ... 939
LocalTimeToGMT ... 939
GMT ... 939
InternetTimeToDateTime ... 940
DateTimeToFileTime ... 940
FileTimeToDateTime ... 941
SetFileTimes ... 941
SameDateTime ... 941

7 System Information ... 941

GetOS ... 942
IsNT ... 942
IsWin2k ... 943
IsXP ... 943
Is256ColorTrayIconAllowed ... 943
GetCPUSpeed ... 944
IsScreenSaverRunning ... 944
GetTempDir ... 944
GetSystemDir ... 945
GetWindowsDir ... 945
GetEnvironmentString ... 945
GetSystemIconIndex ... 946

8 Sound routines ... 946

Sound ... 946
NoSound ... 947
Delay ... 947
PlaySystemSound ... 948
StopSystemSound ... 948

© 1998-2006, UtilMind Solutions®

9 Easy Registry utilities ... 949
IsRegValueExists ... 949
RegReadStr ... 950
RegWriteStr ... 950
RegReadInt ... 951
RegWriteInt ... 951
RegEraseValue ... 951
RegEraseKey ... 951
RegReadFont ... 952
RegWriteFont ... 952
10 Internet utilities ... 952
IsOnline ... 953
GetLocalHostName ... 953
GetLocalIP ... 953
IPToStr ... 954
StrToIP ... 954
ParseURL ... 954
URLToHostName ... 955
URLToFileName ... 956
SockConnect ... 956
SockDisconnect ... 956
11 Miscellaneous ... 957
IsDirectoryExists ... 957
IsDirectoryEmpty ... 958
ExtractFileSize ... 958
FileSizeToStr ... 958
GetShiftState ... 959
GetKeyName ... 959
GetWorkArea ... 960
GetWindowTitle ... 960
PercentsOf ... 960
StartScreenSaver ... 961
LoadResourceToStream ... 961
ExtractResourceToFile ... 962
StreamToString ... 963
PutStringToStream ... 963
GetStringFromStream ... 964
MsgBox ... 965
InfoBox ... 965
ErrorBox ... 965
XORStr ... 966
ShutdownApplication ... 966
HideFromTaskList ... 966
IsMetricSystem ... 967
IsDebug ... 967
Wait ... 967
StopWait ... 968
LogToFile ... 969
ExecAndWait ... 970
ControlAtPoint ... 970
12 Russian code tables ... 971
DOSToKOI ... 971
DOSToWin ... 971

© 1998-2006, UtilMind Solutions®

WinToDOS 971
WinToKOI 972
KOIToDOS 972
KOIToWin 972

13 WindowsXP Themes 972

IsThemeEnabled 973
IsThemeActive 973
IsAppThemed 973
OpenThemeData 974
CloseThemeData 974
DrawThemeBackground 975
DrawThemeParentBackground 975
DrawThemedControl 976

Index 978

© 1998-2006, UtilMind Solutions®

1 Introduction

Welcome!

The Advanced Application Controls (AppControls) is the set of over **92** top quality multipurpose components for all versions of Delphi and C++ Builder for Win32. The package contains everything you need to add neat and, more important, truly professional appearance for your software, making development of great interfaces and internal utilities trully rapid. All components are highly integrated each other and extremely easy to use.

All components is completely automated, so using AppControls, in most cases you will do not need to write ANY line of additional code and don't waste your time to make cosmetic features and professional software interfaces!

Main Features

The AppControls contains components to make the forms to behave like Applicaiton Desktop Toolbars (like taskbar or MS-Office panel) or "magnetic" forms of WinAmp player. Components to add the context-sensitive help without any help-files, additional custom buttons on the form title bar and associated menu items in the system menu; show smooth matrix- or win2k-based animation effects; balloon-style hints; additional color cursors for any VCL or non-VCL object of the forms.

Your applications will be able to auto-run itself on Windows startup with restoring previous forms placements. Upload and hold any files inside of the Delphi/BCB forms. Create shortcuts to the files and read the information from existing shortcuts. Install new file types and associations into the Windows shell. Download files from the Web via HTTP or post data/files to the CGI programs. Auto-upgrade itself from the Web at once a newer release of your application will available. Hook the form messages and hidden application events, sniff the URLs and events from each running instance of Internet Explorer, get the list of all running processes and active TCP/UDP connections, hook ALL key presses, mouse clicks and mouse movements within entire application… and a whole bunch of other great features! See **Application Controls** overview for more details.

Visible controls

Till AppControls **v2.0** the package has only invisible utilities for improving the application look at functionality. However we decided to include additional set of improved visible controls and system utilities. Now it contains:

- advanced labels with number of neat visual effects;
- RichLabel with HTML formatting;
- standard controls with multi-line captions;
- edit boxes which able to have custom button at the right edge of control (acEdit, acNumberEdit and acIPEdit);
- several MS-Outlook style panels (HeaderPanel, SplitterPanel and PicturePanel);
- combobox with MRU history list;
- advanced TreeView and ListView which automatically recognize the data format and sort the columns, and shows arrow-style sorting mark in the header;
- TreeComboBox component which displays the treeview inside the drop-down box and supports AutoComplete feature;
- StatusControl component which can insert ANY visible control to the section of the status bar;
- easy to use TCP/IP client and server components which lets you easily implement your own Internet protocol;
- Internet Explorer-based HTMLViewer component, which allows to specify the HTML content without navigation through the Web;
- components which can automatically hook all events of all Internet Explorer windows, auto-complete the Web forms;
- RSS feed processors;

© 1998-2006, UtilMind Solutions®

- much more, see **Advanced Visible Controls** overview for more details.

All components is highly integrated each others and extremely easy to use. You will not have any problems with components compatibility *(for example when you changing the AlwaysOnTop property in one component, let's suppose, acAppBar, the Topmost property will be changed in acFormTopmost and acFormHook components accordingly)*. Since we ourselves creating the end-user software we are maximally interested in their stability and compatibility.

Compatibility

AppControls compatible with Delphi 2/3/4/5/6/7/2005, Borland Developer Studio 2006, and C++ Builder 3/4/5/6. Has been tested on Win95, Win95OSR2, Win98, Win98SE, WinME, WinNT4, Win2000, WinXP and Win2003.

Improvements

Currently the AppControls pack contains 92 components. However, we constantly updating the package and adding more useful utilities and new great features to existing components. If you have any suggestions to help us to improve the AppControls — please **contact us** and never hesitate to do it for any question.

Updates

Registering of the AppControls entitles you to unlimited support (via email or personal panel at AppControls.Com), minor updates indefinitely and major version updates for 6 month from date of purchase.

WWW

Please visit the AppControls home page at **http://www.appcontrols.com**.

Advanced Application Controls (http://www.appcontrols.com)
Copyright © 1998-2006, Utilmind Solutions. All Rights Reserved.
Documentation created with **Help&Manual**, the best help authoring tool.

2 Overview (Part 1: Application Controls)

1. **acAppAutoRun** - allows to automatically execute your application on each Windows startup. Unlike others similar components, it have some additional features, like displaying message on Windows login, plus have special event which informs you about automatic start. acAppAutoRun is successor of acCaptionButton and may have built-in caption button on form's title bar and associated menu item in the system menu.

2. **acAppStartCounter** - used to count up the number of program startups and log the time when the program was started first time to the registry. If you would like to show welcome message on first start, or reminder message on every twentieth restart or any other operation regarding the count of program startups — this component is that you need! Also it can be useful for you if you want to measure the lifespan of the application and record it to some remote database when the application uninstalls.

3. **acCronJob** - the thread-based alarm implementation for Delphi/BCB which acts like *cronjob*

utility in Unix. The component produces periodical OnAlert events by schedule specified in CRON format (actually it uses extended CRON format with new "seconds" field, to let the event to be triggered even every second).

4. **acAccurateTimer** - the thread-based timer which periodically triggers the OnTimer events by specified Interval. Unlike standard TTimer, which based on Win32API and thus loses timer messages on high application overload, this timer runs in the separate thread and accurately triggers each OnTimer event at right time. It allows to specify the Interval with higher precision (less than 55msec, limited in Win9x) and does not dependent to the system-wide limitations.

5. **acAwayTimer** - monitors user activity and increase its time counter when keyboard and mouse are inactive (user distracted from PC or goes away). Also it triggers special event when system is about to start the screensaver, or after some defined period of user inactivity.

6. **acAppEvents** - intended for modifications of some hidden application properties (of the TApplication class) and hooking the application events (such like idle, activate / deactivate, minimize / restore etc). Besides usual features it's able to animate the application icon, hide the taskbar icon and fix the smooth animated minimize/restore effect for Delphi 2/3/4. The acAppEvents hooks everything application-related. It can even hook ALL key presses and ALL mouse clicks and movements within entire application.

7. **acAppCursors** - allows to specify the cursor images for non-client parts of your application. You can specify your own default cursor for all controls, custom cursors for form captions, regular caption buttons, scroll bars, borders, menus and menu items and so forth… One component can change appearance of cursors for *entire application*, even in standard Windows dialogs (such as OpenDialog / SaveDialog), that you are using in your program.

8. **acOnlyOne** - detects already running instances of the application and disallow to start of second copy of the same program.

9. **acTrayIcon** - handy component for manipulating the system tray icons in the taskbar status area. Allows using the animated icons and contains the number of great features such like additional caption button on form's title bar and menu item in the system menu. In the Windows 2000 and higher it supports the cartoon-style balloon tooltips which appears after minimizing of the form to the system tray (a non intrusive way to display an explanatory message for a status bar icon is to implement a balloon tooltip with its stem pointing to the icon). Also it automatically detects when Windows Explorer crashes or restarts and automatically restores the icon in the tray. Contains a lot of various features which allow you to associate your forms with tray icons and configure the icon's behaviors without ANY single line of code!

10. **acShortcut** - creates shortcuts (also known as shell links) to files and folders in any location, and determines detailed information about existing shortcut files (.lnk).

11. **acExtAssociation** - lets an application to read, modify, install and uninstall the file types (extension associations) in the Windows shell. Component provides with detailed information about any file types installed to the system (ie. pas, .wav, .doc, .gif, .html), determines location

© 1998-2006, UtilMind Solutions®

to executable file, argument string, media-type, icon file, icon index and two icon images (32x32 and 16x16) associated with specified file type.
Also, with acExtAssociation you can install new file types or modify existing, changing the location of executable file or an icon image, associated with specified file type. For example, you can install ".yourdatafile" extension, specify the executable program for this file type and point the file icon that will associated with this file extension in the shell. After opening the any file with ".yourdatafile" extension (clicking the file icon), the application associated with this file type will be executed to process some data from ".yourdatafile".

12. **acClipboard** - monitors the clipboard for changes and triggers special event when the content of the clipboard changes. Also it provides with super easy way to retrieve and put to clipboard the Bitmap images, plain or unicode text and list of files (the bitmap, plain text or file list are usual properties, so the clipboard content can be modified even at design-time!). In case if you are working with some more complicated or custom clipboard formats, you can use several special methods which can help you to access the clipboard just like you working with usual strings or streams.

13. **acSendMail** - used for sending e-mail messages via SMTP servers. The messages can be either in plain text or HTML format, contain attachments or embedded images for HTML contents. You can use the acSendMail for dispatching of messages by mailing lists, specifying multiple recipients.

14. **acAutoUpgrader** - able to automatically upgrade from the Web any Delphi/BCB application. The AutoUpgrader will check your website for the newer releases of your software and, if the newer version is available — it will download and update all files of your application "*on the fly*". All this without a *SINGLE LINE* of code!
The acAutoUpgrader contains built-in Application Update Wizard (screenshot #1), built-in multi-language support (it automatically recognize language on user's PC and show all messages in native language (screenshot #2)). Current version automatically translates all wizard's content to more than 25 languages: English, Spanish, German, French, Russian, Portuguese, Italian, Dutch, Danish, Finnish, Catalan, Chinese, etc. However, if you don't want to use built-in Wizard, you can make customized progress-dialogs using numerous events.
With AutoUpgrader your customers will use only *latest* versions of your software!

15. **acHTTP** - extremely powerful and easy to use WinInet-based HTTP client component which allows to post and get any data from the Web via HTTP protocol. With acHTTP you can grab Web pages, download files and documents (or only their headers without the content), get results of the CGI programs (for example, results of web-based search engines / databases), or even **upload files** to the Web-based programs. In case if you want download documents from local intranet — just specify file:// prefix in the URL instead of http://.

The acHTTP can grab web contents both in binary and text formats, supports cache of Internet Explorer, can pause and resume broken downloads, read data from password protected directories and automatically supports several proxy authentication schemes (basic, digest, NTLM etc).

16. **acHTTPDownloadQueue** -

17. **acRSSFeed** -

© 1998-2006, UtilMind Solutions®

18. **acRSSFeedContainer** -

19. **acAppBar** - lets your forms to behave like an Application Desktop Toolbar — to dock on the edges of the screen like usual taskbar or MS-Office panel. Though appbars are usually docked on an edges of the user's screen, they also can float as usual windows.
When the AppBar is anchored to the screen edge, it can be automatically hidden from screen when other window activated and popup on screen again when user point the mouse to the thin line on screen edge. User can dock or undock the appbar just moving it with mouse pointer. All automatic movements of the appbar can be displayed with smooth sliding effect. Component contains a whole bunch of neat additional features, such like docking and sizing rules, float restrictions, registry saver, animation effects and so forth...

20. **acCaptionButton** - applies an additional custom button to the title bars of your forms + special menu item associated with this caption button to the system menu. Contains **a lot** of customization features such like popup menus, multiple button states, custom tooltips, custom cursors etc.

21. **acAnimationEffect** - adds a smooth animation effects on opening and closing of your windows and dialog boxes. Available effects: *Spin*, *Vortex*, *ScatterGather*, *Spike*, *Fireworks*.

22. **acWin2kEffects** - allows to produce some special effects of the Windows 98 / 2000. You can show roll, slide and alpha-blending animation effects on showing and hiding windows, make your forms translucent, flash the caption bars and taskbar icons.

23. **acFormHelp** - adds the context-sensitive help features to your Delphi/BCB forms without any bulky help files. It traps the context-sensitive help calls and creates its own popup windows from a control's hint. You can choose whether to interpret the hint string as plain text or as kind of rich text allowing you to apply different fonts colors, styles and line breaks. Don't worry about your hints — FormHelp uses the secondary part of a control's hint that is separated by a vertical bar "|". Mouse hints still works as well. With FormHelp, neither help context numbers nor extra help files are required to display context sensitive help. FormHelp's popup windows looks and feels like native context help in standard Microsoft's applications. Even if you're using regular help files, the FormHelp will be great addition and vice versa.
In addition, it can apply the "What's This?" menu item to every control with context-sensitive help in the secondary part of the Hint property.
For easement of context-help authoring, component contains a WYSIWYG help designer (drop acFormHelp onto your form and try to edit secondary part of Hint property of any visible control like TButton, TCheckBox or TGroupBox).

24. **acFormHints** - applies the balloon hints for your controls instead of regular, rectangular hints. Balloon hints are displayed in a cartoon-style "balloon" with rounded corners and a stem pointing to the tool. They can be either single-line or multi-line. You can specify the default balloon direction, text alignment, degree of transparency and other features that affect on hint appearance. One component changes appearance for all hints in entire application.

25. **acFormMagnet** - allows to magnetize your forms to a screen edges and to other forms. Forms can be attached each to others, so when user moves the form, others, attached forms

© 1998-2006, UtilMind Solutions®

will be moved too. With acFormMagnet, your forms can looks and feels like windows of famous WinAmp player.

26. **acFormSystemMenu** - encapsulates the properties and methods of the system menu and allows to work with system menu of your form as with usual TPopupMenu component. You can disable and delete regular menu items, or insert custom menu items to the existing system menu.

27. **acFormTopmost** - controls Z-order of your forms and allows to place your form over other windows. Since acFormTopmost is successor of acCaptionButton, it also able to add the custom button on form's title bar and associated menu item to the system menu. Add "Always on top" feature to your forms without single line of code!

28. **acFormRoller** - allows to reduce or expand (roll up / down) Delphi/BCB forms without minimizing them to taskbar or system tray, giving the impression that the form is shrinking or expanding. Rolling process can be displayed with smooth animation. Since the acFormRoller is successor of the acCaptionButton, it's also able to add the custom button on form's title bar and associated menu item to the system menu.

29. **acFormPlacementSaver** - saves and restores the form placement and Z-order from system registry. Just drop this component onto your form and component will remember the form's placement on each new program restart.

30. **acFormSizeRestrictions** - set up the restrictions to the form size and controls minimum and maximum tracking size of your windows.

31. **acFormResizeGrip** - attaches the SizeGrip rectangle to the bottom-right corner of the form at run-time. This allows the user to resize the form dragging this grip.

32. **acFormBackground** - paints the background (wallpaper) on your form using various gradient effects or bitmap image. Supports 3 smooth gradient effects and 4 styles of drawing the picture as wallpaper: tiled, stretched, tiled with zigzag (like bricks in the wall), and aligning the picture by center of form.

33. **acFormHook** - hooks all windows messages (WM_*xxx*) that passes to your form and allow their handling in special events + allow users to drop files onto your forms. Contains number of great features such like disabling of regular Close button on form's title bar, placing the form on top over other windows, removing the title bar from form, allowing users to move your forms by client area, and so forth…

34. **acEmbeddedForm** - used to insert the form and all its content to any successor of TWinControl (i.e: Panels, GroupBoxes, Labels, pages of PageControl etc). This is better than usual frames that comes with Delphi/BCB beginning from v5 since:
1. You can *dynamically* change the container of the embedded form at run-time (dynamically insert or remove the form from any successor of TWinControl).
2. Still have all functionality of usual form (some controls can be placed only to form or its

© 1998-2006, UtilMind Solutions®

successor).

35. **acFileStorage** - is able to store (upload) files onto your Delphi/BCB forms. If your software requires any additional files (.DLL's, WAV's, .TXT's etc), these files could be uploaded straight onto your form and be extracted from executable file at run-time. Also you can access to stored files directly from memory without extracting them to the disk.

36. **acWavPlayer** - used for playing default system sounds, associated with some certain events, or custom Wave-Audio (*.wav) files uploaded onto your form.

37. **acCursorLocker** - able to lock the movements of the mouse pointer within the Delphi/BCB forms or within bounds of any control on these forms.

38. **acThread** - extremely easy to use enhancement of standard TThread class. All properties and event handlers can be specified directly in the Object Inspector.

39. **acThreadSafeLog** - the log-writter component, which can be safely accessed from various different threads simultaneously, without additional synchronization. It can write the logged text both to specified file or "memo" control, and automatically inserts the specified prefix before each line. The prefix of each record can contain the current time, thread identifier and number of CPU ticks passed from application startup. Also the acThreadSafeLog can automatically send email message with last *N* log entries to administrator in case of fatal error.

40. **acAppGlobalSwitch** - the small utility which may help you to control some general option (boolean value) and its changes in entire application. When you change the boolean value in one acAppGlobalSwitch, other components connected to it, will reflect change, and trigger special event.

41. **acAppGlobalAlert** - the small utility which helps you to receive special "alert events" in various places (forms, units) of your application, when you need to take some specific action in another forms.

42. **acInfoBox** - quick and simpe implementation of smart dialog boxes. You probably already seen these smart dialog boxes with some notification message and *"Don't show me this message again"* option below text. If user will find message annoying and don't want to see this notification again, it simply check this box and message will not appear anymore.
acInfoBox can show the message with rich-formatted text with graphic icon at the left side of dialog, and contains the corresponding check box (with customizeable caption). The state for this check box is kept in registry. All implementation with just one line of code.

43. **acProcessList** - used to retrieve the list of all running system processes. You can retrieve the full list simly calling the Refresh method and gain access to the information about each process, using the component as usual TList. TacProcess objects contains the information about process and has Kill method which able to imediately terminate the process.

© 1998-2006, UtilMind Solutions®

44. **acConnectionList** - retrieves the list of all active Internet connection and returns information in the form similar to **NetStat** tool. You can retrieve the list using Refresh method and get access to the information about each connection, using the component as usual TList, with objects which contains detailed information about connection (local and remote port and host name, protocol, and even possible software behind the protocol).

45. **acIESniffer** - monitors all running instances of Internet Explorer (or just some certain browser window(s)), sniffs the URL address from address line and gives an access to all their properties, methods and events (without implementing/installing ANY Browser Helper Objects). The component can work either in stand-alone Delphi/BCB application or can be put to Explorer Toolbar written with Delphi/BCB. The IESniffer allows:

- list the URLs which currently are available in the address lines of each browser window, and to be notified when the address changes;
- detect when new Internet Explorer window appears and when user close the window;
- hook each event of the Internet Explorer, modify its properties and call their methods, just like you are using usual TWebBrowser component;
- retrieve and modify the content of each page (its text, tags, links, images etc), for example you may highlight some text, remove or change the text, tags etc;
- automatically fill the Web forms;
- redirect navigation to different URL when some keywords are detected in the address line;
- automatically redirect searches from default MSN to another specified search engine (it works when user enters search terms right in address line without specifying correct address);
- prevent popup windows to be opened (documentation deminstrates an example how to eliminate ALL popup windows), and so on…

46. **acIESnifferAutoFillUserInfo** - the "plug-in" for acIESniffer component, used to fill the Web forms with specified information.

Advanced Application Controls (http://www.appcontrols.com)
Copyright © 1998-2006, Utilmind Solutions. All Rights Reserved.
Documentation created with **Help&Manual**, the best help authoring tool.

3 Overview (Part 2: Advanced Visible Controls)

47. **acLabel** - lets you to create eye-catching captions for your application. This is improved label control which able to show the shadow and highlight effects, twinkle captions, and change the appearance when the mouse hovers or clicks the label.

48. **acURLLabel** - the text label used to redirect user to the Web page, or to compose email message to specified recipient, when user clicks this label.

49. **acRichLabel** - the label which able to show the rich text. Some words or characters can be bold, italic or underlined, and you can specify the custom colors for various parts of text. At design-time you can edit the rich text using WYSIWYG editor for Caption property.
The rich text used in this controls is not standard RTF or HTML, however it supports kind of tags, which also used in the context-sensitive help of the acFormHelp component.

50. **acCheckBox** - improved TCheckBox, which supports multi-line captions and able to have non-3D (flat) border for the checkbox. Contains all needed properties for flexible adjustment of multi-line text within client area of control and alignment within its container.

51. **acRadioButton** - improved TRadioButton, which supports multi-line captions and able to have non-3D (flat) border for the radio button. Contains all needed properties for flexible adjustment of multi-line text within client area of control and alignment within its container.

52. **acCheckGroupBox** - easy way to implement to your forms a group box with a checkbox instead of usual caption. When the checkbox is checked, all controls within a group box is enabled. When user uncheck the checkbox, all controls within a group became automatically disabled.

53. **acSplitterPanel** - special panel designed to split itself into two areas, or panes, either horizontally or vertically. Each pane can be easily customized with number of properties (i.e: you can specify minimal size for pane, specify the splitter behavior and appearence, make the panes to be resized proportionally when user resize the window etc). The acSplitterPanels can be nested to create multi-paned splitter regions. Also, it can automatically save and restore the splitter position from the registry.

54. **acHeaderPanel** - implements the functionallity of the header panel which you probably seen in the MS-Outlook or MSN Messenger.

55. **acPicturePanel** - very customizeable panel with caption and icon near the panel contents. This sort of panels very commonly used in the Microsoft-style dialogs. See the screenshot and detailed description inside.

56. **acLinksPanel** - panel with links (labels) and the small icons at the left of labels. You can specify the font and image for each label and hook the event when user clicks the particular label.

57. **acDragPanel** - panel which lets user to move and dock the owner form dragging the panel that covers this form; or simply move the panel within its container. Works either with normal or MDI forms. The DragPanel is great addition to acAppBar.

58. **acButton** - improved TButton, which supports multi-line caption and can be aligned within its container.

59. **acBitBtn** - improved TBitBtn (button with the bitmap glyph), which supports multi-line captions, custom background colors and can be aligned within its container.

60. **acButtonsPanel** - is an easy and elegant way to put standard buttons (like OK, Cancel, Apply, Help or some custom buttons) to your dialog forms. Supports vertical and horizontal buttons (TacBitBtn's), can show the button glyphs, the divider between buttons and the dialog

© 1998-2006, UtilMind Solutions®

contents and even size grip at the bottom-right corner.

61. **acEdit** - enhanced TEdit control. Can align the text by center or right edge control, able to have custom button with bitmap image, can automatically validate entered characters and disallow to type some unwanted letters.

62. **acNumberEdit** - enhanced edit control used for input of the numerical values in decimal, hexadecimal and binary format. It can align the text by center or right edge control and able to have the button with bitmap image, at the right side of edit-control.

63. **acCurrencyEdit** - the edit box used to easily edit currency values with 2 decimal digits (both positive and negative). Returns typed value as Value property.

64. **acIPEdit** - control for editing the IP addresses. This is 100% native VCL component and contains a little bit more features than ActiveX "IPAddress".

65. **acComboDatePicker** - is the panel with group of 3 combo-boxes used to quickly select the date (day, month and year). You can specify the range of years which can be selected, and selectable values for the Day box will be specified automatically, accordingly to number of days per month and leap year. The control contains the number of properties which specifies its look, i.e: date format, text for labels "attached" to the selection boxes, width of boxes, fonts, etc.

66. **acLabeledComboBox** - the ComboBox control with the label attached to control.

67. **acMRUComboBox** - the edit control with the drop-down list which can save and restore the MRU history list (list of most often used lines) from the registry. When user type the text, the combobox can automatically complete the input with the text taken from the history list.

68. **acImagesComboBox** - the enhanced combo box which lets you to specify images for every item of the drop-down list.

69. **acTreeComboBox** - the combo-box with the TreeView embedded to its drop-down window. So, instead of searching and selecting the necessary list item from huge list of drop-down window, user will be able to see an elegant tree-like structure, select the tree nodes, expand and collapse them, export and import the nodes to another TreeView controls and so on.
Like in standard TreeView, the TreeComboBox can display images for each tree node, show image in editor, accept the selection either on one or double click, contains the label attached to control, customizeable image for button + many other neat features, result of combination of ComboBox and TreeView.

70. **acTreeView** - slightly improved standard TreeView control, which allows to mark out some certain node with bold text, color, background, frame around the node, and small picture at the right side. Though, in general, this control intended for those who still develop the apps using Delphi 2/3 or BCB 1/3. The acTreeView compatible with any 32-bit Delphi/BCB versions, but supports ALL features of TTreeView for Delphi 5/6, plus the possibility to choose custom cursor

© 1998-2006, UtilMind Solutions®

images for non-client parts of TreeView control — border, horizontal and vertical scroll bars. new! Added methods that able very qucikly find the node or its absolute index in the TreeView with big number of items, can save the nodes structure to the file or registry.

71. **acListView** - the advanced ListView which can automatically sort columns, displaying the arrow-style sorting mark, which represents the sorting direction (like in the MS-Outlook). It can sorts items automatically and does not require any additional programming. However, the custom sorting are still supported. The acListView recognizes the format of list items and can automatically sort the **text**, **numerical** and **date/time** values when the sorting rules, or any list item changes.
Contains a lot of additional features, such like popup menus for the header columns, tooltips for the list items which doesn't fits the column size (complete customization of tooltip appearance and placement), custom tooltips for every list item, custom cursors for various non-client parts of ListView. It's also able to automatically save and restore sorting rules on exit / startup of your program. Can export all columns and items to TXT or HTML format.
If you're still using old versions of Delphi 2 / C++ Builder 3 you will also pleasantly surprised that the acListView supports features which become available in higher Delphi versions. This is support of XP-themes, grid lines, row selecting, flat scroll bars, hot tracking and hot track styles and so forth…

72. **acDBListView** - the database-aware version of acListView component. Were added just one property — DataSource. When you select the active data source which points to some TQuery component, the acDBListView will be filled with fields / records, accordingly to the SQL statement and specified columns. However the acDBListView can be used just for displaying database records, not for editing.

73. **acTrackBar** -

74. **acSystemImageList** - gives access to the system image list. That's the image list which all icons (32x32 or 16x16) for various file types that Explorer uses. All these icons represents the files within the Windows shell.

75. **acProportionalImage** - used to show the images with correct aspect ratio, so you can be sure that the image will be fully displayed in the image control without any distortions. Also this component allows to align the image within control, determinate the bounds of actual image wihin control, can stretch image correctly (unlike standard TImage which contains a bug and break big images on stretching), gives an access to properties of JPEG image at design-time.

76. **acWebImage** - descendant of acProportionalImage component (see above), which displays the picture located somewhere in the Web. All you need to do is just specify the URL of the picture and component will automatically download by HTTP protocol to display.

77. **acStatusControl** - used to insert any visible control (button, panel, checkbox etc) to the section of the status bar.

78. **acHeader** - the control that has the functionality of Microsoft Outlook Header. Using this control you will be able to create highly customizable Outlook-style headers (rich-formatted text

© 1998-2006, UtilMind Solutions®

and icon at the left or right side of header, two customizeable buttons etc) and integrate them in your applications with other controls from the pack.

79. **acApartPopupMenu** - the PopupMenu component which properly works in several instances of application running as ActiveX control in *Apartment* thread model.
If you ever had pain adapting the standard PopupMenus (with single PopupList object per application) when written DLL forms, in example, for Internet Explorer toolbars, then this popup menu is what you need.

80. **acHTMLViewer** - easy to use HTML viewer, based on WebBrowser control of Internet Explorer. It allows to specify the HTML content without navigation through the Web, and provides easy access to some frequently used properties, intended to change the look and behavior of the emdedded browser window.

81. **acSystemInfo** - provides detailed system information such like machine name, current user name and organization, CPU info and frequency of main processor, serial number of hard disk, DirectX info, paths to the system directories, executable file for default browser, current screen saver, screen resolution and color mode and so forth.

82. **acDateTimeCalc** - calculates number of years, months, days, hours, minutes, seconds and milliseconds between two dates specified by *BeginTime* and *EndTime* properties.

83. **acUnixCrypt** - ultra fast *crypt* implementation for Delphi/C++ Builder. It works exactly as **crypt(key, salt)** Unix command (one-way encryption algorithm), which used to encrypt passwords in .htpasswd files in password protected Web directories.

84. **acStringListContainer** - can be used to specify the string lists at design-time and hold the sets of strings right in the form. Contains some additional features which allows to work with the lists of "Names=Values" (strings separated on two parts, where first part is some "name" and second is "value", separated with some character). Can disallow adding of the duplicate string independently of whether the list is sorted or not (unlike usual TStringList). Contains the methods to put and retrieve the strings in FIFO manner (allows to specify the list capacity (maximum number of strings), so if the list is full and the new string is about to be added, the first will be deleted previously, plus has the method which automatically delete the first string upon retreiving). Also it can store and retrieve the strings from registry.

85. **acTextTemplateConverter** - the utility which translates some specified %keywords% inside the text into some specified values. It can be used as plug-in for acSendMail and acHTMLViewer components to replace some keywords in email/HTML templates (like %recipient_email% or %sender_name%) to their actual values.

86. **acSQLQueryThread** - used to *Open* the SQL queries in separate process, avoiding suspendings of the main application thread.

87. **acSimpleTCPServer** - just very simple TCP Server socket implementation.

© 1998-2006, UtilMind Solutions®

88. **acSimpleTCPClient** - just very simple TCP Client socket implementation.

89. **acHostnameResolver** - used to retrieve the host name by IP address or list of IP addresses, without freezing your application (it resolves hostnames in separate thread, because this lookup process can take rather long time and may completely freeze your application).
This component contains the internal cache for host names which arelady resolved, so it can quickly lookup for cached hostname and not use additional CPU time and bandwidth to resolve the same IP serveral times at row. Plus, the cache can be stored in system registry for further use after program restart.
Also, the acHostnameResolver can be used as simple monitor to check whether Internet connection is active, and to be notified when current PC goes online or offline (online status changes).

90. **acControlKeyState** - can determinate states of *-Caps*, *-Num*, *-Scroll*, *-Control*, *-Shift* keys and change their conditions by blinking the LEDs on the keypad.

91. **acDesktopWallpaper** - will implement all needed functionality to make operations with the Desktop Wallpapers neat and easy.

92. **acQuickAboutBox** - applies simple but cool looking "about" box for your freeware and shareware application. Just drop component onto your main form, set several properties such like program name, short description, copyright notice, email address and URL to your website and About box is done. To show it — just call Execute method.

Advanced Application Controls (http://www.appcontrols.com)
Copyright © 1998-2006, Utilmind Solutions. All Rights Reserved.
Documentation created with **Help&Manual**, the best help authoring tool.

4 Versions History

23.10.2006 3.8

Added 3 new components: **acAppGlobalAlert** and **acAppGlobalSwitch**. Both are small utilities which helps you to control some global application actions in different forms of application; and **acCurrencyEdit**, the component which can be used to easily edit currency values with 2 decimal digits (currently it's rather simple but new features will added in next AppControls release).

Fixed 4 bytes memory leak in **acThread** class (used in acHTTP, acAutoUpgrader, acSendMail and other components).

Changed *CheckUpdate* method in **acAutoUpgrader** component. Added posibility to just check whether update available, without any message boxes and without initiation of the update itself.

Added *OnBeforeSendRequest* event to **acHTTP** component.

Added *ShowGoOnlineMessage* property and *IsGlobalOnline* method to **acHTTP** component, plus added *ioIgnoreUnknownCertificateAuthority* option to the *InternetOptions* property.

Added processing of Alt+Backspace to **acEdit** component and its descendants (unfortunately it was

not supported before by our mistake).

The **acFormHints** component now can be used in ActiveX-based forms in apartment application model.

…and many other minor improvements and fixes…

01.01.2006 3.7.6
Installation fixes for Delphi 2006.

20.12.2005 3.7.5
Update with support of **Delphi 2006**, plus some minor fixes.

29.09.2005 3.7.4
Fixed **acSendMail** component (it appears that some SMTP servers are incorrectly process huge MimeEncoded email messages without linebreaks, so SendMail now breaks the lines in MimeEncoded attachments.

25.09.2005 3.7.3
Fixed **acHTTP** component, now all non-latin characters over 7bits are correctly "URLEncoded" for POST queries.

Fixed issues with displaying effects on MDI windows using **acAnimationEffect** and **acWin32Effects** components;

Fixed memory leak in **acTrayIcon** component, occured when application runs minimized at startup.

13.07.2005 3.7.2
Fixed incompatibilities with some 3rd party unicode controls.

Fixed bug in **acInfoBox** component ("property does not exists" message, which also could occur on using AutoUpgrader component).

29.06.2005 3.7.1
Added *OnUploadCGITimeoutFailed* event to **acHTTP** component.

Bugfix in *OpenURL* function, now opens URL in any default Web browser, including FireFox.

14.06.2005 3.7
Added 4 new components: **acHTTPDownloadQueue**, **acRSSFeed**, **acRSSFeedContainer** and **acTrackBar**. Unfortunately they are not documented yet, the description on these components and their features will be shortly, in next release (although they are fully functional and easy to use, most features are self-describing by properties and events).

Added *Authentication* feature to **acSendMail** component (allows to specify username and password for SMTP server).

Fixes in **acHTTP** component:

© 1998-2006, UtilMind Solutions®

- it was not returned from "Busy" state in case of file creation error, when components downloads some data to specified OutputFileName;
- added OnOutputFileError event;
- removed HideOutputFile property and added OutputFileAttributes property instead (allows to specify the file attributes for downloaded file, when it incomplete (being downloaded), and complete);
- implemented "rollback" feature which starts downloading again from beginning in case if content of Web file/document has been changed while downloading, or file has been saved incorrectly because of disk error;
- fixes in proxy authentication;
- fixed bug with caching of files received by HTTPS protocol (actually documents received by HTTPS won't be cached, but all cache flags now ignored in case of work via secure protocol).

Fixes in **acFormHelp** and **acInfoBox**: long strings now wraps to next line after ".", ",", ":", ";", "/" and "-" characters.

The *UseF1* property has been returned back again to **acFormHelp** component. (Allows to disable invoking the context sensitive help from mouse pointer on pressing F1 button when False).

13.12.2004 **3.6**

Support of Win32 platform of Delphi 2005.

Big improvements and changes in **acHTTP** component:

- Added *CacheOptions* and *InternetOptions* properties, 2 sets of various options used to specify various behavours of HTTP requests and how the component should work with standard cache;
- Removed *UseCache* and *UseCookies* properties, since they replaced by much more flexible *CacheOptions* and *InternetOptions* properties;
- Added *ProxyUsername* and *ProxyPassword* properties used to specify the required login information for secure proxy authentication *before* the HTTP request;
- Added *Pause* and *Resume* methods, used to terminate and resume downloading;
- Added 2 optional parameters to *Abort* method, which can specify whether to delete the file which are being downloaded and whether to terminate running thread imediately, without releasing internet handles;
- Modified *HTTPHeaderInfo* event: added *ETag* parameter.

Some fixes which solves the problems with compiling the AppControls in C++ Builder without modifications of HPP files.

Added *OnTemplateChanged* and *OnParamChanged* events to **acTextTemplateConverter** component. The **acHTMLViewer** now automatically refreshes the view accordingly to changed template.

18.11.2004 **3.5.3**

Fixed bug in **acInfoBox** component, now the text correctly displays in the message boxes with disabled "Don't show this message again" check box, so the **acAutoUpgrader** now correctly displays the "upgrade prompt" when the newer version of application are available.

10.11.2004 **3.5.2**

Added new component: **acComboDatePicker** (panel with group of 3 combo-boxes used to quickly select the date (day, month and year).

© 1998-2006, UtilMind Solutions®

Added *lpAboveCenter* and *lpBelowCenter* parameters to **acAttachedLabel** class, so the components with attached label (acEdit, acNumberEdit, acLabeledComboBox, acMRUComboBox, etc) can have attached label centered below or above the control.

The **acSystemInfo** now correctly detects Win2003 server OS.

Fixed bug in **acIPEdit** which has allowed to enter values over 255 to the cells of IP edit.

Fixed another bug in **acThread** class, the thread has never returned to the "ready" state after execution.

08.11.2004 **3.5.1**

Fixed bug in **acInfoBox** component, now the text correctly displays in the message boxes with disabled "Don't show this message again" check box.

More fixes in **acThread** class.

04.11.2004 **3.5**

In general — tons of improvements, several new components, and support of **ThemeEngine** by http://www.ksdev.com (currently customers must have source codes of AppControls to recompile the visual controls so they will support themes of ThemeEngine).

New components:

acAccurateTimer	timer based in separate thread, much more accurate than standard system timer).
acCronJob	the alarm timer schedule. Produces periodical OnAlert events, which occurs in periods specified in CRON format of Unix)
acLinksPanel	panel with labels and images at the right. You can specify the font and image for each label and hook the event when user clicks the particular label.

Fixes and changes:

- Added *AutoCheckDelay* property to **acAutoUpgrader** component, plus many minor fixes (i.e: faster application restart with minimum CPU load, validation of downloaded files before replacing previous etc).
- Added tdNone value to *TopLevelDir* property of **acFileStorage** component. Now it can extract the files to ANY directory specified in *Subdirectory* property (not only to standard folders as was in previous versions).
- Added *CustomFields* property to **acIESnifferAutoFillUserInfo** component, which allows to specify custom fieilds/tokens for auto-filling of the web forms.
- Added *AddPath* method, *ShowTreePathInEdit* and *TreePathSeparator* properties to **acTreeComboBox** component (allows to display the full path to node in the edit/selection box) + fixed the bug which has allowed to enter text to non-editable combobox.
- Added *HeaderFont* and *ItemPopupMenu* property to **acListView** component (*ItemPopupMenu* allows to specify the PopupMenu for list items (different of PopupMenu for entire ListView control)), plus added *CopySelectionToClipboard* method (copies the text from selected rows/columns to clipboard + it automatically copies the text when user press Ctrl+C or Ctrl+Ins keys).
- Added *ItemPopupMenu*, *AlwaysExpandRoot*, *AlwaysFullExpand* and *BoldItem* properties to

© 1998-2006, UtilMind Solutions®

acTreeView component (*BoldItem* allows to specify the "bold" node, which can have its own text style, color, frame, and image at the right of text).

- Added *SendMail* property set, *GetLastRecords* and *EmailLastRecords* methods to **acThreadSafeLog** component (now it able to send email message with last *N* log entires to administrator in case of some fatal error).
- Fixed some issues with **acAppBar** component, which occured when the *BorderStyle* property was bsNone (borderless skinned forms).
- The **acFormHelp** component now invokes the content sensitivity popup help for disabled controls, plus fixed the problem with displaying "What's This?" menu item for controls with non-standard popup menu.
- Modified **acAwayTimer** component: the *AwayTime* property now have Cardinal type and returns value of milliseconds passed since last user activity, instead of TDateTime value.
- Modified **acButtonsPanel** component: *CustomBeforeOK* property changed to *CustomButtonPosition*. Also added "Apply" parameter to *OnApplySettings* event (allows to validate the submission of dialog box on "applying" modified content).
- Fixed memory/resource leak in **acThread** component (the handle of suspended thread was not destroyed on destroying the TacThread class).
- Fixed cosmetic bug in **acSplitterPanel** component, now it correctly specifies the *BorderWidth* property.
- The drop-down tree view of the **acTreeComboBox** component now displays the drop-down shadow under it + supports the mouse wheel to scroll the tree items.
- Many others minor improvements and fixes, mostly to make the AppControls compatible with **ThemeEngine**…

14.01.2004 **3.4.2**

Added new feature to **acThreadSafeLog** component, which allows to limit the maximum size of the log file (see new *MaxSize* property, *OnMaxSizeReached* event and *ClearLog* method).

11.01.2004 **3.4.1**

Few minor bugfixes in **acFormMagnet** component (to make it working propertly if the parent of MDI children was changed at run-time).
Changed the name of icon resource of **acAutoUpgrader** component, therefore the default application icon set in project options would remain.

03.01.2004 **3.4**

Added new component, **acThreadSafeLog**, the log-writter which can be safely used from different threads simultaneously without additional syncronization.

The **acSendMail** component now supports two threading models and can be used to send e-mails from different threads simultaneously, without creating new separate thread (see new *Threaded* property and description on threaded and non-threaded models).

Fixed critical problem with resources of **acAutoUpgrader** component under Windows 98/ME.

Some fixes in the **acAppBar** component, now it does not "reserve desktop space" for hidden forms.

© 1998-2006, UtilMind Solutions®

Minor fixes in the *OnChange* event handler of **acNumberEdit** component (now it not occurs when the component is loading).

The type of *IconResID* property of the **acInfoBox** component has been changed from Integer to String (to allow to specify text-based resource identifiers).

16.12.2003 **3.3.1**

Fixed bug in **acHTTP** component. Now it allows to upload files to password protected directories and allows to specify the login information in *OnPasswordRequest* event handler if it's required. (Previous version released the uploading stream after sending the data, so the upload request could not be automatically retried. However, it still advisable to have correct Username/Password before uploading data to password protected directories, to avoid unnecessary double-uploading).

12.12.2003 **3.3**

Fixed some bugs in **acAppBar** component which can lead to strange behaviors when the form is undocking, if application uses *more than one* AppBar.

Changed the parameters of *OnWBStatusTextChange* event handler of **acIESniffer** component. Now it allows to change the status text which appears in browser window.

Fixed bug in **acCaptionButton**, which has displayed system menu on click of caption button on minimized MDI form.

Fixed bug in **acCheckGroupBox**, now it don't cuts the caption of checkbox when big fonts used on the form.

Added *Alignment* property to **acButtonsPanel** component which can arrange the buttons by left or right side, or center of panel.

Added *ShowAccelChar* property to **acBitBtn** and **acButtonsPanel** components. This property controls whether the buttons should display ampersand (&), or just underline the accelerator character after it.

Added Turkish language support to **acAutoUpgrader** component.

Many minor bugfixes and improvements in other components (acOnlyOne, acBitBtn, acInfoBox etc).

25.11.2003 **3.2**

Added **acTextTemplateConverter** component. Basically it can be used as plug-in for **acSendMail** component to replace the %KEYWORDS% in the message body and headers to the strings specified in the TemplateConverter.

Added *MsgDateTime* property to **acSendMail** component (allows to specify the date and time of outgoing message).

Added *Bottommost* property to **acFormHook** component (constantly keeps the form under other windows when True).

Improvements/bugfixes in **acStringListContainer** component:
Now it correctly modifies the Names/Values between the separator + the separator can contain more than one character.
Added *IgnoreCase* parameter to *IndexOfName* and *IndexOfValue* methods which allows to find

© 1998-2006, UtilMind Solutions®

the index of string in the list without case sesnsitivety.

The **acAppBar** component now in correctly displays the borderless application desktop toolbars.

Minor cosmetic bugfixes in **acLabel** component.

20.11.2003 **3.1.1**

Fixed critical bug which crashed IDE at design-time.

19.11.2003 **3.1**

Two new components: **acIESnifferAutoFillUserInfo** (plug-in for **acIESniffer** to automatically fill the forms with some basic user details) and **acApartPopupMenu** (popup menu which correctly works on ActiveX forms in Apartment thread model).

Many improvements in **acIESniffer** component: now it can automatically fill the Web forms (with **acIESnifferAutoFillUserInfo** plug-in), redirect user to the custom search engine instead of default MSN, allows to specify the custom URL for search bar (at the left side of IE window), specify zoom factor for viewed pages and some other features.

Improvements in **acTreeComboBox** component:

Now it can automatically complete words that user typed in the input box (added *AutoComplete*, *AutoCompleteDelay* and *AutoCompleteIgnoreCase* properties);

Added *SelectNodeByText*, *FindNode* and *FindNodeByAbsoluteIndex* methods.

Improvements/changes in **acOnlyOne** component:

The *SwitchToPrevious* property (used to activate the first program instance) renamed to *SwitchToFirst*;

Added *CloseFirst* property (can be used to automatically close first program instance);

Added new method, *CloseFirstInstance* which allows to close previous program instance + described an example how to close the application built with **acOnlyOne** from external program and how switch to previous application instance before any form will be created (see it in the manual on acOnlyOne);

Also we have included to the distribution package new command-line utility, named CloseApp, which can be used in setup programs to close the application before the setup will overwrite the executable file (comes with source code, see it in the Tools directory of distribution package).

Minor improvements in other components:

Fixed problem with displaying the dialog boxes over the **acAppBar** when it's *AlwaysOnTop* property is True.

Added *SetSelection* method to **acEdit**, **acNumberEdit**, **acLabeledComboBox** and **acTreeComboBox** components;

Added *OnSubmit* method and *SubmitButton* property to **acMRUComboBox** component;

The *HideOnRestore* option of **acTrayIcon** component has been renamed to *HideIconOnRestore* + added new option *MinimizeToTrayOnDeactivate*. The *PlaySysActionSounds* property also become an option (see *Options* property);

Few fixes in the **acInfoBox** component (now it allows to copy the content of the message box by Ctrl+C or Ctrl+Ins keys), plus now it doesn't allow to display several InfoBox'es simultaneously, one over previous, plus now it brings the activated InfoBox to the foreground (useful when displayed from minimized app);

© 1998-2006, UtilMind Solutions®

Added *saveOnExit* and *saveOnDestroy* parameters to the *Options* property of *RegistrySaver* structure of **acEdit** component and all its descendants. The options now allows to specify the events which could automatically save the current text in the edit box to registry.

22.10.2003 **3.0.1**

Fixed installation package for C++ Builder 5/6.

Fixed bug in **acExtAssociation** component (occasionally it didn't allowed to specify the file extension).

Fixed bug in **acHTTP** component. Now it can download non-binary ASCII content into the file specified in *OutputFileName* property.

The **acTrayIcon** component now refreshes its icon in system tray area in 15 seconds, if the application was started automatically on Windows startup (this improvement was imlementeted due to strange behavior or bug in WinXP, where some tray icons may appear invisible if the program was executed on startup).

Added *ClearMRU* method to **acMRUComboBox** component.

21.09.2003 **3.0**

Third version of AppControls. Big redesign and optimization of almost all internal structures and classes + **9** new components: **acClipboard**, **acPrcessList**, **acConnectionList**, **acIESniffer**, **acButtonsPanel**, **acTreeComboBox**, **acWebImage**, **acStringListContainer** and **acHostnameResolver**.

Improvements in various existing components (see the description of new properties and methods in the manuals for each component):

Added *IsMinimized* property to **acTrayIcon** component (allows to determinate whether application is minimized to system tray);
Added *WaitTimeout* property to **acHTTP**, **acSendMail**, **acThread** and **acSQLQueryThread** components (allows to specify maximum timeout needed to complete operation or die);
Added *OnActiveTimer* property to **acAwayTimer** component (the timer-based event which only occurs when user actively using the PC);
Added *InstallTime* property to **acAppStartCounter** component (writes and retrieves a timestamp when the application was first executed);
Added *MinimizeIfNoInfoFound* property to **acFormPlacementSaver** component (when True, it allows to maximize the application window on its first start (if no placement information was found in registry));
Added *FindNode*, *FindNodeByAbsoluteIndex* and *FindAbsoluteIndexByNode* methods to **acTreeView** component;
Added *MaxWidth* property to **acFormHints** component. The new property allows to wrap longs strings in balloon-style hints in Delphi3/BCB3 and higher.
Changes in *OnHeaderInfo* event of **acHTTP** component (added ErrorCode parameter which passes the status code of the HTTP request);
The **acSendMail** component now encodes attachment in ~500 times faster.
All components which works with registry now retrieving values from registry for users which don't have administration rights on NT-family machines.

Improvements/changes in **acAutoUpgrader** component:

Added *CheckUpdateAndWaitResult* and *RestartApplication* methods;
Added *UseCache* property;

© 1998-2006, UtilMind Solutions®

Changed *OnEndUpgrade* event handler;
The built-in application update wizard now supports *Icelanding* and *Polish* languages.

Improvements in **acListView** component:
Added *CheckOnClick* property (specifies whether the list items should be checked/unchecked on click);
Added *OnCheckBoxClick* event (occurs when user clicks the checkbox at the left of item);
Also now the acListView can show special label at the center of report-style view, which only visible if the list is empty (see *EmptyListMessage* and *ShowEmptyListMessage* properties).

Many other improvements and lots of new functions in acUtils unit (see description on "Bonus Routines" in manuals).

06.04.2003 **2.7**

Fixed critical bug of previous version, which did not allowed to terminate the debug process in Delphi/BCB IDE (on Ctrl+F2).

Added *HideFileLocation* property to *Wizard* structure of **acAutoUpgrader** component (now it allows the location from where the file are downloading)

Added *OnFormCloseButtonClick* event to **acFormHook** component (it notifies when user clicks the close [X] button on form's title bar).

Fixed small bug in **acOnlyOne** component, now it correctly switches to previous application instance.

Fixed small bug in **acNumberEdit** component (it did not reacted to Enter and Escape key presses, which is very important in dialogs).

Improvements in **acListView** component:
The *OnColumnResize* event was renamed to *OnColumnResizeEnd*, plus was added *OnColumnResizeBegin* event which occurs when user just about to resize the list column and allows to disallow the resizing.
Added *SetColumnOrders* method (can set the orders of column or copy the orders from another ListView).
The *BestFit* method now can specify best fit by widths of text on the header, as well as by widths of list items.
Added *ExportToTXT* and *ExportToHTML* methods.

Added **acHostnameResolver** component (used to lookup for the hostname by specified IP or list of IP addresses).

15.03.2003 **2.6**

Improvements in **acHTTP** component:
Added *Timeouts* structure (helps to specify timeouts for HTTP requests) and *OnRedirected* event (occurs if the server has redirected the HTTP query to another location).

Improvements in **acAppBar** component:
Added *DontSlideOnTaskBarWhenAutoHide* property (when True, the windows taskbar is not obscured when the appbar slides from screen's side) + fixed some problems occured in Windows XP (flickering when the floating AppBar moved after "undocking").

Improvements in **acTrayIcon** component:

© 1998-2006, UtilMind Solutions®

Now it able to restore the icon in the system tray after crash or restart of Windows Explorer. Also added *Icon16* property, which can be used together with Icon property and allows to separately specify high color tray icon (for Windows ME and XP) and 16-colors icon (for Win95/98/NT4/2000, unfortunately older version can only show 16-color icons in the tray).
Also added *WhatsThisHint* and *PopupMenuWhatsThisItem* properties, which can be used if you wish to show "What's This?" popup help window when user clicks proper menu item in popup menu associated with tray icon.
Also added some *OnBaloonXXX* events (can be used to respond on events of balloon-style tooltip of tray icons of Win2000 and higher), and *HideInfoBalloon* method (used to hide the balloon-style tooltip without waiting when it timeout will be expired).

Improvements in **acListView**:
Now it better supports Windows XP themes. Added HeaderColor property which allows to specify the color for column headers for uniform look. Added FindCaption method which can be used to quickly find the list item by caption.

Fixed bug in **acFormPlacementSaver** component, in previous versions it did not saved the form placement if it was closed after call of "Form.Close" method.

Added *OnFormShown* event to **acFormHook** component (occurs at once *after* displaying the form, and can be used instead of OnShow form's event which occurs *before* it actually displays).

All visual controls, like acBitBtn, acCheckBox, acRadioButton, acCheckGroupBox, acListView, acPanel and all its descendants (HeaderPanel, PicturePanel) supports the Windows XP themes in lower Delphi/BCB versions (lower than Delphi 7, which already comes with built-in WinXP Theme API). However, our controls can correctly show Windows XP styles even if they was built with Delphi 2 or BCB 3.

Many other minor bugfixes (i.e: fixed *GetCPUSpeed* function which raised 'division by zero' on P4 3.06Ghz) and improvements.

30.10.2002 **2.5.3**

Improved all controls with *AttachedLabel*. Now the attached labels will be look correctly with any pixel per inch resolution.

Added *ShowHeaderArrow* property to **acHeaderPanel** component. Now the component can show the arrow near the caption, displaying that the caption of header panel have a popup menu.

Some cosmetic bugfixes in **acSplitterPanel** component (now it correctly restores the placement of the panes from the registry, even if the form has been resized on program startup).

Added *FlickerFreeMaximize* property to **acFormPlacementSaver** component. (When it is set to True and the component is about to restore the previous placement of your form (maximize the form) on program startup, the component will do it without flickering).

Added *OnCaptionDblClick* event to **acCaptionButton** and **acTrayIcon** components. The event occurs when user double clicks the title bar of the form.

Fixed bug in **acFormPlacementSaver** component, in previous versions it did not saved the form placement if it was closed after call of "Form.Close" method.

22.10.2002 **2.5.2**

Release mostly with improvements in acHTTP and acAutoUpgrader components:

© 1998-2006, UtilMind Solutions®

Added the *OnHeaderInfo* and *OnDoneInterrupted* events to **acHTTP** component. The *OnHeaderInfo* allows to receive the headers of the remote document and decide whether you want to continue the download the content.

Added *MiscFilesUpgrade* property and *OnFileStart* event to **acAutoUpgrader** component. These improvements allows to receive the headers of response from HTTP server before downloading the data and decide whether you want to download and upgrade the files which sizes remains unchanged since previous release of your application.
Also added *OnDoOwnCloseAppMethod* event, which allows to execute own method which will close the application before the application upgrade.

Bugfixes:
Fixed minor bug in **acAwayTimer** component (component for detecting the user's activity/inactivity).
The **acWin2kEffects** component now correctly refresh the form content after displaying the effect described in *AnimateOnShow* structure.

Minor improvements:
The **acSystemInfo** component now detects and allows to specify the screen resolutions of Pivot monitors: 480x640, 600x800, 768x1024, 864x1152, 960x1280, 1024x1280, 1200x1600, 1080x1920, 1200x1920.

15.10.2002 **2.5.1**

Improved the threading model for all components which uses acCustomThread class (**acHTTP**, **acAutoUpgrader**, **acSendMail** and **acSQLQueryThread**). New improvements allows to use these components in ActiveX forms even in Delphi 6/7, C++ Builder 6 and higher.

Added Spanish and Simplified Chinese languages to built-in *"Application Update Wizard"* of **acAutoUpgrader** component. Now it supports 13 natural languages.

Fixed minor bug in **acWin2kEffects** component, which did not displayed the animation effect correctly on closing of the child forms.

Added *BiDiMode* property for *AttachedLabel* structure used in all our Edit and ComboBox controls.

21.09.2002 **2.5**

Support of Delphi 7.

Added *OnProxyAuthenticationRequest* event to **acHTTP** and **acAutoUpgrader** components. This event should be used to prompt user for username and password required to access the Web via secure proxy server which requires authentication.

Added **acAwayTimer** component.

Big redesign and optimization in the internal template classes for all components of the AppControls pack.

2.4.8

Support of C++ Builder 6.
Added 3 new languages to the built-in *"Application Update Wizard"* of **acAutoUpgrader** component: Estonian, Danish and Ukrainian.

© 1998-2006, UtilMind Solutions®

2.4.5
Added *AppPriorityClass* property to **acAppEvents** component.

Fixed problem with *Alignment* property in **acRichLabel** component.

Added *State* property to **acCheckGroupBox** (allows to make the checkbox "grayed").

A lot of bugfixes and improvements, especially 4 components added in previous release (i.e: added possiblity to specify custom background for that panels or draw the background programmatically using *OnPaint* event).

18.04.2002 **2.4.4**
Added 4 new components: **acSplitterPanel** (panel which splits itself to two proportional areas, like seen at Outlook Express), **acPicturePanel** (panel with caption, icon and some text, similar to panels often used in Microsoft's dialog boxes), **acHeaderPanel** (panel with title bar and buttons on it. Similar controls used in MS-Outlook and Explorer of Windows XP) and **acHeader** (header bar with ability to specify custom background and buttons. This control also built-in to acHeaderPanel to show the title bar).

24.02.2002 **2.4.3**
The **acFormHelp** component now works with toolbars and menus by Developer Express.

Added *AttachedLabel* structure to all edit and combobox controls (**acEdit**, **acNumberEdit**, **acIPEdit**, **acLabeledComboBox** (this is new component), **acMRUComboBox**, **acImagesComboBox**).

Added Portuguese and Traditional Chinese languages to the **acAutoUpgrader** component.

08.01.2002 **2.4.2**
Added *FlashBackground* method to **acWin2kEffects** component (to allow the form flash in background mode until user click the window with mouse or active at with keybard).

Added *MaxClientWidth*, *MaxClientHeight*, *MinClientWidth* and *MinClientHeight* properties to **acFormSizeRestrictions** component. These properties allows to specify the resize boundaries for the form.

17.12.2001 **2.4.1**
Added new event to **acListView** component (*OnColumnResize*).
Improvements in **acSystemTCPClient**, **acSimpleTCPServer** and **acThread**.

02.12.2001 **2.4**
Added 4 new components: **acEmbeddedForm**, **acImagesComboBox**, **acSimpleTCPServer**, **acSimpleTCPClient**.

Improvements in the **acAutoUpgrader**:
1. Improved integration with external setup programs (in case if the upgrade made by external program, the self-upgrading module now will wait for termination of setup);
2. Corrected function that calculates the estimated time left in the download progress (this also fixed in OnProgress event of **acHTTP** component. (New formula: X := FileSize / BytesRead *

© 1998-2006, UtilMind Solutions®

ElapsedTime - ElapsedTime);
3. Added Italian language support for *"Application Update Wizard"*.

Added couple of new optional properties (*Margins* structure) to the **acFormHelp** + it now completely supports Jordan Russel's Toolbar2000 buttons.

Improvements in **acFormHints** component. Now you can select the style of hint dynamically in the *OnShowHint* event.

Bugfixes in download mechanism of **acHTTP**/**acAutoUpgrader** components (fixed problems with allocation the memory for streams for big files (>4MB)).

Added forgotten in previous release the component editor for *Columns* property of the **acDBListView**.

New property in **acOnlyOne**, new event in **acFormPlacementSaver**, huge improvements in **acListView** and **acFileStorage** components. A lot of others minor improvements and optimizations.

31.10.2001 **2.3.6**
Added **acDBListView** component — db-aware version of acListView.

The **acFormHelp** component now supports any sophistical buttons and menu items (Toolbar2000 items for instance) as controls which can automatically invoke the help pointer.

Fixed minor bug in **acAppBar** component, discovered in Win2000 (Win95/98/NT4 works ok), when user tryed to undock the form double-clicking it to restore it to "floating" state.

Improved **acAutoUpgrader** component: now it correctly supports external setup-files, added new option to *UpgradeMethod* property. Minor improvements in the self-upgrading mechanism.

24.10.2001 **2.3.5**
Implemented couple of new features for tooltips of the **acListView** component. First — added possibility to specify custom multiline tooltips for list items. Second — added new property *ShowScrollTips*. When it True and user drags the thumb of vertical scrollbar, the list-view shows the hint with caption of top list item of current sorted column.

Fixed bug in **acEdit** component. (Values of *PasswordChar* and *ValidateChars* properties was ignored.)

Added Dutch localization resources to **acAutoUpgrader** component. Thanks to Carl Peeraer for translation.

18.10.2001 **2.3.4**
Added **acSendMail** component, extremely easy to use implementation of SMTP-client.

Implemented some new properties and events for **acAutoUpgrader** component, which allows to use external installation programs for local extraction downloaded setup-file in "silent mode", fixed a couple of spelling mistakes in localization of Application Update Wizard.

Fixed bug in **acAppBar** component. (There was strange behavior on docking the AppBar to the right or bottom sides of screen, with pre-configured minimal size of form.) Also added new property

© 1998-2006, UtilMind Solutions®

(*TitleBar*) which allows to show the title bar on form docked to the screen edge.

Fixed major design-time bug in **acFormSystemMenu** component. The Delphi IDE has crashed on removing the component from form at design-time.

Added *Range* property to the **acHTTP** component, which allows to download certain blocks of data from remote binary files or implement custom mechanism for resuming of broken downloads (if you don't use IE cache).

Added *MIMEType* property to **acExtAssociation** component. The new property allows to specify and determinate the media type of file extension.

24.09.2001 **2.3.3**
Minor bug fixed in self-upgrading mechanism of **acAutoUpgrader** component (*Upgrader3.pas*).

23.09.2001 **2.3.2**
Improved **acDesktopWallpaper** component. Added *AutoFormat* property.
Minor bugfixes in **acAutoUpgrader** component (in Wizard form and self-upgrading module).

19.09.2001 **2.3.1**
Added **acUnixCrypt** component. Fixed some minor bugs in the **acAutoUpgrader**, added German language support.

10.09.2001 **2.3**
acAutoUpgrader Pro released! Pro-version contains built-in "Application Update Wizard", a lots of customization features, proxy support, multi-language support and much more! See press release to check out what's new in the **acAutoUpgrader**.

Added *OnPasswordRequest* event to **acHTTP** component. This event allows user to enter his username and password, required for access to password protected Web directory, then retry HTTP request.

Bufixes and improvements in the **acFormHelp** component:
1.Fixed problem with appearance of context-sensitive help windows without shadow;
2.Added *ParentFont* property which helps to make uniform appearance for context-sensitive help as well as for usual controls;
3.Added *UseF1* property, which can allow or disallow displaying of the context-sensitive help windows when user press F1 key;
4."What's This?" menu item does not pops up anymore if the control already have own popup menus (even non-standard, like Jordan's Russels Toolbar2000 menus).

Some minor bugfixes in other components…

27.08.2001 **2.2.2**
Added *AddHeaders* property to **acHTTP** component. This property allows to specify additional headers for the HTTP request.

03.08.2001 **2.2.1**
Minor bugfixes in **acCaptionButton**, **acAppBar** and **acFormSystemMenu** components.

© 1998-2006, UtilMind Solutions®

Added new component, **acDragPanel**. DragPanel is great addition to **acAppBar** to allow user to move and dock the form dragging the panel that covers form.

15.07.2001 **2.2**

A lots of improvements, Delphi 6 support, and two new components: **acFormSystemMenu** and **acFormStatusControl**.

Changes and improvements:
acFormCursors component has been completely rewritten and renamed to the **acAppCursors**. Now just one acAppCursors component can change appearance of cursor images for non-clients objects in *entire* application, in any form, even non-VCL based (i.e: Save/Open file dialogs).

acGradientForm component renamed to **acFormBackground**. Improved component can paint the form background using 3 various gradient effects, or 4 styles of drawing the wallpaper: tiled, stretched, tiled with zigzag (like bricks in the wall), or aligning the picture by center.

Added some new properties which allows the **acFormHelp** to put "What's This?" menu item to the main and/or system menu(s) of your form. Additionally you can point the "What's This?" button on the toolbar (or any control, in general). When user clicks the button or menu item, the cursor will be changed to a question mark with a pointer. If the user then clicks a control in the form, the control receives a message to show the context-sensitive help, taken from secondary part of the control's Hint property. Also added *ShowHelpPointer* method which allows to invoke the help-pointer "manually".

Minor bugfixes and optimizations in the **acCaptionButton** component. The caption buttons now refreshes much faster when the form's caption changes + fixed small bug in buttons wich have own popup menu.

Added *OnCaptionChanged* event to **acFormHook** component + minor bugfix in the *NoCaption* property.

Added *SortMarkJustify* property to the **acListView** component. The arrow-style sort mark now can be displayed either near the text or near the edge of header section.

Added *oDblClickRestore* option to the **acTrayIcon** component. This option allows to prevent restoring of the form associated with the tray icon on double click.

Minor bugfix in the **acCheckGroupBox** component.

Changes and improvements in the core structures (TacComponent and TacFormHookComponent).

Added **6 new cursors** to the IDE (crArrowResize, cr3dArrowResize, cr3dArrowWarning, crPen, crZoomIn and crZoomOut).

21.05.2001 **2.1**

Fixed bug in the **acTrayIcon** component, which was prevented the Win2000 from shutting down (or logging off) when the tray icon was visible in the taskbar status area.

19.05.2001 **2.0**

Second version of the AppControls. The package now contains new section with improved visual controls.

Were added **15** new components: **acHTTP**, **acLabel**, **acURLLabel**, **acRichLabel**, **acCheckBox**,

© 1998-2006, UtilMind Solutions®

acRadioButton, **acCheckGroupBox**, **acButton**, **acBitBtn**, **acEdit**, **acNumberEdit**, **acIPEdit**, **acMRUComboBox**, **acTreeView**, **acListView**.

Fixed small bug in the **acFormHints** component. In previous releases the balloon-style hint could be displayed semi-transparent (even if Transparent property is False), when user moves the mouse too quickly and hint did not have time to disappear.

Added some new properties and methods the the **acAppBar** component.

04.05.2001 **1.3**

Added **acFormResizeGrip** component which attaches the SizeGrip rectangle to the bottom-right corner of the form at run-time. This allows the user to resize the form dragging this grip.

Improved **acAppBar**. Added several new properties and methods for more flexible management of the AppBar window.

27.04.2001 **1.2**

Added **acExtAssiciation**, **acShortcut** and **acDateTimeCalc** components.

Updated the **acFormHelp** component. The text of the context-sensitive help now can be adjusted within a popup window. Fixed problem with triggering of the OnHide event by OnShow.

Improved the **acAppBar** component. Fixed small bug with auto-hiding the AppBar on program startup and added two new events OnHiding/OnUnhiding. Fixed minor problem with "unhiding" of the AppBar when it need to accept some data by drag'n'drop. Also updated component able to store its settings either to the system registry or INI files.

06.04.2001 **1.1d**

Improved the **FormHelp-designer**, to override ALL others installed property editors for *Hint* property, and to let you to edit the context-sensitive help in WYSIWYG mode.

Added several new useful routines to **acUtils** unit.

04.04.2001 **1.1**

Improved the **acFormHelp** component. New version of the FormHelp can apply the "What's This?" menu item to every control with context-sensitive help in the secondary part of hint, and invokes the context help either after pressing the "Help" button on the title bar or after selecting "What's This?" menu item.

Updated **acAnimationEffect** component. The smooth effect now based on system-wide events istead of using the *OnShow* and *OnHide* form events.

Improved the **acAutoUpgrader** component. Updated version can upgrade the applications even if it used by many users at the same time (i.e.: users start the program from network drive) and the executable file is locked.

31.03.2001 **1.0.4**

Fixed some bugs in the **CaptionButton** class and all its successors (acCaptionButton, acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost). New version more correctly displays the caption buttons if the form don't have a regular buttons (when Form.BorderStyle=bsNone). Fixed some problems with appearing of the tooltips (hints) when the form contains main menu. Custom cursors for the caption buttons is flickersfree now.

© 1998-2006, UtilMind Solutions®

Updated **acWavPlayer** component (some minor optimizations).

28.03.2001 **1.0.3**

Added new component, **acWin2kEffects**, which enables to produce some special effects of the Windows 98 / 2000. You can show roll, slide and alpha-blending animation effects on showing and hiding windows, make your forms *translucent*, flash the caption bars and taskbar icons.

Improved the **acTrayIcon** component. Now it supports the cartoon-style balloon tooltips — a nonintrusive way to display an explanatory message for a status bar icon, implementation a balloon tooltip with its stem pointing to the icon. The ToolTip will disappear when clicked, but you can also specify a time-out value. The information balloon can appears after minimizing of the forms to the system tray or after calling the *ShowInfoBalloon* method. New feature will works on ***Windows 2000 and later*** only. You can call the information balloons in lower Windows versions, however nothing will happends.

Cosmetic improvements and bugfixes in **acOnlyOne** and **acAnimationEffect** components. Added 7 useful GDI routines to **acGraphics** unit.

20.03.2001 **1.0.2**

Fixed the **acFormMagnet** component. Now forms with acFormMagnet can attract and attach other forms more correctly, independently of taskbar placement.

Added 5 new events to the **acAppEvents** component. New events lets you to hook ALL keypresses and ALL mouse clicks/movements within entire application.

16.03.2001 **1.0.1**

Improved the **acFormRoller** component - added smooth animation effect on rolling the form + added property wich let programers specify custom "Roll up / Roll down" tooltips.

Fixed small bug in the **acAppAutoRun** component which incorrectly works with long an EXE filenames in previous release.

18.02.2001 **1.0**

First official release. Distributed from AppControls.com (http://www.appcontrols.com) as part of UtilMind Solutions® Network. A lot of bugfixes and improvements of the beta version. **acSystemInfo** component adapted for Win2000.

24.01.2001 **1.0b**

Initial Beta release. Currently for beta-testers only. Package contains set of 27 components:

- acAppAutoRun *(updated since release for internal usage)*
- acAppStartCounter *(new)*
- acAutoUpgrader *(updated)*
- acAppEvents *(new)*
- acOnlyOne *(updated)*
- acTrayIcon *(updated)*
- acAppBar *(new)*
- acCaptionButton *(updated)*
- acGradientForm *(new)*
- acAnimationEffect *(updated)*
- acFormCursors *(new)*

© 1998-2006, UtilMind Solutions®

- acFormHelp *(updated)*
- acFormHints *(new)*
- acFormMagnet *(updated)*
- acFormTopmost *(new)*
- acFormRoller *(new)*
- acFormPlacementSaver *(updated)*
- acFormSizeRestrictions *(new)*
- acFileStorage *(updated)*
- acWavPlayer *(new)*
- acCursorLocker *(new)*
- acFormHook *(new)*
- acThread *(new)*
- acSystemInfo *(new)*
- acDesktopWallpaper *(updated)*
- acControlKeyState *(updated)*
- acQuickAboutBox *(new)*

All components, even that that was distributed by UtilMind before, has been completely redesigned, many properties, methods and event handlers has changed, so our old customers may found some incompatibility with previous versions of such components like AutoRunner, AutoUpgrader, Magnet and FormHelp.

Added some additional colors and cursors which installs with the package to the Delphi/BCB IDE.

Advanced Application Controls (http://www.appcontrols.com)
Copyright © 1998-2006, Utilmind Solutions. All Rights Reserved.
Documentation created with **Help&Manual**, the best help authoring tool.

5 Installation instructions

Package without source code

to Delphi 2

1. Download "AppControlsD2.zip" file.
2. Create "..\Lib\AppControls" directory.
3. Unzip files and copy them to "..\Lib\AppControls".
4. Start Delphi 2 IDE.
5. Select "Component \ Install..." menu item.
6. Press "Add" button and select "_AACReg.pas" file.
7. Rebuild library.

to Delphi 3

1. Download "AppControlsD3.zip" file.
2. Create "..\Lib\AppControls" directory.
3. Unzip files and copy them to "..\Lib\AppControls".
4. Start Delphi 3 IDE.
5. Open "AppControlsD3.dpk" file.
6. Install package to the components palette ("Install" button).

to Delphi 4

1. Download "AppControlsD4.zip" file.
2. Create "..\Lib\AppControls" directory.
3. Unzip files and copy them to "..\Lib\AppControls".
4. Start Delphi 4 IDE.

5. Open "AppControlsD4.dpk" file.
6. Install package to the components palette ("Install" button).

to Delphi 5
1. Download "AppControlsD5.zip" file.
2. Create "..\Lib\AppControls" directory.
3. Unzip files and copy them to "..\Lib\AppControls".
4. Start Delphi 5 IDE.
5. Open "AppControlsD5.dpk" file.
6. Install package to the components palette ("Install" button).

to Delphi 6
1. Download "AppControlsD6.zip" file.
2. Create "..\Lib\AppControls" directory.
3. Unzip files and copy them to "..\Lib\AppControls".
4. Start Delphi 6 IDE.
5. Open "AppControlsD6.dpk" file.
6. Install package to the components palette ("Install" button).

to Delphi 7
1. Download "AppControlsD7.zip" file.
2. Create "..\Lib\AppControls" directory.
3. Unzip files and copy them to "..\Lib\AppControls".
4. Start Delphi 7 IDE.
5. Open "AppControlsD7.dpk" file.
6. Install package to the components palette ("Install" button).

to Delphi 2005
1. Download "AppControlsD2005.zip" file.
2. Create "..\Lib\AppControls" directory.
3. Unzip files and copy them to "..\Lib\AppControls".
4. Start Delphi 2005 IDE.
5. Open "AppControlsD2005.dpk" file.
6. Install package to the components palette (right-click on "AppControlsD2005.bpl" node in the Project Manager and select "Install" menu item).

to C++ Builder 3
1. Download "AppControlsCB3.zip" file.
2. Create "..\Lib\AppControls" directory.
3. Unzip files and copy them to "..\Lib\AppControls".
4. Start C++ Builder 3 IDE.
5. Open "AppControlsCB3.bpk" file.
6. Select "Project \ Make AppControlsCB3" menu item.
7. Select "Component \ InstallPackages" menu item.
8. Press "Add" button and select "AppControlsCB3.bpl" file.
9. When you will build a project with Internet components (i.e: acHTTP, acAutoUpgrader, acSendMail etc), linker may raise some errors (i.e: Unresolved external 'InternetOpenA', 'InternetCrackURLA' etc). To avoid this, please add INET.LIB file (in CBuilder\Lib directory) to your project. The INET.LIB contains references to functions of WinInet.dll (standard Internet library).

to C++ Builder 4
1. Download "AppControlsCB4.zip" file.
2. Create "..\Lib\AppControls" directory.
3. Unzip files and copy them to "..\Lib\AppControls".
4. Start C++ Builder 4 IDE.

© 1998-2006, UtilMind Solutions®

5. Open "AppControlsCB4.bpk" file.
6. Install package to the components palette ("Install" button).
7. When you will build a project with Internet components (i.e: acHTTP, acAutoUpgrader, acSendMail etc), linker may raise some errors (i.e: Unresolved external 'InternetOpenA', 'InternetCrackURLA' etc). To avoid this, please add INET.LIB file (in CBuilder\Lib directory) to your project. The INET.LIB contains references to functions of WinInet.dll (standard Internet library).

to C++ Builder 5

1. Download "AppControlsCB5.zip" file.
2. Create "..\Lib\AppControls" directory.
3. Unzip files and copy them to "..\Lib\AppControls".
4. Start C++ Builder 5 IDE.
5. Open "AppControlsCB5.bpk" file.
6. Install package to the components palette ("Install" button).
7. When you will build a project with Internet components (i.e: acHTTP, acAutoUpgrader, acSendMail etc), linker may raise some errors (i.e: Unresolved external 'InternetOpenA', 'InternetCrackURLA' etc). To avoid this, please add INET.LIB file (in CBuilder\Lib directory) to your project. The INET.LIB contains references to functions of WinInet.dll (standard Internet library).

to C++ Builder 6

1. Download "AppControlsCB6.zip" file.
2. Create "..\Lib\AppControls" directory.
3. Unzip files and copy them to "..\Lib\AppControls".
4. Start C++ Builder 6 IDE.
5. Open "AppControlsCB6.bpk" file.
6. Install package to the components palette ("Install" button).
7. When you will build a project with Internet components (i.e: acHTTP, acAutoUpgrader, acSendMail etc), linker may raise some errors (i.e: Unresolved external 'InternetOpenA', 'InternetCrackURLA' etc). To avoid this, please add WININET.LIB file (in CBuilder\Lib directory) to your project. The WININET.LIB contains references to functions of WinInet.dll (standard Internet library).

Source Code

1. Create "..\Lib\AppControls" directory.
2. Unzip files and copy them to "..\Lib\AppControls".
3. Start Delphi / C++ Builder IDE.
4. Select "File | Open" menu item and point to "AppControlsDx.dpk" or "AppControlsCBx.bpk" ('x' is the version of Delphi or BCB).
   Note: Delphi 2 users please select "Component | Install..." menu item, add "_AACReg.pas" file and rebuild the library.
5. Click "Compile" button in the package box to compile source files, then "Install" button to install it to the component palette.

Note for C++ Builder

When you are using the Internet components (i.e: acHTTP, acAutoUpgrader, acSendMail, acIPEdit etc, or just functions from acInet module), please don't forget to add INET.LIB to your project (it can be found at "CBuilder\Lib" directory). This file contains the references to routines from WinInet.dll. So if you got linker error such like following:

```
[Linker Error] Unresolved external 'InternetCrackUrlA' referenced from
C:\PROGRAM FILES\BORLAND\CBUILDER5\PROJECTS\LIB\APPCONTROLSCB5.LIB|acInet.pas
```

please don't worry and be aware that InternetCrackUrlA are used to parse the URL (split URL to domain name, port, document name etc). To solve this problem, just add INET.LIB to your project (use "Project | Add to project" menu item in C++ Builder IDE).

© 1998-2006, UtilMind Solutions®

Remark: In C++ Builder 6 and higher add WININET.LIB instead of INET.LIB.

Advanced Application Controls (http://www.appcontrols.com)
Copyright © 1998-2006, Utilmind Solutions. All Rights Reserved.
Documentation created with **Help&Manual**, the best help authoring tool.

6 Registration Information

Advanced Application Controls is SHAREWARE. This means that you can try it out for free, but if you like it and want to use it you have to register it with the author. Before continue read and accept license agreement please.

The only difference between the unregistered and registered versions is that the registered one has not message box with remind to register and works without Delphi (C++ Builder) running. You can also purchase the source code, if you would like to have it, and be able to compile or modify the AppControls on any 32bit version of Delphi or C++ Builder.

If you would like to use the AppControls and receive full, unrestricted version, priority support or even source code — you have to purchase proper license.

All prices are in United States Dollars (USD). Registering entitles you to unlimited support via E-Mail, minor version updates indefinitely and major version updates for 1 year from date of purchase. You can use registered components in any number of projects, there is no deployment and royaltee fees.

Registration types:

Full, unrestricted version without source code:
Single developer license:
- **https://secure.element5.com/register.html?productid=139996 - $ 149.95**

Site license:
- https://secure.element5.com/register.html?productid=139999 - $ 499.95

Full version including 100% Source Code:
Single developer license:
- **https://secure.element5.com/register.html?productid=140003 - $ 299.95**

Site license:
- https://secure.element5.com/register.html?productid=140007 - $ 899.95

Upgrade to source code *(only if you already have license)*
Single developer:
- https://secure.element5.com/register.html?productid=140010 - $ 120

Company/organisation (site license):
- https://secure.element5.com/register.html?productid=140011 - $ 410

Upgrade to source code *(only if you already have v2.x license)*
Single developer license without source code:
- https://secure.element5.com/register.html?productid=192473 - $ 30

Single developer license with source code
- https://secure.element5.com/register.html?productid=192474 - $ 45

Site license without source code:
- https://secure.element5.com/register.html?productid=192477 - $ 90

Site license with source code:

- https://secure.element5.com/register.html?productid=192478 - $ 135

Comments

1. ***Site license*** covers a single organisation in one location (building complex). If you buy a site license, you may use the software in unlimited number of your company's computers withing this area. Site license is very cost-effective if you have many computers (many software developers).
2. ***Educational site license*** — same as site license described above but for educational institutions (schools, colleges, universities etc).
3. ***World-wide license*** covers all computers of single organisation world-wide. If your company has many branches even with thousands of computers, world license covers them all. Please contact us if you would like purchase the world-wide license.

See license agreement for more details.

Advanced Application Controls (http://www.appcontrols.com)
Copyright © 1998-2006, Utilmind Solutions. All Rights Reserved.
Documentation created with **Help&Manual**, the best help authoring tool.

7 License Agreement

Copyright

The Advanced Application Controls (software) is Copyright © 1998-2005, by Utilmind Solutions® (Utilmind). All rights reserved.
The authors - Utilmind Solutions® and Aleksey Kuznetsov (founder of Utilmind), exclusively own all copyrights to the Advanced Application Controls (AppControls) and all other products distributed by Utilmind Solutions®.

Liability disclaimer

THIS SOFTWARE IS DISTRIBUTED "AS IS" AND WITHOUT WARRANTIES AS TO PERFORMANCE OF MERCHANTABILITY OR ANY OTHER WARRANTIES WHETHER EXPRESSED OR IMPLIED. YOU USE IT AT YOUR OWN RISK. THE AUTHOR WILL NOT BE LIABLE FOR DATA LOSS, DAMAGES, LOSS OF PROFITS OR ANY OTHER KIND OF LOSS WHILE USING OR MISUSING THIS SOFTWARE.

Restrictions

You may not attempt to reverse compile, modify, translate or disassemble the software in whole or in part. You may not remove or modify any copyright notice or the method by which it may be invoked.

Operating license

Unregistered version

You may distribute the unregistered version of software freely, provided that all files are included and remain unmodified and that no extra files have been added to the package. You may not ask any money for the distribution. You may use the unregistered version of software free of charge for testing purposes, but if you want to use it for other purposes than testing - you have to register it with the author.

Registered version (single user license)

Once you have registered, you will receive a personal registered copy via email and login information to access your personal area at AppControls.com. This copy may not be copied or lend. You have the non-exclusive right to use registered version of the software only by a single person,

on a single computer at a time. You may physically transfer the software from one computer to another, provided that the software is used only by a single person, on a single computer at a time. In group projects where multiple persons will use the software, you must purchase an individual license for each member of the group or purchase site license. Use over a "local area network" (within the same locale) is permitted provided that the software is used only by a single person, on a single computer at a time. Use over a "wide area network" (outside the same locale) is strictly prohibited under any and all circumstances.

Registered version (site/team license)
Once you have registered, you will receive a personal registered copy via email and login information to access your personal area at AppControls.com. This copy may not be copied or lend. You have the non-exclusive right to use and transfer registered version of software on any number of computers by your company or your team only in one location (building complex). If you purchase a site license, you may use the program in an unlimited number of your company's computers within this area.

Registered version (Educational site license)
Once you have registered, you will receive a personal registered copy via email and login information to access your personal area at AppControls.com. This copy may not be copied or lend. You have the non-exclusive right to use and transfer registered version of software on any number of computers by your educational organisation (school/college/university etc) in one location (building complex). If you buy a educational site license, you may use the program in an unlimited number of your edicational organisation's computers within this area.

Registered version (World-wide license)
Once you have registered, you will receive a personal registered copy via email and login information to access your personal area at AppControls.com. This copy may not be copied or lend. You have the non-exclusive right to use and transfer registered version of software on any number of computers by your company or your team world-wide. If your company has many branches even with thouthands of computers, world wide license covers them all.

Notes (clarification)
"Single-user license" means "single-developer license". "Site license" means that it can be used by any number of software developers within your company.
You can use purchased components in ANY number of your projects and deploy the "end-user" software to ANY number of your users/customers without any additional royalty fees. However you are not permitted to distribute the component itself (the source code or .dcu files of components).

Back-up and transfer
You may make one copy of the software solely for "back-up" purposes, as prescribed by international copyright laws. You must reproduce and include the copyright notice on the back-up copy.

Terms
This license is effective until terminated. You may terminate it by destroying the program, the documentation and copies thereof. This license will also terminate if you fail to comply with any terms or conditions of this agreement. You agree upon such termination to destroy all copies of the program and of the documentation, or return them to author.

Other rights and restrictions
All other rights and restrictions not specifically granted in this license are reserved by authors.

© 1998-2006, UtilMind Solutions®

Advanced Application Controls (http://www.appcontrols.com)
Copyright © 1998-2005, Utilmind Solutions. All Rights Reserved.
Documentation created with **Help&Manual**, the best help authoring tool.

8 Contact

If you have questions about our products or other concerns you may have, please email us to **info@appcontrols.com**. We will respond you as soon as possible (usually no more than 24 hours).

Please visit our Web site: **http://www.appcontrols.com** (updates, FAQs, user's forums)

Email is always quickest, but alternatively you can snail-mail us at:
UtilMind Solutions,
Teremkivska-7, k.34,
Kiev, 03187, UKRAINE

Advanced Application Controls (http://www.appcontrols.com)
Copyright © 1998-2005, Utilmind Solutions. All Rights Reserved.
Documentation created with **Help&Manual**, the best help authoring tool.

9 Components (Part I)

9.1 acAccurateTimer

9.1.1 acAccurateTimer

Overview
The acAccurateTimer component is the thread-based timer which periodically triggers the OnTimer events by specified Interval, but unlike standard TTimer, which based on Win32API and thus loses timer messages on high application overload, this timer runs in the separate thread and accurately triggers each OnTimer event at right time. It also allows to specify the Interval with higher precision (less than 55msec, limited in Win9x) and does not dependent to the system-wide limitations.

How to use
Drop the acAccurateTimer on your form, specify the Interval (amount of time, in milliseconds, that passes before the component initiates another OnTimer event), and write the OnTimer event handler to execute an action at regular intervals.

See also
acCronJob and acAwayTimer components.

9.1.2 Properties

9.1.2.1 Active

Applies to
acAccurateTimer, acAwayTimer and acAwayTimer components.

Declaration
```
property Active: Boolean; // True by default
```

Description
The Active property controls whether the acAccurateTimer is active and periodically triggers OnTimer events, with timer interval specified in the Interval property.

Set Active property to True to enable the timer, othwise set it to False to disable it.

See also
Interval property and OnTimer event.

9.1.2.2 Interval

Applies to
acAccurateTimer component.

Declaration
```
property Interval: Integer;
```

Description
The Interval property determines the amount of time, in milliseconds, that passes before the timer component initiates another OnTimer event.

Interval determines how frequently the OnTimer event occurs. Each time the specified interval passes, the OnTimer event occurs.

Use Interval to specify any cardinal value as the interval between OnTimer events. The default value is 1000 (one second).

Use Reset method to reset the timer and start counting the amounts of time from current moment.

See also
Active property;
Reset method;
OnTimer event.

9.1.3 Methods

9.1.3.1 Reset

Applies to
acAccurateTimer component.

Declaration
```
procedure Reset;
```

Description
The Reset method resets the timer and start counting the intervals from the current moment. After calling the Reset method, the next OnTimer event will occur after passing the amount of time specified in the Interval property.

See also
Active and Interval properties;
OnTimer event.

© 1998-2006, UtilMind Solutions®

9.1.4 Events

9.1.4.1 OnTimer

Applies to
acAccurateTimer component.

Declaration
```
property OnTimer: TNotifyEvent;
```

Description
The OnTimer event occurs when a specified amount of time, determined by the Interval property, has passed. Write OnTimer event handler to execute an action at regular intervals.

The Interval property of a timer determines how frequently the OnTimer event occurs. Each time the specified interval passes, the OnTimer event occurs.

Use Reset method to reset the timer and start counting the amounts of time from current moment.

See also
Active and Interval properties;
Reset method.

9.2 acAnimationEffect

9.2.1 TacAnimationEffect

Overview
The acAnimationEffect component implements smooth animation effects on opening and closing of your windows and dialog boxes.

How to use ?
Simply drop acAnimationEffect onto your form and make AnimateOnShow and AnimateOnHide properties True, and your form will appeared and disappeared with extremely cool animation effects without any line of code!

To choose definite animation effect - specify it in EffectType property. By default, component will show effects randomly.

To show animation effects manually - call Open and Close methods.

See also
acWin2kEffects component.

9.2.2 Properties

9.2.2.1 AnimateOnShow

Applies to
acAnimationEffect and acWin2kEffects components.

Declaration
```
property AnimateOnShow: Boolean;
```

Description

© 1998-2006, UtilMind Solutions®

The AnimateOnShow property controls whether you want to show Open Animation Effect (specified in EffectType property) during appearing window on screen.

See also
AnimateOnHide property.

9.2.2.2 AnimateOnHide

Applies to
acAnimationEffect and acWin2kEffects components.

Declaration
```
property AnimateOnHide: Boolean;
```

Description
The AnimateOnHide property controls whether you want to show Close Animation Effect (specified in EffectType property) during disappearing window from screen.

See also
AnimateOnShow property.

9.2.2.3 Delay

Applies to
acAnimationEffect component.

Declaration
```
property Delay: Byte; // (CPU Frequency * Delay) ticks
```

Description
The Delay property controls the delay in milliseconds between every step of animation. By default, Delay is 100 ticks.

9.2.2.4 EffectType

Applies to
acAnimationEffect component.

Declaration
```
type
  TacAnimationEffectType = (etRandom, etSpin, etVortex,
etScatterGather, etSpike, etFireworks);

property EffectType: TacAnimationEffectType;
```

Description
The EffectType property specifies the type of AnimationEffect, which will be played on calling the Open or Close methods.

Besides the type of effect you may specify the number of animation Frames and Steps.

If EffectType is etRandom, acAnimationEffect component will show random effect - Spin, Vortex, ScatterGather, Spike or Fireworks. You may also specify number of FireworkBlasts.

© 1998-2006, UtilMind Solutions®

9.2.2.5 FireworkBlasts

Applies to
acAnimationEffect component.

Declaration
```
property FireworkBlasts: Byte;
```

Description
The FireworkBlasts property cpecifies the number of blast-points for Firework animation effect. Default value is 10 blasts. See EffectType property for more info.

9.2.2.6 Frames

Applies to
acAnimationEffect component.

Declaration
```
property Frames: Byte;
```

Description
The Frames property specifies the number of animation frames. Default value is 6.

See also
EffectType and Steps properties.

9.2.2.7 Steps

Applies to
acAnimationEffect component.

Declaration
```
property Steps: Byte;
```

Description
The Steps property specifies the number of steps on drawing the animation effect. Default value is 50 steps.

See also
EffectType and Frames properties.

9.2.3 Methods

9.2.3.1 Open

Applies to
acAnimationEffect component.

Declaration
```
procedure Open;
```

Description
The Open method draws smooth animation effect which simulate opening of window.

Example
```
procedure TMainForm.ButtonClick(Sender: TObject);
begin
```

© 1998-2006, UtilMind Solutions®

```
  with TAboutForm.Create(Application) do
   try
     acAnimationEffect1.Open;
     ShowModal;
     acAnimationEffect1.Close;
   finally
     Free;
   end;
end;
```

See also
EffectType property and example of usage of the Open method;
Close method and OnOpenEffect and OnCloseEffect events.

9.2.3.2 Close

Applies to
acAnimationEffect component.

Declaration
```
procedure Close;
```

Description
The Close method draws smooth animation effect which simulate closing of window.

Example
```
procedure TMainForm.ButtonClick(Sender: TObject);
begin
  with TAboutForm.Create(Application) do
   try
     acAnimationEffect1.Open;
     ShowModal;
     acAnimationEffect1.Close;
   finally
     Free;
   end;
end;
```

See also
EffectType property and example of usage of the Close method.
Open method and OnOpenEffect and OnCloseEffect events.

9.2.4 Events

9.2.4.1 OnShowEffect

Applies to
acAnimationEffect component.

Declaration
```
property OnShowEffect: TNotifyEvent;
```

Description
The OnShowEffect event occurs when the form is about to appear on screen and the component preparing to produce the Open animation effect.

Example

© 1998-2006, UtilMind Solutions®

```
// Following example demonstrates how to prevent
// displaying of the effect after first show:
procedure TRegistrationForm.acAnimationEffect1ShowEffect(Sender:
TObject);
begin
  acAnimationEffect1.AnimateOnShow := False;
end;
```

See also
OnHideEffect event;
Open, Close methods.

9.2.4.2 OnHideEffect

Applies to
acAnimationEffect component.

Declaration
```
property OnHideEffect: TNotifyEvent;
```

Description
The OnHideEffect event occurs when the form is about to be closed or hidden and the component is about to produce the Close animation effect.

See also
OnShowEffect event;
Open, Close methods.

9.3 acAppAutoRun

9.3.1 TacAppAutoRun

Overview
The acAppAutoRun component is able to run your program on every Windows startup (after system reboot). Simply drop this component onto your main form, set up a few properties and your program will automatically running after every reboot.

With acAppAutoRun you can easily determinate when your program started by user or automatically on Windows startup. When it started automatically, your program will have "/autorun" command line parameter that can be easily determined on OnAutoRun event or by standard *ParamStr(1)* function. You can check this parameter in your program to identify whether the program was started automatically or by user.

Key properties
AutoRun controls whether the application is able to auto-run on startup.
RunFor controls for whom this app should be started (for current user or for all users on current machine).

events
OnAutoRun occurs when application started on startup.

Cool features
acAppAutoRun is successor of acCustomCaptionButton class. It may have corresponding button on the form's title bar and the menu item associated with this button in the system menu.

© 1998-2006, UtilMind Solutions®

How to change the glyphs for auto-run button ?

To change the glyphs for roll up/down buttons you have to modify the ACCB_AUTORUN_ENABLED and ACCB_AUTORUN_DISABLED bitmap resources in the "CaptionButtons.res" file of the AppControls pack. You can change them using standard Image Editor program which supplies with every Delphi/BCB version.

After modifying of these glyphs in the "CaptionButtons.res", they will always with you in any programs, so you will do not need to specify the glyphs every time when you drop component onto form.

See also

Other components which contains a built-in caption button: acTrayIcon, acFormHelp, acFormTopmost, acFormRoller.

9.3.2 Properties

9.3.2.1 AutoRun

Applies to

acAppAutoRun component

Declaration

```
property AutoRun: Boolean;
```

Description

The AutoRun property controls whether the application is able to be automatically restarted on Windows startup.

How does it works?

If you set AutoRun property to True, acAppAutoRun will install or uninstall your program to/from the list of startup programs, in `'\Software\Microsoft\Windows\CurrentVersion\Run'` (or `'\Software\Microsoft\Windows\CurrentVersion\RunOnce'` if RunOnce property is True) registry key. RootKey can be HKEY_CURRENT_USER or HKEY_LOCAL_MACHINE, dependently of RunFor setting.

Every program installed with acAppAutoRun will have an Application.Title property value as name of registry key, and Application.ExeName + '/autorun' as value of registry key.

When you switching the AutoRun property value, caption button state will be changed accordingly (if True or if False). Menu item in the system menu will be checked (if True) or unchecked if False.

Notes

1. Design-time property value may be differ to run-time value. This value will be read from the system registry on program start. If your program was not installed to the registry, acAppAutoRun component will use value specified at design-time.

If you would like to make the AutoRun always True on FIRST program startup, you can use the acAppStartCounter component. See an example for more details.

2. When the acAppAutoRun installs the program to the startup list, it stores the line with '/autorun'

© 1998-2006, UtilMind Solutions®

command-line parameter, used to determinate whether the program has been started automatically on Windows startup, or manually by user (by another program). So, if the program was started automatically on Windows startup, it contains the '/autorun' argument in the command-line + the OnAutoRun event occurs. Use the ParamStr(1) function to determinate how the program has been started programatically.

See also
RunFor, RunOnce and MenuItem properties;
OnAutoRun event.

9.3.2.2 AutoStarted

Applies to
acAppAutoRun component.

Declaration
```
property AutoStarted: Boolean; // read only and run-time only
```

Description
The AutoStarted property determines whether the program current working session was executed automatically on Windows startup. It returns True if the program was started after reboot, or False otherwise, if it was started by user or other non-system program.

Actually, it just checks whether the command line which was used on execution of program are contains the '/autorun' parameter. If this parameter exists, the acAppAutoRun thought that it was executed on Windows startup.

9.3.2.3 CaptionButton

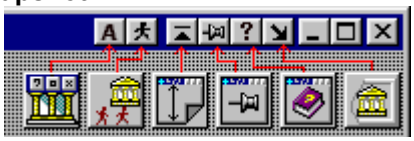
Applies to
acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost and acFormRoller components.

Declaration
```
property CaptionButton: TacACaptionButton;
```

Description
The CaptionButton is the set of properties for managing an additonal buttons on the form's title bar:

BtnOrder	controls the button placement order on the title bar;
Cursor	specifies the cursor image when button released;
CursorDown	specifies the cursor image when button pressed;
Enabled	enables or disables button;
Hint	specifies the button's tooltip;
SeparatorWidth	width between current and previous buttons;
ShowHint	enables or disables the tooltip (hint);
Visible	shows or hides the button from the title bar.

Snapshot



9.3.2.3.1 BtnOrder

Applies to
acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller components as

© 1998-2006, UtilMind Solutions®

subproperty of CaptionButton.

Declaration

```
property BtnOrder: TacBtnOrder;
```

Description

The BtnOrder property indicates position of caption button, the order in which buttons will be displayed. BtnOrder calculates from right side of title bar to left side (see snapshotSnapshot below).

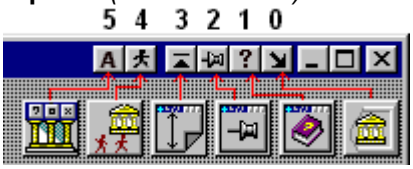
Initially, the button order is always the order in which the components were added to the form, but you can change this by changing the BtnOrder property. The value of the BtnOrder property is unique for each ancestor of the acCaptionButton on the form (like TabOrder property of ancestors of TWinControl). The first caption button added to the form has a BtnOrder value of 0, the second is 1, the third is 2, and so on. These values determine where a control is on the title bar.

If you attempt to give a button a BtnOrder value greater than the number of caption buttons on the form minus one (because numbering starts with 0), AppControls won't accept the new value, but will enter the value that assures the component will be the last in the button order.

The caption button with the BtnOrder value of 0 is the custom button which placed on right side of title bar, after regular system buttons.

To separate the buttons by additional empty space — use SeparatorWidth property. For example snapshot below shows that acAppAutoRun component have BtnOrder = 4 (fifth position from the left of system buttons) and SeparatorWidth = 4 pixels.

Snapshot *(buttons orders):*



See also

SeparatorWidth property and acCaptionButton component

9.3.2.3.2 Cursor

Applies to

acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller components as subproperty of CaptionButton.

Declaration

```
property Cursor: TCursor;
```

Description

The Cursor property controls the mouse cursor shape used when the mouse moves over CaptionButton in released state.

See also

CursorDown property and acCaptionButton component.

9.3.2.3.3 CursorDown

Applies to

acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller components as

© 1998-2006, UtilMind Solutions®

subproperty of CaptionButton.

Declaration

```
property CursorDown: TCursor;
```

Description

The CursorDown property controls the mouse cursor shape used when the mouse moves over CaptionButton in pressed state.

See also

Cursor property and acCaptionButton component.

9.3.2.3.4 Enabled

Applies to

acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller components as subproperty of CaptionButton.

Declaration

```
property Enabled: Boolean;
```

Description

The Enabled property controls whether the button on the title bar responds to mouse and keyboard messages. If Enabled is True, button responds normally. If Enabled is False, button become disabled and user can not press that button.

See also

Visible property of CaptionButton and acCaptionButton component.

9.3.2.3.5 Hint

Applies to

acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller components as subproperty of CaptionButton.

Declaration

```
property Hint: String;
```

Description

The Hint property is the text string that can appear when mouse pointer moves over the caption button.

If the CaptionButton's ShowHint property is False, the Help Hint won't appear, but the other hints still will.

Default Hint string for acAppAutoRun component is 'Auto-run at StartUp'. You may change this string at run- or design-time if you're writting multilingual program.

See also

acCaptionButton component and ShowHint property.

9.3.2.3.6 SeparatorWidth

Applies to

acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller components as subproperty of CaptionButton.

© 1998-2006, UtilMind Solutions®

Declaration

```
property SeparatorWidth: Byte; // in screen pixels
```

Description

The SeparatorWidth property controls the space between current and previous by order caption button in screen pixels.

See also

BtnOrder property and acCaptionButton component.

9.3.2.3.7 ShowHint

Applies to

acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller components as subproperty of CaptionButton.

Declaration

```
property ShowHint: Boolean;
```

Description

The ShowHint property controls whether CaptionButton can show the Hint string when mouse pointer moves over button. If ShowHint is True, Hint will appears.

See also

acCaptionButton component and Hint property.

9.3.2.3.8 Visible

Applies to

acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller components as subproperty of CaptionButton.

Declaration

```
property Visible: Boolean;
```

Description

The Visible property determines whether the caption button currently visible on the title bar. Make Visible property True if you would like to show the button on title bar, or False if you would like to hide the button.

When you changing the Visible property, placements of others button will automacially recalculated and other buttons with higher BtnOrder will be shifted to the right side of the title bar, filling the empty space.

See also

Enabled property of CaptionButton and acCaptionButton component.

9.3.2.4 LogonNotice

Applies to

acAppAutoRun component.

Declaration

```
type
  TacAutoRunLogonNotice = class(TPersistent)
  published
    property Caption: String;
```

© 1998-2006, UtilMind Solutions®

```
    property Enabled: Boolean;
    property Text: String;
  end;

property LogonNotice: TacAutoRunLogonNotice;
```

Description

The LogonNotice is the list of properties for working with logon message that can be displayed on system startup. If you would like to set the notice message on Windows startup make Enabled property True. To specify or get the notice message to / from system registry — use Caption (caption of dialog box) and Text properties.

This notice message usually stored in the System Registry, in 'HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\WinLogon' key.

9.3.2.4.1 Caption

Applies to

acAppAutoRun component as subproperty of LogonNotice.

Declaration

```
property Caption: String;
```

Description

The Caption property specifies the caption for a notice dialog message that could be displayed after restarting of your computer.

See also

Text property for LogonNotice.

9.3.2.4.2 Enabled

Applies to

acAppAutoRun component as subproperty of LogonNotice.

Declaration

```
property Enabled: Boolean;
```

Description

The Enable property controls whether you would like to show logon notice on Windows startup.

9.3.2.4.3 Text

Applies to

acAppAutoRun component as subproperty of LogonNotice.

Declaration

```
property Text: String;
```

Description

The Text property specifies the text for notice message on Windows startup. This message will be displayed if you've enabled the LogonNotice feature.

See also

Caption property for LogonNotice.

© 1998-2006, UtilMind Solutions®

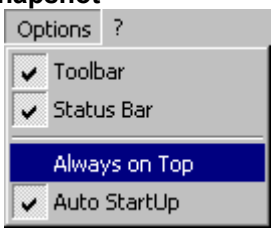
9.3.2.5 MenuItem

Applies to
acAppAutoRun component.

Declaration
```
property MenuItem: TComponent;
```

Description
The MenuItem property can be used to associate the menu item of your application, which should reflect current state of AutoRun property. When user clicks that menu item, the state of AutoRun property will be changed accordingly.

As you can see, not only standard MenuItems can be assigned to this property. You associate to it ANY component which has "Checked" or "Down" property (anything that can reflect the state of boolean value, i.e: non-standard menu item or "speed button").

Snapshot


<----------

See also
AutoRun property.

9.3.2.6 RunFor

Applies to
acAppAutoRun component.

Declaration
```
type
  TacRegLocation = (rlCurrentUser, rlLocalMachine);

property RunFor: TacRegLocation;
```

Description
The RunFor property controls for whom your program should be started:
`rlCurrentUser` — for current user only;
`rlLocalMachine` — for all users on current PC.

See also
RunOnce property.

9.3.2.7 RunOnce

Applies to
acAppAutoRun component.

Declaration
```
property RunOnce: Boolean;
```

© 1998-2006, UtilMind Solutions®

Description

The RunOnce property controls whether the application should be started before others startup programs and only once.

When the program was started on startap with RunOnce flag True, AutoRun flag will be cleaned and automatically gets False.

See also

RunFor property.

9.3.2.8 SystemMenu

Applies to

acCaptionButton component and other components that uses TacCustomCaptionButton as ancestor class - acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller.

Declaration

```
property SystemMenu: TacSystemMenu;
```

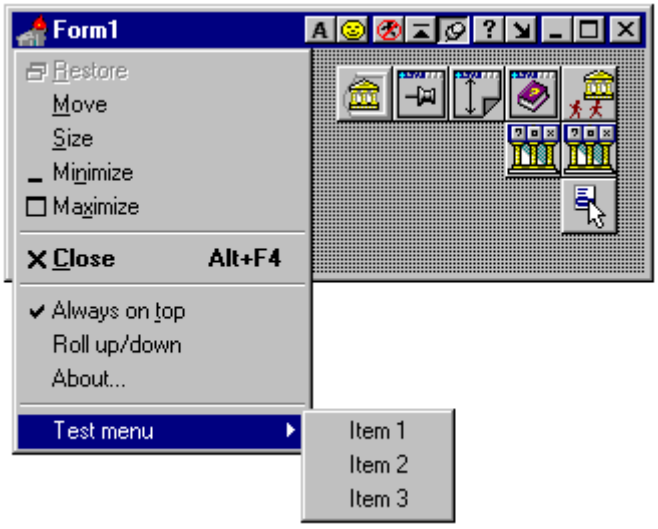
Description

The System Menu is the popup menu that appears when you click on the program icon on the upper-left corner of form's title bar. The SystemMenu property is the list of properties that manages the menu item associated with current button on the form's title bar.

Properties

ApplyToMenu	applies or removes the menu item associated with current button from the system menu;
Caption	text for the title of menu item;
Position	position of current menu item in the system menu;
Separators	specifies the separators for menu item, to separatate it from previous and / or next menu item of the system menu.

Snapshot



See also

acFormSystemMenu component.

© 1998-2006, UtilMind Solutions®

9.3.2.8.1 ApplyToMenu

Applies to

acCaptionButton component and all other components which uses CaptionButton as base ancestor class - acAppAutoRun, acTrayIcon, acFormTopmost, acFormRoller. ApplyToMenu is subproperty of SystemMenu.

Declaration

```
property ApplyToMenu: Boolean;
```

Description

The ApplyToMenu property controls whether the CaptionButton currently have the menu item assiciated with this button. Set ApplyToMenu property to True to add according menu item (with text specified by Caption property, in position specified by Position property) to the form's system menu and False to remove.

See also

Caption property.

9.3.2.8.2 Caption

Applies to

acCaptionButton component and all other components which uses CaptionButton as base ancestor class - acAppAutoRun, acTrayIcon, acFormTopmost, acFormRoller. Caption is subproperty of SystemMenu.

Declaration

```
property Caption: String;
```

Description

The Caption property specifies the text for menu item in the form's system menu, associated with current caption button on the title bar. If Caption is not specified, text from Hint property will be taken as item title.

See also

Hint property of CaptionButton

9.3.2.8.3 Position

Applies to

acCaptionButton component and all other components which uses CaptionButton as base ancestor class - acAppAutoRun, acTrayIcon, acFormTopmost, acFormRoller. Position is subproperty of SystemMenu.

Declaration

```
property Position: Word;
```

Description

The Position property determines the position for menu item in the system menu associated with current caption button on the form's title bar. Change the Position value to move current menu item by system menu.

Maximum value for Position property is the current maximum number of items in the system menu (since orders for menu items starts with 0).

See also

© 1998-2006, UtilMind Solutions®

Separators property.

9.3.2.8.4 Separators

Applies to
acCaptionButton component and all other components which uses CaptionButton as base ancestor class - acAppAutoRun, acTrayIcon, acFormTopmost, acFormRoller. Separators is subproperty of SystemMenu.

Declaration
```
type
  TacMenuSeparators = set of (seBefore, seAfter);

property Separators: TacMenuSeparators;
```

Description
The Separators property specifies the separators for current menu item to separate it from previous and / or next menu items of the system menu. To set separator before menu item — set `seBefore` to True. To set separator after menu item — set `seAfter` to True.

Specify Separators to separate current menu item in the system menu from another menu items.

See also
Position property.

9.3.3 Events

9.3.3.1 OnAutoRun

Applies to
acAppAutoRun component.

Declaration
```
property OnAutoRun: TNotifyEvent;
```

Description
The OnAutoRun event occurs only when the program has been started on the Windows startup (after system reboot). If program started manually or by another program — OnAutoRun will not occurs.

On starting the application, acAppAutoRun component will check whether the '/autorun' argument are presented between the command line parameters. If the *ParamStr(1)* contains the '/autorun' line, the OnAutoRun occurs to inform the application about automatic starting on Windows startup

Example
```
procedure TForm1.acAppAutoRun1AutoRun(Sender: TObject);
begin
  Application.ShowMainForm := False;
  acTrayIcon1.MinimizeToTray;
end;

{ ...or, in case if you are using acAppEvents component to manage
Application properties, you can write following handler for OnAutoRun
event: }
procedure TForm1.acAppAutoRun1AutoRun(Sender: TObject);
begin
  acAppEvents1.ShowMainForm := False;
```

© 1998-2006, UtilMind Solutions®

```
  acAppEvents1.ShowTaskIcon := False;
  acTrayIcon1.MinimizeToTray;
end;
```

You can also check this parameter yourself, to identify whether the program was started automatically or by user. For example, if the program was started on Windows startup, you want to make some specific operations without showing the main form:

```
program MyProgram;

uses
  Forms,
  Main in 'Main.pas' {Form1};

{$R *.RES}

begin
  if ParamStr(1) = '/autorun' then
   begin
    // ... some operations
   end
  else
   begin
    Application.Initialize;
    Application.CreateForm(TForm1, Form1);
    Application.Run;
   end;
end.
```

See also

AutoRun property.

9.3.3.2 OnButtonClick

Applies to

acAppAutoRun component.

Declaration

```
type
  TacAppAutoRunButtonClickEvent = procedure(Sender: TObject; AutoRun:
Boolean) of object;

property OnButtonClick: TacAppAutoRunButtonClickEvent;
```

Description

The OnButtonClick event occurs when user clicks the button on title bar or selects the menu item in system menu, associated with this acAppAutoRun component. The ***AutoRun*** parameter determines the current button state (ot) and value of AutoRun property.

See also

OnClick property of TacCaptionButton component.

9.4 acAppBar

9.4.1 TacAppBar

Overview

The acAppBar component lets your forms to behave like an Application Desktop Toolbar — to dock on the edges of the screen like usual taskbar or MS-Office panel. Though appbars are usually

© 1998-2006, UtilMind Solutions®

docked on an edges of the user's screen, they also can float as usual windows.

When the AppBar is anchored to the screen edge, it can be automatically hidden from screen when other window activated and popup on screen again when user point the mouse to the thin line on screen edge. User can dock or undock the appbar just moving it with mouse pointer. All automatic movements of the appbar can be displayed with smooth sliding effect. Component contains a whole bunch of neat additional features, such like docking and sizing rules, float restrictions, registry saver, animation effects and so forth...

How to use ?

Most simple and quickest way to test the Application Desktop Toolbar features — drop acAppBar component on your form, compile and execute your program. Then try to move your form driving it to the screen edges. Form will be docked and undocked by mouse pointer. You may change the Placement property in your program and docking placement will be changed accordingly with smooth animation effect (if Sliding is enabled).

You can also specify initial placement at design-time, changing value of Placement property. To disallow docking on some screen edges — use AllowedEdges property. To let the AppBar automatically hide from screen — make AutoHide property True.

Also, the acAppBar can automatically save and restore form placement on every program restart, see RegistrySaver structure.

Basic features / properties

AllowedEdges	specifies whether your form is able or unable to be docked to enumerated edges;
AlwaysOnTop	ensures that the appbar is always visible, even when you run another programs in a maximized window;
AutoHide	controls whether AppBar can be hidden (reduced to a thin line) at the edge of screen;
Placement	controls current AppBar placement. Change the Placement property to dock / undock the AppBar to specified edge. The replacement can be shown with smooth animation effect if Sliding effect is enabled;
LastDockingPlace	determines the previous AppBar placement. When user doubleclicks the AppBar, it will restore previous placement;
Docking	structure controls the AppBar size when form is docked to the screen edge. You can enable or disable docking rules, specify maximum and minimum horizontal and vertical size when form is docked, specify width and height of the AppBar when it docked to the screen edge;
Sizing	structure controls sizing rules of the AppBar. You can specify the horizontal and vertical increment of the form with AppBar;
Sliding	structure controls whether you would like to show automatic AppBar movements with smooth sliding effect. In example, your form is docked to the screen edge and it able to auto-hide from screen. When user move mouse pointer to thin line on screen edge, for will appears with sliding effect. Time of effect can be specified in the SlideTime property;
FloatRestrictions	structure controls the size restrictions when form in normal, "floating" state (when Placement property is bpFloat);
RegistrySaver	saves and restores from the system registry all AppBar settings to the on program restart;

© 1998-2006, UtilMind Solutions®

TaskIcon — specifies whether you would like to show the program icon on taskbar when the AppBar docked to screen edge.

Some frequently asked questions

Q. When user doubleclicks the form, the AppBar will restore previous placement [*LastDockingPlace property*]. Is there a way to avoid re-positioning of the AppBar ?
A. To disallow restoring of previous placement on double click, just hook OnDblClick event.

Q. I have docked the AppBar to the left border of screen on program startup but the form appears with the same width as in "floating" state!
A. Seems that your form contains a lot of non-resizeable (not aligned) controls and should have enough space to show them, however can not auto-scroll for displaying. Just set "AutoScroll" property of your form to True.

Q. I have docked the form to the screen edge, but seems I can not resize it. Your demo programs allows to resize docked appbars. I have also checked Enabled property of Sizing structure and it's True. What I doing wrong?
A. Sure, you can not resize forms with fixed BorderStyle (i.e.: bsDialog or bsSingle). Just make your form resizeable, set BorderStyle property to bsSizeable or bsSizeToolWin.

Q. Is there a way to delay the popup when anchored to the edge of the screen [*when the AppBar is "autohidden"*]? (I've had a couple users complain that they moved their mouse to fast when trying to get to a desktop icon or somewhere else and went to far and caused the appbar to pop out even though they quickly moved it away)
A. Yes. You should hook OnUnhiding event and unhide the AppBar with delay (delay can be implemented with TTimer component), using ShowHiddenAppBar method. Please check out an **example** for more details.

Q. I have a form with AppBar on it. There is a panel on the form. This hides part of the form. When I try to drag the form to dock it to the side of the screen or to undock it obviously now it doesn't work. Could you suggest a way in which I could fire the event which docks, undocks and moves the form when the user clicks or tries to drag the panel component.
A. In **v1.7** we have added bonus component, **TacDragPanel**. You can drop this panel to the AppBar and user will be able to move and dock the appbar dragging this panel.

9.4.2 Properties

9.4.2.1 AllowedEdges

Applies to

acAppBar component.

Declaration

```
type
  TacAppBarPlacement = (bpLeft, bpTop, bpRight, bpBottom, bpFloat);
  TacAppBarEdges = set of TacAppBarPlacement;

property AllowedEdges: TacAppBarEdges;
```

© 1998-2006, UtilMind Solutions®

Description

The AllowedEdges property is the set of possible placements for the Application Desktop Toolbars and defines the screen edges to which the AppBar can be docked.

To allow the AppBar to be docked to the left and right edge of screen — make both `bpLeft` and `bpRight` True. To allow the AppBar to be docked to the top and bottom edges — make `bpTop` and `bpBottom` properties True. To disallow the "undocking" — set the `bpFloat` to False.

Example

For example, you would like to allow your AppBar to be docked to the top and bottom edges of screen, disallow docking to the left and right edges and disallow the floating. In other words, the form should be always docked to the top or bottom edge of screen only and can not be undocked.

In this case you should set `bpTop` and `bpBottom` to True and other settings to False:

```
acAppBar1.AllowedEdges := [bpTop, bpBottom];
```

See also

Placement and LastDockingPlace properties.

9.4.2.2 AlwaysOnTop

Applies to

acAppBar component.

Declaration

```
property AlwaysOnTop: Boolean;
```

Description

The AlwaysOnTop property ensures that the appbar is always visible, even when you run another programs in a maximized window. AlwaysOnTop determines whether the appbar can always stay on top over other windows. If AlwaysOnTop is True, system will place your form at the top of the Z order and your form will always on top over other windows. If AlwaysOnTop is False, another windows can be placed over your form.

See also

AutoHide property.

9.4.2.3 AutoHide

Applies to

acAppBar component.

Declaration

```
property AutoHide: Boolean;
```

Description

The AutoHide property controls whether the AppBar can be hidden (reduced to edge of screen) when your program deactivates and mouse pointer moves outside of the AppBar form. The "autohidden" AppBar can be redisplayed when mouse point to the thin line at the screen edge.

When you set the AutoHide to True, make sure that the AlwaysOnTop are also True. If AutoHide is True but AlwaysOnTop is False, the AppBar will pop out below the others windows and will "sinks" behind them.

Note

© 1998-2006, UtilMind Solutions®

If you will try to dock the AppBar to the edge of screen with other, already anchored "autohidden" AppBar, component will be unable to hide. In this case AutoHide property value became False and the OnUnableToAutoHide event occurs.

See also
AlwaysOnTop and DontSlideOnTaskBarWhenAutoHide properties;
OnUnableToAutoHide, OnHiding and OnUnhiding events.

9.4.2.4 Docking

Applies to
acAppBar component.

Declaration
```
type
  TacAppBarDocking = class
  published
    property Enabled: Boolean;

    // all values in screen pixels
    property Height: Word;
    property Width: Word;
    property MaxHeight: Word;
    property MaxWidth: Word;
    property MinHeight: Word;
    property MinWidth: Word;
  end;
```

Description
The docking structure used to specify some rules for docked AppBars (when they are anchored to the screen edge).

Width, MaxWidth and MinWidth properties controls the width of the AppBar when it docked to left or right edge of screen.

Height, MaxHeight and MinHeight properties controls the height of the AppBar when it docked to top or bottom edge of screen.

The width and height of docked appbar can be specified and determinated using Width and Height properties. Width property controls current width of the appbar when it docked to the left or right edge of screen. Height property controls current current height when the appbar docked to top or bottom edge.

To enable or disable sizing restrictions — use Enabled property.

See also
FloatRestrictions structure, which specifies rules for undocked, "floating" forms.

9.4.2.4.1 Enabled

Applies to
all sub-structures of acAppBar component.

Declaration
```
property Enabled: Boolean;
```

© 1998-2006, UtilMind Solutions®

Description
The Enabled property of Docking structure controls whether the AppBar should use docking rules specified by MaxWidth, MaxHeight, MinWidth and MinHeight properties.

9.4.2.4.2 Width

Applies to
acAppBar component as subproperty of Docking structure.

Declaration
```
property Width: Word;
```

Description
The Width property determines current height of the AppBar when it docked to left or right edge of screen.

See also
Height property of Docking structure.

9.4.2.4.3 Height

Applies to
acAppBar component as subproperty of Docking structure.

Declaration
```
property Height: Word;
```

Description
The Height property determines current height of the AppBar when it docked to top or bottom edge of screen.

See also
Width property of Docking structure.

9.4.2.4.4 MaxWidth

Applies to
acAppBar component as subproperty of Docking and FloatRestriction structures, and to the acFormSizeRestrictions component.

Declaration
```
property MaxWidth: Word;
```

Description
The MaxWidth property of Docking structure specifies maximum width for the appbars docked to left or right edge of screen.

See also
MaxWidth, MinWidth, MaxHeight and MinHeight properties.

9.4.2.4.5 MaxHeight

Applies to
acAppBar component as subproperty of Docking and FloatRestriction structures, and to the acFormSizeRestrictions component.

Declaration

© 1998-2006, UtilMind Solutions®

```
property MaxHeight: Word;
```

Description
The MaxHeight property of Docking structure specifies maximum height for the appbars docked to top or bottom edge of screen.

See also
MaxWidth, MinWidth, MaxHeight and MinHeight properties.

9.4.2.4.6 MinWidth

Applies to
acAppBar component as subproperty of Docking and FloatRestriction structures, and to the acFormSizeRestrictions component.

Declaration
```
property MinWidth: Word;
```

Description
The MinWidth property of Docking structure specifies minimum width for the appbars docked to left or right edge of screen.

See also
MaxWidth, MinWidth, MaxHeight and MinHeight properties.

9.4.2.4.7 MinHeight

Applies to
acAppBar component as subproperty of Docking and FloatRestriction structures, and to the acFormSizeRestrictions component.

Declaration
```
property MinHeight: Word;
```

Description
The MinHeight property of Docking structure specifies minimum height for the appbars docked to top or bottom edge of screen.

See also
MaxWidth, MinWidth, MaxHeight and MinHeight properties.

9.4.2.5 DontSlideOnTaskBarWhenAutoHide

Applies to
acAppBar component.

Declaration
```
property DontSlideOnTaskBarWhenAutoHide: Boolean;
```

Description
The DontSlideOnTaskBarWhenAutoHide property controls whether the AppBar should not appear over the taskbar when it slides from the screen side (when user activates it from hidden state).

By default, this property set to False, so it will slide from whole side of screen. Set it to False if you don't want to obscure other application desktop toolbars, when your AppBar became activated.

See also

© 1998-2006, UtilMind Solutions®

AutoHide property.

9.4.2.6 FloatRestrictions

Applies to
acAppBar component.

Declaration
```
type
  TacAppBarFloatRestrictions = class
  published
    property Enabled: Boolean;

    property MaxWidth: Word;
    property MaxHeight: Word;
    property MinWidth: Word;
    property MinHeight: Word;
  end;
```

Description
The FloatRestrictions structure used to specify some restrictions to the form size, when the appbar in normal, "floating" state. To specify maximum and minimum form size use MaxWidth, MaxHeight, MinWidth and MinHeight properties.

To enable or disable floating restrictions — use Enabled property.

See also
Docking structure which specifies sizing rules for docked appbars.
acFormSizeRestrictions component.

9.4.2.6.1 Enabled

Applies to
all sub-structures of acAppBar component.

Declaration
```
property Enabled: Boolean;
```

Description
The Enabled property FolatRestrictions structure specifies whether you would like to use sizing restrictions for undocked, "floating" appbars. Set Enabled to True if you would like to specify maximum / minimum size for "floating" appbar.

9.4.2.6.2 MaxWidth

Applies to
acAppBar component as subproperty of FloatRestictions and Docking structures, and to the acFormSizeRestrictions component.

Declaration
```
property MaxWidth: Word;
```

Description
The MaxWidth property controls the maximum width of your form. If size restrictions is Enabled and MaxWidth value other than 0, you will unable to make horizontal resize of your window with width value greater than specified in MaxWidth property.

© 1998-2006, UtilMind Solutions®

If you would not to specify restriction for maximum form width, leave this value 0.

See also
MaxHeight property of FloatRestrictions structure.

9.4.2.6.3 MaxHeight

Applies to
acAppBar component as subproperty of FloatRestictions and Docking structures, and to the acFormSizeRestrictions component.

Declaration
```
property MaxHeight: Word;
```

Description
The MaxHeight property controls the maximum height of your form. If size restrictions is Enabled and MaxHeight value other than 0, you will unable to make vertical resize of your window with height value greater than specified in MaxHeight property.

If you would not to specify restriction for maximum form height, leave this value 0.

See also
MaxWidth property of FloatRestrictions structure.

9.4.2.6.4 MinWidth

Applies to
acAppBar component as subproperty of FloatRestictions and Docking structures, and to the acFormSizeRestrictions component.

Declaration
```
property MinWidth: Word;
```

Description
The MinWidth property controls the minimum width of your form. If size restrictions is Enabled and MinWidth value other than 0, you will unable to make horizontal resize of your window with width value lower than specified in MinWidth property.

If you would not to specify restriction for minimum form width, set this value to 0.

See also
MinHeight property of FloatRestrictions structure.

9.4.2.6.5 MinHeight

Applies to
acAppBar component as subproperty of FloatRestictions and Docking structures, and to the acFormSizeRestrictions component.

Declaration
```
property MinHeight: Word;
```

Description
The MinHeight property controls the minimum height of your form. If size restrictions is Enabled and MinHeight value other than 0, you will unable to make vertical resize of your window with height value lower than specified in MinHeight property.

© 1998-2006, UtilMind Solutions®

If you would not to specify restriction for minimum form height, set this value to 0.

See also
MinWidth property of FloatRestrictions structure.

9.4.2.7 LastDockingPlace

Applies to
acAppBar component.

Declaration
```
type
  TacAppBarPlacement = (bpLeft, bpTop, bpRight, bpBottom, bpFloat);

property LastDockingPlace: TacAppBarPlacement;
```

Description
The LastDockingPlace property determines previous placement of the AppBar. Previous AppBar placement always saved for further restoring it on doubleclick.

To prevent restoring previous placement on doubleclicking — hook OnDblClick event.

See also
Placement property and OnDblClick event.

9.4.2.8 Placement

Applies to
acAppBar component.

Declaration
```
type
  TacAppBarPlacement = (bpLeft, bpTop, bpRight, bpBottom, bpFloat);

property Placement: TacAppBarPlacement;
```

Description
The Placement property controls current placement of the AppBar on screen. When Placement is `bpLeft`, `bpRight`, `bpTop` or `bpBottom`, the AppBar is anchored to according screen edge. `Placement = bpFloat` means that form in currently undoked, normal "floating state".

When the AppBars in floating state, they are looks and feels like usual forms. They may have title bar, system menu and caption buttons. However, when the AppBars became docked, it lose all controls in non-client area and shifts all other application windows from screen edge giving enough space for the Application Toolbar (unless "autohideable" appbars with AutoHide = True).

User can change placement of the AppBar moving the form on screen with mouse, or you can modify it programatically at run-time, changing value of the Placement property.

After changing the Placement, previous property value will be assigned to LastDockingPlace property. Previous placement always saved for further restoring it on doubleclick.

See also
AutoHide, AlwaysOnTop and LastDockingPlace properties.
OnDblClick event.

© 1998-2006, UtilMind Solutions®

9.4.2.9 RegistrySaver

Applies to
acAppBar, acListView and acHostnameResolver components.

Declaration
```
type
  TacAppBarRegistrySaver = class
  public
    procedure Load;
    procedure Save;
  published
    property Enabled: Boolean;
    property Storage: TacAppBarStorageType;

    property RegLocation: TacRegLocation;
    property RegKey: String;

    property IniFileName: String;
    property IniSection: String

    property RestoreProperties: TacAppBarRestoreProperties;
    property Options: TacAppBarRegOptions;
  end;
```

Description
The acAppBar component able to automatically save all settings (placement, autohide mode, topmost mode, docking/sizing rules etc) on closing the AppBar and restore them again on next program startup. For this purpose used RegistrySaver structure which, disite the name, can store AppBar settings either in the system registry or .INI file.

How to use ?
To make settings restored on every program startup — set Enabled to True.

To specify where to store the settings (in the system registry or in INI file) — use Storage property.
- If **`Storage = useRegistry`**, all settings will be stored in the system registry. In this case you must point exact location where to store settings in the RegLocation and RegKey properties. You don't need to specify IniFileName and IniSection values. Leave them blank.

- If **`Storage = useIniFile`**, settings will be stored in the INI file, specified in the IniFileName property. Section of the Ini-file where to store AppBar settings should be specified in the IniSection property. In this case you don't need to specify registry keys and you may leave RegKey and RegLocation properties blank.

To specify which settings should be restored on every startup — use RestoreProperties property.

By default the RegistrySaver update stored properties when the AppBar is about to be closed. If you would like to update saved settings after *every* movement or resizing — make `saveOnMoveResize` = True in the Options property.

9.4.2.9.1 Enabled

Applies to
all sub-structures of acAppBar component.

Declaration
```
property Enabled: Boolean;
```

© 1998-2006, UtilMind Solutions®

Description
The Enabled property of RegistrySaver structure controls whether you would like to automatically restore AppBar settings on every program startup, and save it when application closes.

See also
RestoreProperties and Options properties.

9.4.2.9.2 IniFileName

Applies to
acAppBar component as subproperty of RegistrySaver structure.

Declaration
```
property IniFileName: String;
```

Description
IniFileName property contains the name of the .INI file, where you would like to store all appbar settings. To specify contstant which identifies the section in the .INI file — use IniSection property.

Note

This property will be used only if you use .INI file as storage (`Storage = useIniFile`).

See also
IniSection and Storage properties of RegistrySaver structure.

9.4.2.9.3 IniSection

Applies to
acAppBar component as subproperty of RegistrySaver structure.

Declaration
```
property IniSection: String;
```

Description
IniSection property specifies the constant which identifies the section in the .INI file, specified in the IniFileName property.

Note

This property will be used only if you use .INI file as storage (`Storage = useIniFile`).

See also
IniFileName and Storage properties of RegistrySaver structure.

9.4.2.9.4 Options

Applies to
acAppBar component as subproperty of RegistrySaver structure.

Declaration
```
type
  TacAppBarRegOptions = set of (saveOnClose,
                                saveOnMoveResize);

property Options: TacAppBarRegOptions;
```

Description

© 1998-2006, UtilMind Solutions®

The Options property controls when RegistrySaver should save the AppBar properties, enumerated in the RestoreProperties property. By default, AppBar saves configuration when your application is about to be closed (when `saveOnClose` is True). However, you can save all settings after every form movement or resizing, just set `saveOnMoveResize` to True.

See also
RestoreProperties property.

9.4.2.9.5 RegKey

Applies to
acAppStartCounter, acFormPlacementSaver components, and acAppBar / acListView components as subproperty of RegistrySaver structure.

Declaration
```
property RegKey: String;
```

Description
The RegKey property defines the path to the key in the system registry where you would like to store the AppBar settings (placement / restrictions etc). All stored settings will be updated every time when user move or close the form (dependently of Options)

Default property value is `'\Software\CompanyName\ProgramName\AppBar'`. You should point the RegKey to location where stored the configuration settings of your own application. In example - `'\Software\UtilMind Solutions\The Cool Program\AppBar Settings'`.

Note
This property will be used only if you use system registry as storage (`Storage = useRegistry`).

See also
RegLocation property.

9.4.2.9.6 RegLocation

Applies to
acAppStartCounter, acFormPlacementSaver components, and acAppBar / acListView components as subproperty of RegistrySaver structure.

Declaration
```
property RegLocation: TacRegLocation;
```

Description
The RegLocation specifies the root key of the hierarchy of subkeys, where you would like to store the AppBar settings. RegLocation property can point to HKEY_CURRENT_USER (rlCurrentUser) or HKEY_LOCAL_MACHINE (rlLocalMachine) root keys.

When RegLocation is `rlCurrentUser`, acAppBar stores all settings in the registry section for current logged user. When RegLocation is `rlLocalMachine` settings will be saved for ALL users of current machine.

Note
This property will be used only if you use system registry as storage (`Storage = useRegistry`).

© 1998-2006, UtilMind Solutions®

See also
RegKey property.

9.4.2.9.7 RestoreProperties

Applies to
acAppBar component as subproperty of RegistrySaver structure.

Declaration
```
type
  TacAppBarRestoreProperties = set of (rfAllowedEdges, rfFormSize,
rfDocking, rfFloatRestrictions, rfSizing, rfSliding);

property RestoreProperties: TacAppBarRestoreProperties;
```

Description
The RestoreProperties property of the RegistrySaver structure specifies which properties, saved after previous program execution, you would like to restore on every program startup.

Values	Meaning
rfAllowedEdges	restores value of AllowedEdges property when True;
rfFormSize	restores the form Placement and size when True;
rfDocking	restores Docking rules when True;
rfFloatRestrictions	restores FloatRestrictions;
rfSizing	restores Sizing rules;
rfSliding	restures Sliding settings.

See also
Options property.

9.4.2.9.8 Storage

Applies to
acAppBar component as subproperty of RegistrySaver structure.

Declaration
```
type
  TacAppBarStorageType = (useRegistry, useIniFile);

property Storage: TacAppBarStorageType;
```

Description
The Storage property specifies where to store AppBar settings, in the system registry or in the INI file.

- If **Storage = useRegistry**, all settings will be stored in the system registry. In this case you must point exact location where to store settings in the RegLocation and RegKey properties. You don't need to specify IniFileName and IniSection values. Leave them blank.
- If **Storage = useIniFile**, settings will be stored in the INI file, specified in the IniFileName property. Section of the Ini-file where to store AppBar settings should be specified in the IniSection property. In this case you don't need to specify registry keys and you may leave RegKey and RegLocation properties blank.

© 1998-2006, UtilMind Solutions®

9.4.2.10 Sizing

Applies to
acAppBar component.

Declaration
```
type
  TacAppBarSizing = class
  published
    property Enabled: Boolean;

    property HorizontalIncrement: Word; // in pixels
    property VerticalIncrement: Word;
  end;
```

Description
The Sizing structure used to specify some rules on resizing of the appbars. You can specify horizontal and vertical increments (when user resize the appbar, it can be resized by specified number of pixels per mouse movement). To allow or disallow resizing of the AppBar — use Enabled property.

See also
FloatRestrictions structure.

9.4.2.10.1 Enabled

Applies to
all sub-structures of acAppBar component.

Declaration
```
property Enabled: Boolean;
```

Description
The Enabled property of Sizing structure controls whether user can resize the AppBar window. When Enabled is False, user can NOT resize appbars, even if appbar in "floating" state and form's BorderStyle is [bsSizeable].

See also
HorizontalIncrement and VerticalIncrement properties of Sizing structure.

9.4.2.10.2 HorizontalIncrement

Applies to
acAppBar component as subproperty of Sizing property.

Declaration
```
property HorizontalIncrement: Word; // 1 pixel by default
```

Description
The HorizontalIncrement controls the horizontal sizing increment for the appbars (both docked and "floating"). Usually you resize forms forms with 1 pixel precision. HorizontalIncrement property lets users to resize forms by specified number of pixels per mouse movement.

See also
VerticalIncrement property of Sizing structure.

© 1998-2006, UtilMind Solutions®

9.4.2.10.3 VerticalIncrement

Applies to
acAppBar component as subproperty of Sizing property.

Declaration
```
property VerticalIncrement: Word; // 1 pixel by default
```

Description
The VerticalIncrement controls the vertical sizing increment for the appbars (both docked and "floating"). Usually you resize forms forms with 1 pixel precision. VertocalIncrement property lets users to resize forms by specified number of pixels per mouse movement.

See also
HorizontalIncrement property of Sizing structure.

9.4.2.11 Sliding

Applies to
acAppBar component.

Declaration
```
type
  TacAppBarSliding = class
  published
    property Enabled: Boolean;
    property SlideTime: Word; // in milliseconds
  end;
```

Description
The sliding effect used to animate all automatic movements of the AppBar. This effect can be displayed when you programatically changing the Placement property, or when the hidden AppBar (with AutoHide = True), popup to the screen.

To enable or disable sliding effect — use Enabled property. To specify duration of effect — use SlideTime property.

9.4.2.11.1 Enabled

Applies to
all sub-structures of acAppBar component.

Declaration
```
property Enabled: Boolean;
```

Description
The Enabled property of Sliding structure determines is the sliding effect currently enabled.

When the Enabled is True, all automatic movements of the AppBar will be displayed with smooth sliding effect.

To specify duration of the sliding effect — use SlideTime property.

See also
SlideTime property of Sliding structure.

© 1998-2006, UtilMind Solutions®

9.4.2.11.2 SlideTime

Applies to
acAppBar component as subproperty of Sliding structure.

Declaration
```
property SlideTime: Word; // in milliseconds
```

Description
The SlideTime property specifies duration of sliding effect, in milliseconds, which displayed on all automatic movements of the AppBar (if sliding effect is enabled).

See also
Enabled property of Sliding structure.

9.4.2.12 TaskIcon

Applies to
acAppBar component.

Description
```
type
  TacAppBarShowBehavior = (sbShowAlways, sbHideAlways, sbOnFloat);

property TaskIcon: TacAppBarShowBehavior;
```

Declaration
The TaskIcon property controls whether you would like to show the program icon on the taskbar when the AppBar docked to screen edge.

Default property value is `sbOnFloat`, that means that task icon should appears only when the AppBar is in undocked, "floating" state.

Set TaskIcon to `sbShowAlways` to make the task icon ALWAYS visible on the taskbar, even if AppBar is docked to screen edge. When TaskIcon is `sbHideAlways` — task icon will always hidden.

See also
TitleBar property.

9.4.2.13 TitleBar

Applies to
acAppBar component.

Description
```
type
  TacAppBarShowBehavior = (sbShowAlways, sbHideAlways, sbOnFloat);

property TitleBar: TacAppBarShowBehavior;
```

Declaration
The TitleBar property controls whether you would like to show the form's title bar (window caption) when the AppBar docked to screen edge.

Default property value is `sbOnFloat`, that means that title bar should appears only when the

© 1998-2006, UtilMind Solutions®

AppBar is in undocked, "floating" state.

Set TitleBar to `sbShowAlways` to make the tiele bar ALWAYS visible, even if AppBar is docked to screen edge. When TaskIcon is `sbHideAlways`, the title bar will be always hidden (captionless form).

See also
TaskIcon property.

9.4.3 Methods

9.4.3.1 IsMouseOverAppBar

Applies to
acAppBar component.

Declaration
```
function IsMouseOverAppBar: Boolean;
```

Description
The IsMouseOverAppBar function returns True if mouse is over the form (AppBar) and False if mouse is outside of AppBar.

See also
Example

9.4.3.2 ShowHiddenAppBar

Applies to
acAppBar component.

Declaration
```
procedure ShowHiddenAppBar(Show: Boolean);
```

Description
The ShowHiddenAppBar method pops up the hidden AppBar (when AutoHide property is True and anchored to the screen edge).

When *Show* parameter is True, AppBar unhides itself from behind the screen, and hides it back if *Show* parameter is False.

Remarks
When you call the ShowHiddenAppBar method, the OnHiding and OnUnhiding events will NOT occurs!

See also
AutoHide property; OnHiding and OnUnhiding events.

9.4.3.3 UpdateBar

Applies to
acAppBar component.

Declaration
```
procedure UpdateBar;
```

Description

© 1998-2006, UtilMind Solutions®

The UpdateBar method updates the form placement and repaints the AppBar on screen.

See also
Placement property.

9.4.4 Events

9.4.4.1 OnActivate

Applies to
acAppBar and acAppEvents components.

Declaration
```
property OnActivate: TNotifyEvent;
```

Description
The OnActivate event for an appbar occurs when the appbar becomes active (i.e. when user clicks the client area with a mouse). Your appbar becomes active when it is initially run or when focus is shifted from another form or Windows application to your form (AppBar).

See also
OnDeactivate event.

9.4.4.2 OnClick

Applies to
acAppBar and acTrayIcon components.

Declaration
```
property OnClick: TNotifyEvent;
```

Description
The OnClick event occurs when the user clicks the form with acAppBar component. Typically, this is when the user presses and releases the primary mouse button with the mouse pointer over the form.

See also
OnDblClick event.

9.4.4.3 OnDblClick

Applies to
acAppBar and acTrayIcon components.

Declaration
```
property OnDblClick: TNotifyEvent;
```

Description
The OnDblClick event occurs when user double-clicks the primary mouse button over the form (AppBar). Use the OnDblClick event to write code that responds to the double-click event.

Usually, when user doubleclicks the Application Toolbar, the AppBar should restore its previous placement (see LastDockingPlace property). Hooking of the OnDblClick event will prevent the AppBar from auto-restoring previous placement.

See also
OnClick event and LastDockingPlace property.

© 1998-2006, UtilMind Solutions®

9.4.4.4 OnDeactivate

Applies to
acAppBar and acAppEvents components.

Declaration
```
property OnDeactivate: TNotifyEvent;
```

Description
The OnDeactivate event occurs when the user switches from your form (AppBar) to another form or Windows application. Use the OnDeactive event to do any special processing you want to occur before the AppBar is deactivated.

See also
OnActivate event.

9.4.4.5 OnDocked

Applies to
acAppBar component.

Declaration
```
type
  TacAppBarDockedEvent = procedure(Sender: TObject; NewPlacement:
TacAppBarPlacement) of object;

property OnDocked: TacAppBarDockedEvent;
```

Description
The OnDocked event occurs when the form has been docked to the screen edge or has restored to normal, "floating" state. *NewPlacement* parameter shows new placement of the AppBar.

To prevent docking to specific screen edge — use AllowedEdges property or OnDocking event.

See also
AllowedEdges property and OnDocking, OnDockMove events.

9.4.4.6 OnDocking

Applies to
acAppBar component.

Declaration
```
type
  TacAppBarDockingEvent = procedure(Sender: TObject; var NewPlacement:
TacAppBarPlacement) of object;

property OnDocking: TacAppBarDockingEvent;
```

Description
The OnDocking event occurs when the AppBar is about to be docked to the screen edge or be undocked (restored to its normal, "floating" state). Use *NewPlacement* parameter to specify another appbar placement for docking.

See also
AllowedEdges property and OnDocked, OnDockMove events.

© 1998-2006, UtilMind Solutions®

9.4.4.7 OnDockMove

Applies to
acAppBar component.

Declaration
```
property OnDockMove: TacAppBarDockedEvent;
```

Description
TheOnDockMove event occurs when user moves the AppBar to the screen edge with mouse.

See also
OnDocked, OnDocking, OnExitSizeMove events.

9.4.4.8 OnExitSizeMove

Applies to
acAppBar component.

Declaration
```
property OnExitSizeMove: TNotifyEvent;
```

Description
The OnExitSizeMove enent occurs after it has exited the moving or sizing mode.

See also
OnDocked, OnDocking, OnDockMove events.

9.4.4.9 OnFullScreenApp

Applies to
acAppBar component.

Declaration
```
type
  TacAppBarFullScreenAppEvent = procedure(Sender: TObject; Open:
Boolean) of object;

property OnFullScreenApp: TacAppBarFullScreenAppEvent;
```

Description
The OnFullScreenApp notifies the AppBar whether a full screen application window is open. When the full screen applications is opens, Open parameter will be True, or False otherwise.

See also
OnOtherAppBarPosChanged and OnWIndowArrange events.

9.4.4.10 OnHiding

Applies to
acAppBar component.

Declaration
```
type
  TacAppBarHidingEvent = procedure(Sender: TObject; var AllowOperation:
Boolean) of object;
```

© 1998-2006, UtilMind Solutions®

```
property OnHiding: TacAppBarHidingEvent;
```

Description
The OnHiding event occurs when the docked appbar with AutoHide = True is about to hide from screen. This event occurs when the appbar losing focus (user activates another program) and appbar disappears from screen.

Set *AllowOperation* parameter in the event handler to False to discard the operation. See an example for more details.

Remarks
The OnHiding and OnUnhiding events does not occurs when you call ShowHiddenAppBar method programatically.

See also
AutoHide property and OnUnhiding event.

9.4.4.11 OnOtherAppBarPosChanged

Applies to
acAppBar component.

Declaration

```
property OnOtherAppBarPosChanged: TNotifyEvent;
```

Description
The OnOtherAppBarPosChanged event occurs when other Application Toolbar application (i.e. taskbar or MS-Office panel) changes its placement on screen.

See also
OnFullScreenApp and OnWindowArrange events.

9.4.4.12 OnUnableToAutoHide

Applies to
acAppBar component.

Declaration

```
property OnUnableToAutoHide: TNotifyEvent;
```

Description
The OnUnableToAutoHide event occurs when user trying to dock the "autohideable" AppBar (with AutoHide = True) to the screen edge where another hidden Application toolbar already docked. Unforunately, Windows can not dock more than one hidden AppBar per screen edge.

When AppBar is unable to auto-hide, the AutoHide property becames False.

See also
AutoHide property.

9.4.4.13 OnUnhiding

Applies to
acAppBar component.

Declaration

```
type
```

© 1998-2006, UtilMind Solutions®

```
  TacAppBarHidingEvent = procedure(Sender: TObject; var AllowOperation:
Boolean) of object;

property OnUnhiding: TacAppBarHidingEvent;
```

Description

The OnUnhiding event occurs when the docked and hidden appbar (with AutoHide = True) is about to appear on screen. This event occurs when user moves the mouse pointer over thin line on screen edge and appbar became visible.

Set *AllowOperation* parameter in the event handler to False to discard the operation. See an example for more details.

Remarks

The OnHiding and OnUnhiding events does not occurs when you call ShowHiddenAppBar method programatically.

See also

AutoHide property and OnHiding event.

9.4.4.14 OnWindowArrange

Applies to

acAppBar component.

Declaration

```
type
  TacAppBarWindowArrangeEvent = procedure(Sender: TObject; Beginning:
Boolean) of object;

property OnWindowArrange: TacAppBarWindowArrangeEvent;
```

Description

The OnWindowArrange event occurs when AppBar receives system message about arranging windows.

See also

OnFullScreenApp and OnOtherAppBarPosChanged events.

9.4.5 Frequently Asked Questions

Answers to some questions which we receiving rather often

Q. When user doubleclicks the form, the AppBar will restore previous placement [*LastDockingPlace property*]. Is there a way to avoid re-positioning of the AppBar ?
A. To disallow restoring of previous placement on doubleclick, just hook OnDblClick event.

Q. I have docked the AppBar to the left border of screen on program startup but the form appears with the same width as in "floating" state!
A. Seems that your form contains a lot of non-resizeable (not aligned) controls and should have enough space to show them, however can not auto-scroll for displaying. Just set "AutoScroll" property of your form to True.

© 1998-2006, UtilMind Solutions®

Q. I have docked the form to the screen edge, but seems I can not resize it. Your demo programs allows to resize docked appbars. I have also checked Enabled property of Sizing structure and it's True. What I doing wrong?
A. Sure, you can not resize forms with fixed BorderStyle (i.e.: bsDialog or bsSingle). Just make your form resizeable, set BorderStyle property to bsSizeable or bsSizeToolWin.

Q. Is there a way to delay the popup when anchored to the edge of the screen [*when the AppBar is "autohidden"*]? (I've had a couple users complain that they moved their mouse to fast when trying to get to a desktop icon or somewhere else and went to far and caused the appbar to pop out even though they quickly moved it away)
A. Yes. You should hook OnUnhiding event and unhide the AppBar with delay (delay can be implemented with TTimer component), using ShowHiddenAppBar method. Please check out an **example** for more details.

Q. I have a form with AppBar on it. There is a panel on the form. This hides part of the form. When I try to drag the form to dock it to the side of the screen or to undock it obviously now it doesn't work. Could you suggest a way in which I could fire the event which docks, undocks and moves the form when the user clicks or tries to drag the panel component.
A. In AppControls **v2.2.1** we have added bonus component, **TacDragPanel**. You can drop this panel to the AppBar and user will be able to move and dock the appbar dragging this panel.

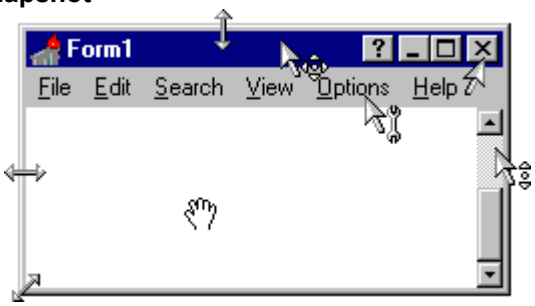
9.5 acAppCursors

9.5.1 TacAppCursors

Overview

The acAppCursors component allows to specify the cursor images for non-client parts of your application. You can specify your own default cursor for all controls, custom cursors for form captions, regluar caption buttons, scroll bars, borders, menus and menu items and so forth... One component can change appearance of cursors for entire application, even in standard Windows dialogs (such as OpenDialog / SaveDialog), that you are using in your program.

Snapshot



You can specify a custom cursors for following objects:
Client area and Inactive form

Client	used for client area of controls, if the control haven't custom cursor image (only when Cursor property of this control is crDefault/crArrow);
ClientNonVCL	used for client area of non-VCL controls and dialog boxes. In example, in client areas of the standard OpenDialog / SaveDialog windows.
Transparent	used when the form is covered by another window.

© 1998-2006, UtilMind Solutions®

Form borders:
Left, Top, Right, Bottom sides, Upper-left, Upper-right, Bottom-left, Bottom-right corners and non-resizeable border.

Regular buttons on title bar:
Close, Maximize, Minimize, Help.

Scroll bars:
Horizontal scrollbar, Vertical scrollbar.

Another objects:
Caption, Main menu, System menu, Menu item

See also
Additional cursors in Delphi/BCB IDE.

9.5.2 Properties

9.5.2.1 BorderBottom

Applies to
acAppCursors component.

Declaration
```
property BorderBottom: TCursor;
```

Description
Specifies appearance of the mouse cursor when mouse pointer moves over lower horizontal border of window.

See also
BorderBottom, BorderBottomLeft, BorderBottomRight, BorderLeft, BorderRight, BorderTop, BorderTopLeft, BorderTopRight, BorderNonResize properties.

9.5.2.2 BorderBottomLeft

Applies to
acAppCursors component.

Declaration
```
property BorderBottomLeft: TCursor;
```

Description
Specifies appearance of the mouse cursor when mouse pointer moves over lower-left corner of a window border.

See also
BorderBottom, BorderBottomLeft, BorderBottomRight, BorderLeft, BorderRight, BorderTop, BorderTopLeft, BorderTopRight, BorderNonResize properties.

9.5.2.3 BorderBottomRight

Applies to
acAppCursors component.

Declaration
```
property BorderBottomRight: TCursor;
```

© 1998-2006, UtilMind Solutions®

Description
Specifies appearance of the mouse cursor when mouse pointer moves over lower-right corner of a window border.

See also
BorderBottom, BorderBottomLeft, BorderBottomRight, BorderLeft, BorderRight, BorderTop, BorderTopLeft, BorderTopRight, BorderNonResize properties.

9.5.2.4 BorderLeft

Applies to
acAppCursors component.

Declaration
```
property BorderLeft: TCursor;
```

Description
Specifies appearance of the mouse cursor when mouse pointer moves over left vertical border of window.

See also
BorderBottom, BorderBottomLeft, BorderBottomRight, BorderLeft, BorderRight, BorderTop, BorderTopLeft, BorderTopRight, BorderNonResize properties.

9.5.2.5 BorderNonResize

Applies to
acAppCursors component.

Declaration
```
property BorderNonResize: TCursor;
```

Description
Specifies appearance of the mouse cursor when mouse pointer moves over the border of a window that does not have a sizing border.

See also
BorderBottom, BorderBottomLeft, BorderBottomRight, BorderLeft, BorderRight, BorderTop, BorderTopLeft, BorderTopRight, BorderNonResize properties.

9.5.2.6 BorderRight

Applies to
acAppCursors component.

Declaration
```
property BorderRight: TCursor;
```

Description
Specifies appearance of the mouse cursor when mouse pointer moves over right vertical border of window.

See also
BorderBottom, BorderBottomLeft, BorderBottomRight, BorderLeft, BorderRight, BorderTop, BorderTopLeft, BorderTopRight, BorderNonResize properties.

© 1998-2006, UtilMind Solutions®

9.5.2.7 BorderTop

Applies to
acAppCursors component.

Declaration
```
property BorderTop: TCursor;
```

Description
Specifies appearance of the mouse cursor when mouse pointer moves over upper horizontal border of window.

See also
BorderBottom, BorderBottomLeft, BorderBottomRight, BorderLeft, BorderRight, BorderTop, BorderTopLeft, BorderTopRight, BorderNonResize properties.

9.5.2.8 BorderTopLeft

Applies to
acAppCursors component.

Declaration
```
property BorderTopLeft: TCursor;
```

Description
Specifies appearance of the mouse cursor when mouse pointer moves over upper-left corner of a window border.

See also
BorderBottom, BorderBottomLeft, BorderBottomRight, BorderLeft, BorderRight, BorderTop, BorderTopLeft, BorderTopRight, BorderNonResize properties.

9.5.2.9 BorderTopRight

Applies to
acAppCursors component.

Declaration
```
property BorderTopRight: TCursor;
```

Description
Specifies appearance of the mouse cursor when mouse pointer moves over upper-right corner of a window border.

See also
BorderBottom, BorderBottomLeft, BorderBottomRight, BorderLeft, BorderRight, BorderTop, BorderTopLeft, BorderTopRight, BorderNonResize properties.

9.5.2.10 BtnClose

Applies to
acAppCursors component.

Declaration
```
property BtnClose: TCursor;
```

Description

© 1998-2006, UtilMind Solutions®

Specifies appearance of the mouse cursor when mouse pointer moves over Close button on the title bar.

See also
BtnMaximize, BtnMinimize, BtnClose, BtnHelp properties.

9.5.2.11 BtnHelp

Applies to
acAppCursors component.

Declaration
```
property BtnHelp: TCursor;
```

Description
Specifies appearance of the mouse cursor when mouse pointer moves over Help button on the title bar.

See also
BtnMaximize, BtnMinimize, BtnClose, BtnHelp properties.

9.5.2.12 BtnMaximize

Applies to
acAppCursors component.

Declaration
```
property BtnMaximize: TCursor;
```

Description
Specifies appearance of the mouse cursor when mouse pointer moves over Maximize button on the title bar.

See also
BtnMaximize, BtnMinimize, BtnClose, BtnHelp properties.

9.5.2.13 BtnMinimize

Applies to
acAppCursors component.

Declaration
```
property BtnMinimize: TCursor;
```

Description
Specifies appearance of the mouse cursor when mouse pointer moves over Minimize button on the title bar.

See also
BtnMaximize, BtnMinimize, BtnClose, BtnHelp properties.

9.5.2.14 Caption

Applies to
acAppCursors component.

Declaration

© 1998-2006, UtilMind Solutions®

```
property Caption: TCursor;
```

Description
Specifies appearance of the mouse cursor when mouse pointer moves over title bar of window.

9.5.2.15 Client

Applies to
acAppCursors component.

Declaration
```
property Client: TCursor;
```

Description
Specifies the cursor image when the mouse pointer moves over client area of any control (form, button, panel and so on), if the control has default cursor image (crDefault).

Remarks
The cursor image specified in the Client property appears only if the control are not contains own, custom cursor image for its client area (cursor specified in Client property appears over the control only if Cursor property of control is crDefault). In case If the control contains its own cursor image, the cursor specified in Client property will not be used.

The Client cursor works only for VCL objects, for controls written in Delphi. If you are using some ActiveX controls, or separate windows which contains the non-VCL controls — use ClientNonVCL property to change the cursor appearance.

See also
ClientNonVCL property.

9.5.2.16 ClientNonVCL

Applies to
acAppCursors component.

Declaration
```
property ClientNonVCL: TCursor;
```

Description
Specifies the cursor image when the mouse pointer moves over client area of any non-VCL cotnrol. Non-VCL controls is the ActiveX-based components on your forms, or controls used in standard dialogs (for example in TOpenDialog, TSaveDialog windows).

See also
Client property.

9.5.2.17 MainMenu

Applies to
acAppCursors component.

Declaration
```
property MainMenu: TCursor;
```

Description
Specifies appearance of the mouse cursor when mouse pointer moves over main menu of window.

© 1998-2006, UtilMind Solutions®

See also
MenuItem and SystemMenu properties.

9.5.2.18 MenuItem

Applies to
acAppCursors component.

Declaration
```
property MenuItem: TCursor;
```

Description
Specifies appearance of the mouse cursor when mouse pointer moves over menu item (main or popup menu).

See also
MainMenu and SystemMenu properties.

9.5.2.19 ScrollHorz

Applies to
acAppCursors component.

Declaration
```
property ScrollHorz: TCursor;
```

Description
Specifies appearance of the mouse cursor when mouse pointer moves over horizontal scrollbar.

See also
ScrollVert property.

9.5.2.20 ScrollVert

Applies to
acAppCursors component.

Declaration
```
property ScrollVert: TCursor;
```

Description
Specifies appearance of the mouse cursor when mouse pointer moves over vertical scrollbar.

See also
ScrollHorz property.

9.5.2.21 SystemMenu

Applies to
acAppCursors component.

Declaration
```
property SystemMenu: TCursor;
```

Description
Specifies appearance of the mouse cursor when mouse pointer moves over system menu (over icon at the left side of the title bar).

© 1998-2006, UtilMind Solutions®

See also
MenuItem and MainMenu properties.

9.5.2.22 Transparent

Applies to
acAppCursors component.

Declaration
```
property Transparent: TCursor;
```

Description
Specifies the cursor image for the form, when it currently covered by another window (for example, modal dialog box opened from this form).

9.5.3 Events

9.5.3.1 OnMouseOverObject

Applies to
acAppCursors component.

Declaration
```
type
  TacMouseOverObjectEvent = procedure(Sender: TObject; AppObject:
    TacAppObject; WndHandle: THandle; var Cursor: TCursor) of object;

property OnMouseOverObject: TacMouseOverObjectEvent;
```

Description
The OnMouseOverObject event occurs whenever mouse pointer moves over form's hot spot object. Write the OnMouseOverObject event handler to determinate the hotspot under the mouse pointer (AppObject parameter), determinate the window handle (WndHandle parameter) and determinate or change the mouse cursor for determined hotspot object.

See also
Example

9.6 acAppEvents

9.6.1 TacAppEvents

Overview
The acAppEvents component is intended for modifications of some hidden properties the TApplication class (such like appearance of hints) and hooking application events (such like idle, activate / deactivate, minimize / restore application messages). Besides usual features it's able to animate the application icon, hide / show the taskbar icon and fix the smooth animated minimize/restore effect for Delphi 2/3/4 (see FixedMinimizeRestoreAnimation property).

The acAppEvents hooks everything application-related. It can even hook ALL keypresses and ALL mouse movements within entire application.

How to use
To change the appearance of Hints — edit HintColor, HintFont, HintHidePause, HintPause and HintShortPause properties.

© 1998-2006, UtilMind Solutions®

To animate the application icon - make AnimateIcon property True, specify the AnimateInterval and point the ImageList property to TImageList collection which contains parts of animated icon.

To hook some "hidden" application events — handle the OnActivate, OnDeactivate, OnMinimize, OnRestore, OnIdle, OnHint events. See full list of supported events.

new! To hook all keypresses and mouse movements/clicks withing the ENTIRE application — use OnAppKeyDown, OnAppKeyUp, OnAppMouseMove, OnAppMouseDown and OnAppMouseUp events.

See also
acFormHook component.

9.6.2 Properties

9.6.2.1 AnimateIcon

Applies to
acAppEvents component.

Declaration
```
property AnimateIcon: Boolean;
```

Description
The AnimateIcon property controls whether you would like to animate images assigned to ImageList property. If Animate is True, acAppEvents will increase ImageIndex value, with interval specified in AnimateInterval property. If ImageList is not assigned and you have specified just one icon (in the Icon property), setting True to Animate will take not effect.

Icons can be animated only if
1. AnimateInterval value is higher than 0 (milliseconds), *and…*
2. Animate property is True, *and…*
3. Multiple images assigned to ImageList.

See also
AnimateInterval, ImageList, ImageIndex properties.

9.6.2.2 AnimateInterval

Applies to
acAppEvents and acTrayIcon components.

Declaration
```
property AnimateInterval: Word; // in milliseconds
```

Description
The AnimateInterval property determines in milliseconds the time interval (in milliseconds) that passes before the acAppEvents component increases the ImageIndex value to animate images assigned to ImageList.

Icons can NOT be animated when
1. AnimateInterval value is 0;
2. AnimateIcon property is False;
3. No images assigned to ImageList.

See also

© 1998-2006, UtilMind Solutions®

AnimateIcon, ImageList, ImageIndex properties.

9.6.2.3 AppPriorityClass

Applies to
acAppEvents component.

Declaration
```
type
  TacAppPriorityClass  = (apNormal, apIdle, apHigh, apRealTime);

property AppPriorityClass: TacAppPriorityClass;
```

Description
The AppPriority property determines and sets the priority class for the specified process. This value together with the priority value of each thread of the process determines each thread's base priority level.

There is a possible values for this property:

Values	Meaning
apNormal	Specify this priority class for a process with no special scheduling needs.
apIdle	Specify this class for a process whose threads run only when the system is idle. The threads of the process are preempted by the threads of any process running in a higher priority class. An example is a screen saver. The idle-priority class is inherited by child processes.
apHigh	Specify this class for a process that performs time-critical tasks that must be executed immediately. The threads of the process preempt the threads of normal or idle priority class processes. An example is Windows Task List, which must respond quickly when called by the user, regardless of the load on the operating system. Use extreme care when using the high-priority class, because a high-priority class application can use nearly all available CPU time.
apRealTime	Specify this class for a process that has the highest possible priority. The threads of the process preempt the threads of all other processes, including operating system processes performing important tasks. For example, a real-time process that executes for more than a very brief interval can cause disk caches not to flush or cause the mouse to be unresponsive.

9.6.2.4 FixedMinimizeRestoreAnimation

Applies to
acAppEvents component.

Declaration
```
property FixedMinimizeRestoreAnimation: Boolean;
```

Description
The FixedMinimizeRestoreAnimation property controls whether you would like to fix the smooth minimize / restore / maximize animation effect in your application. Make this property True if you woud like to fix the animation effect and add truly professional appearance for your application.

Note
This property available in Delphi 2/3/4 and C++Builder 1/3/4 only, since this animation problem already fixed in Delphi 5+ and C++Builder 5+.

© 1998-2006, UtilMind Solutions®

9.6.2.5 HintColor

Applies to
acAppEvents component.

Declaration
```
property HintColor: TColor;
```

Description
The HintColor property specifies the background color for regular hint boxes of the controls in the application. Default value is clInfoBk.

See also
HintFont property and acFormHints component.

9.6.2.6 HintFont

Applies to
acAppEvents component.

Declaration
```
property HintFont: TFont;
```

Description
The HintFont property specifies the font for text in regular hint windows. To modify a font, you change the value of the Color, Name, Size, or Style properties of the font object.

See also
HintColor property and acFormHints component.

9.6.2.7 HintHidePause

Applies to
acAppEvents component.

Declaration
```
property HintHidePause: Integer; // in milliseconds
```

Description
Use HintHidePause to specify a wait time in milliseconds, which is different from the default value of 2500 ms or 2.5 seconds that is set in the constructor. The Help Hint for the control or menu item is specified in the Hint property.

Note
The default is a predefined constant value that is several times longer than HintPause.

See also
HintPause and HintShortPause properties.

9.6.2.8 HintPause

Applies to
acAppEvents component.

Declaration
```
property HintPause: Integer; // in milliseconds
```

© 1998-2006, UtilMind Solutions®

Description
The HintPause property determines the time interval that passes when the user places the mouse pointer on a control before the control's Help Hint specified in its Hint property appears. The interval is in milliseconds. The default value is 800 milliseconds.

See also
HintShortPause and HintHidePause properties.

9.6.2.9 HintShortPause

Applies to
acAppEvents component.

Declaration
```
property HintShortPause: Integer; // in milliseconds
```

Description
The HintShortPause property specifies the pause to wait before bringing up a hint if a hint has already been shown. It is used to reduce the flicker when moving the mouse quickly over a set of buttons that all have help hints on. The default value is 50ms.

See also
HintPause and HintHidePause properties.

9.6.2.10 Icon

Applies to
acAppEvents and acTrayIcon components.

Declaration
```
property Icon: TIcon;
```

Description
The Icon property determines the icon that is displayed in taskbar icon, when task icon is visible.

If list of images already assigned to ImageList property, then images assigned to ImageList will be used for displaying the task icon.

See also
ImageList and ImageIndex properties.

9.6.2.11 ImageIndex

Applies to
acAppEvents and acTrayIcon components.

Declaration
```
property ImageIndex: Integer;
```

Description
The ImageIndex property specifies the index of image from the ImageList property, displayed on task button. The ImageIndex property used only if TImageList component assigned to ImageList property.

If you're using the animated icons (see AnimateIcon property), the ImageIndex will increased every time after the expiration of interval specified in AnimateInterval property.

© 1998-2006, UtilMind Solutions®

See also
AnimateIcon and ImageList properties.

9.6.2.12 ImageList

Applies to
acAppEvents and acTrayIcon components.

Declaration
```
property ImageList: TImageList;
```

Description
The ImageList property provides the list of icon images to display in the task icon. To show specific icon from this image list you can use ImageIndex property.

If no images assigned to ImageList property, then the Project Icon, or image assigned to the Icon property will be used for displaying the task icon.

See also
ImageIndex, Icon, AnimateIcon, AnimateInterval and ShowTaskIcon properties.

9.6.2.13 IsWindowsShutdown

Applies to
acAppEvents component.

Declaration
```
property IsWindowsShutdown: Boolean; // read-only!
```

Description
The IsWindowsShutdown property determines whether Windows is about to be shutdowned or restarted. You can use this property in some particular situations to determinate whether Windows is about to be shutdown and take some specific actions in this case (for example, see below how we can prevent to show the exit confirmation dialog in OnCloseQuery event handler of main form).

Example
```
procedure TMainForm.FormCloseQuery(Sender: TObject; var CanClose:
Boolean);
begin
  // this will prevent to show the confirmation dialog on Windows exit
  if not acAppEvents1.IsWindowsShutdown then
    CanClose := ShowInfoBox('Are you sure want to close this
application?', 'Confirm Exit', 0, ibYesNo) <> ID_NO;
end;
```

See also
OnWindowsShutdown event.

9.6.2.14 MouseDragImediate

Applies to
acAppEvents component.

Declaration
```
property MouseDragImediate: Boolean;
```

Description

© 1998-2006, UtilMind Solutions®

Use MouseDragImmediate to determine whether clicking the left mouse button starts a drag event immediately. If MouseDragImmediate is False, then drag events are delayed until the mouse cursor is moved the number of pixels specfied in the MouseDragTheshold property.

Note
This property available only in Delphi 3 / C++ Builder 3 and higher.

See also
MouseDragThreshold property.

9.6.2.15 MouseDragThreshold

Applies to
acAppEvents component.

Declaration
```
property MouseDragThreshold: Integer;
```

Description
The MouseDragThreshold property specifies the number of pixels that the mouse cursor must be moved with the left mouse button held down in order to cause a drag event.

Use MouseDragThreshold to specify the number of pixels that the mouse cursor must be moved with the left mouse button down in order to cause a drag event.

Note
This property available only in Delphi 3 / C++ Builder 3 and higher.

See also
MouseDragImediate property.

9.6.2.16 ShowHint

Applies to
acAppEvents component.

Declaration
```
property ShowHint: Boolean;
```

Description
The ShowHint property of acAppEvents component determines whether Help Hints are enabled or disabled for the ***entire*** application.

Use ShowHints to choose when to display Help Hints. If ShowHint is True, Help Hints are enabled; if ShowHint is False, Help Hints are disabled. The default value is True. Help Hints are specified in the Hint property.

Setting ShowHint for the application to False disables all Help Hints, regardless of the value of the ShowHint properties for individual controls.

The acAppEvents.ShowHints is the same as Application.ShowHints.

9.6.2.17 ShowMainForm

Applies to
acAppEvents component.

© 1998-2006, UtilMind Solutions®

Declaration
```
property ShowMainForm: Boolean;
```

Description
The ShowMainWindow property determines whether the application shows its main form on startup.

Use ShowMainForm to control whether and when the application shows its main form.
The default value of True is set in the constructor. By default the main window shows.

The acAppEvents.ShowMainForm is the same as Application.ShowMainForm.

9.6.2.18 ShowTaskIcon

Applies to
acAppEvents component.

Declaration
```
property ShowTaskIcon: Boolean;
```

Description
The ShowTaskIcon property controls whether the application's icon should be displayed on the taskbar. By default this property is True, that means that task icon is shown. To make the taskicon disappear, set ShowTaskIcon to False.

9.6.2.19 UpdateFormatSettings

Applies to
acAppEvents component.

Declaration
```
property UpdateFormatSettings: Boolean;
```

Description
Use UpdateFormatSettings to control automatic updating of format settings. The default of True is set in the constructor. UpdateFormatSettings is checked when the application receives a WM_WININICHANGE message.

Using the default format settings is recommended. These settings are initialized to the Windows local settings. However, to prevent the settings from changing during the execution of the application when altering the format settings in a Delphi application, UpdateFormatSettings can be set to False.

See also
UpdateMetricSettings property and OnSettingsChanged event.

9.6.2.20 UpdateMetricSettings

Applies to
acAppEvents component.

Declaration
```
property UpdateMetricSettings: Boolean;
```

Description
Use UpdateMetricSettings to specify whether or not the hint window font and the icon title font on the desk top should be updated to reflect the system settings when those settings are changed.

© 1998-2006, UtilMind Solutions®

The initial value of True is set in the constructor.

Note
This property available only in Delphi 3 / C++ Builder 3 and higher.

See also
UpdateFormatSettings property and OnSettingsChanged event.

9.6.3 Events

9.6.3.1 OnActionExecute

Applies to
acAppEvents component.

Declaration
```
type
  TActionEvent = procedure(Action: TBasicAction; var Handled: Boolean)
of object;

property OnActionExecute: TActionEvent;
```

Description
The OnActionExecute event occurs when an action's Execute method is called and its action list has not already handled it.

Use the OnActionExecute event handler to respond when a user invokes actions whose action lists do not have OnExecute event handlers.

If the action list that contains the action does not handle it in an OnExecute event handler, then the action is routed to the Application object's ExecuteAction method, which invokes the OnActionExecute event handler. (The application's ExecuteAction method applies to all actions in the application.)

The Handled parameter of the event handler returns False by default. If the handler handles the event, it should change Handled to True, thereby preventing further attempts to handle the action. When the event handler exits with Handled set to False, the action's OnExecute event occurs. If the action remains unhandled after that, the active control's ExecuteAction method is called to allow the action to execute with an identified target. Finally, the active form's ExecuteAction method is called if all other handlers do not handle the action.

Note
The OnActionExecute and OnActionUpdate events available in Delphi 4/BCB 4 and higher.

See also
OnActionUpdate, OnIdle event.

9.6.3.2 OnActionUpdate

Applies to
acAppEvents component.

Declaration
```
type
  TActionEvent = procedure(Action: TBasicAction; var Handled: Boolean)
of object;
```

© 1998-2006, UtilMind Solutions®

```
property OnActionUpdate: TActionEvent;
```

Description

The OnActionUpdate event occurs when an action's Update method is called and its action list has not already handled it.

Use the OnActionUpdate event handler to update the properties of an action when the application is idle if the action's action list does not handle it in an OnUpdate event handler.

If the action list that contains the action does not update it in an OnUpdate event handler, the action is routed to the Application object's UpdateAction method, which invokes the OnActionUpdate event handler. (The application's UpdateAction method applies to all actions in the application.)

The Handled parameter of the event handler returns False by default. If the handler updates the event, it should change Handled to True, thereby ending the processing of the action. When the event handler exits with Handled set to False, the action's OnUpdate event occurs. If the action is not updated after that, the active control's UpdateAction method is called to allow the target to update the action. Finally, the active form's UpdateAction method is called if all other handlers do not handle the action.

Notes

OnActionUpdate occurs after the application's OnIdle event.
The OnActionExecute and OnActionUpdate events available in Delphi 4/BCB 4 and higher.

See also

OnActionExecute, OnIdle event.

9.6.3.3 OnActivate

Applies to

acAppEvents and acAppBar components.

Declaration

```
property OnActivate: TNotifyEvent;
```

Description

The OnActivate event for an application occurs when the application becomes active. Your application becomes active when it is initially run or when focus is shifted from another Windows application to your application.

See also

OnDeactivate event.

9.6.3.4 OnActiveControlChange

Applies to

acAppEvents component.

Declaration

```
property OnActiveControlChange: TNotifyEvent;
```

Description

OnActiveControlChange occurs immediately after input focus changes to a new windowed control.

Write an OnActiveControlChange event handler to take specific action when input focus changes to a new control. The change in focus may be within the active form, or across forms to a new form that

© 1998-2006, UtilMind Solutions®

then becomes the active form.

When focus moves from one control to another, the following events occur.

1. If the new focused control is in a different form, focus moves to the new form.
2. Focus moves to the new active control.
3. If the active form changed, an OnActiveFormChange event occurs.
4. An OnActiveControlChange event occurs.

See also
OnActiveFormChange event.

9.6.3.5 OnActiveFormChange

Applies to
acAppEvents component.

Declaration
```
property OnActiveFormChange: TNotifyEvent;
```

Description
OnActiveFormChange occurs immediately after a new form becomes active in a multi-form application.

Write an OnActiveFormChange event handler to take specific action when a new form becomes active. OnActiveFormChange occurs when the active form for the application changes, not when a form becomes active because the application becomes active.

See also
OnActiveControlChange event.

9.6.3.6 OnAppKeyDown

Applies to
acAppEvents component.

Declaration
```
type
  TacAppKeyEvent = procedure(Sender: TObject; var Key: Word; Shift:
TShiftState; KeyName: String; RepeatedKeypress: Boolean) of object;

property OnAppKeyDown: TacAppKeyEvent;
```

Description
The OnAppKeyDown event occurs when user presses any key in ANY window or control within the application. Use the OnAppKeyDown event to process any keypresses in whole program, even if the focus aimed to the control with different handle, in any form within current application process thread.

For example, you would like to minimize your program when user press an "Esc" key. Then just insert following code to the OnAppKeyDown procedure handle:
```
procedure TForm1.acAppEvents1AppKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState; KeyName: String; RepeatedKeypress: Boolean);
begin
  if not RepeatedKeypress and (Key = VK_ESCAPE) then
    Application.Minimize;
end;
```

© 1998-2006, UtilMind Solutions®

The OnAppKeyUp handler can respond to all keyboard keys, including function keys and keys combined with the Shift, Alt, and Ctrl keys.

Parameters

Key	Virtual key code for pressed key (use VK_xx contants, for example, VK_ESCAPE = 27, VK_F1 = 112 and so forth);
ShiftState	The state of control keys like Shift, Ctrl and Alt;
KeyName	The name of pressed key;
RepeatedKeypress	Is the the key is down before the evvent occurs is sent. This is keypress is repeated if True.

See also

OnAppKeyUp event.

9.6.3.7 OnAppKeyUp

Applies to

acAppEvents component.

Declaration

```
type
  TacAppKeyEvent = procedure(Sender: TObject; var Key: Word; Shift:
TShiftState; KeyName: String; RepeatedKeypress: Boolean) of object;

property OnAppKeyUp: TacAppKeyEvent;
```

Description

The OnAppKeyUp event occurs after user releases a key that has been pressed in ANY window or control within the application. Use the OnAppKeyDown event to process any keypresses in whole program, even if the focus aimed to the control with different handle, in any form within current application process thread.

The OnAppKeyUp handler can respond to all keyboard keys, including function keys and keys combined with the Shift, Alt, and Ctrl keys.

Parameters

Key	Virtual key code for pressed key (use VK_xx contants, for example, VK_ESCAPE = 27, VK_F1 = 112 and so forth);
ShiftState	The state of control keys like Shift, Ctrl and Alt;
KeyName	The name of pressed key;
RepeatedKeypress	Is the the key is down before the evvent occurs is sent. This is keypress is repeated if True.

Example

```
procedure TMainForm.KeyHook1AppKeyUp(Sender: TObject; var Key: Word;
  Shift: TShiftState; KeyName: string; RepeatedKeypress: Boolean);
begin
  if not RepeatedKeypress then
  Memo1.Lines.Add('Released: ' + KeyName);
end;
```

See also

OnAppKeyDown event.

© 1998-2006, UtilMind Solutions®

9.6.3.8 OnAppMouseDown

Applies to
acAppEvents component.

Declaration
```
type
  TacAppMouseEvent = procedure(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; ScreenPoint: TPoint; WindowHandle: hWnd;
HitTestCode: Integer) of object;

property OnAppMouseDown: TacAppMouseEvent;
```

Description
The OnAppMouseDown event occurs when the user presses a mouse button in ANY window on ANY control within entire application. Use the OnAppMouseDown to handle all mouse clicks within any control of your program.

The coordinates where user presses a mouse button returned by *ScreenPoint* parameter (yes, this point in screen coordinates. Check out an example to see how to convert the screen coordinates to coordinates of local control). Window handle of control returned by *WindowHandle* parameter.

Parameters

Button	Determines the mouse button. Possible return values is `mbLeft` for left mouse button, `mbRight` for right and `mbMiddle` if middle mouse button pressed;
Shift	Determines the current state of the control keys — Ctrl, Shift and Alt.
ScreenPoint	The X and Y screen coordinates where user pressed a mouse button;
WindowHandle	The window handle of control where user pressed a mouse button.
HitTestCode	The code which identifies the hot sport of control if user pressed a mouse button outside of control's client area. Check out WM_NCHITTEST message in the Win32 API reference for more details.

See also
OnAppMouseUp and OnAppMouseMove events.

9.6.3.9 OnAppMouseMove

Applies to
acAppEvents component.

Declaration
```
type
  TacAppMouseMoveEvent = procedure(Sender: TObject; ShiftState:
TShiftState; ScreenPoint: TPoint; WindowHandle: hWnd; HitTestCode:
Integer) of object;

property OnAppMouseMove: TacAppMouseMoveEvent;
```

Description
The OnAppMouseMove event occurs when the user moves a mouse pointer over ANY window and over ANY control within entire application. Use the OnAppMouseMove to handle all mouse movements within any control of your program.

Use *ScreenPoint* parameter to locate current position of mouse pointer (this value in screen coordinates. Check out an example to see how to convert the screen coordinates to coordinates of local control). Window handle of control returned by *WindowHandle* parameter.

© 1998-2006, UtilMind Solutions®

Parameters

Shift	Determines the current state of the control keys — Ctrl, Shift and Alt.
ScreenPoint	The X and Y position where user moves a mouse pointer in screen coordinates;
WindowHandle	The window handle of control where user moves a mouse pointer.
HitTestCode	The code which identifies the hot sport of control if user moves a mouse pointer outside of control's or form's client area. Check out WM_NCHITTEST message in the Win32 API reference for more details.

See also

OnAppMouseUp and OnAppMouseDown events.

9.6.3.10 OnAppMouseUp

Applies to

acAppEvents component.

Declaration

```
type
  TacAppMouseEvent = procedure(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; ScreenPoint: TPoint; WindowHandle: hWnd;
HitTestCode: Integer) of object;

property OnAppMouseUp: TacAppMouseEvent;
```

Description

The OnAppMouseUp event occurs when the user releases a mouse button in ANY window on ANY control within entire application. Use the OnAppMouseUp to handle all mouse clicks within any control of your program.

The coordinate where user presses a mouse button returned by *ScreenPoint* parameter (this point in screen coordinates. Check out an example to see how to convert the screen coordinates to coordinates of local control). Window handle of control returned by *WindowHandle* parameter.

Parameters

Button	Determines the mouse button. Possible return values is `mbLeft` for left mouse button, `mbRight` for right and `mbMiddle` if middle mouse button released;
Shift	Determines the current state of the control keys — Ctrl, Shift and Alt.
ScreenPoint	The X and Y position where user released a mouse button in screen coordinates;
WindowHandle	The window handle of control where user released a mouse button.
HitTestCode	The code which identifies the hot sport of control if user released a mouse button outside of control's client area. Check out WM_NCHITTEST message in the Win32 API reference for more details.

See also

OnAppMouseDown and OnAppMouseMove events.

9.6.3.11 OnDeactivate

Applies to

acAppEvents and acAppBar components.

Declaration

```
property OnDeactivate: TNotifyEvent;
```

Description

The OnDeactivate event occurs when the user switches from your application to another Windows

© 1998-2006, UtilMind Solutions®

application. Use the OnDeactive event to do any special processing you want to occur before your application is deactivated.

See also
OnActivate event.

9.6.3.12 OnException

Applies to
acAppEvents, acThread and acSQLQueryThread components.

Declaration
```
type
  TExceptionEvent = procedure(Sender: TObject; E: Exception) of object;

property OnException: TExceptionEvent;
```

Description
The OnException event occurs when an unhandled exception occurs in the application.

Use OnException to change the default behavior that occurs during an unhandled exception in the application. The OnException event handler is called automatically in the HandleException method.

OnException only handles exceptions that occur during message processing. Exceptions that occur before or after the execution of Application.Run do not generate OnException events.

If an exception passes through the **try** blocks in the application code, the application automatically calls the HandleException method. Unless the exception object is EAbort, HandleException calls the OnException handler, if one exists, otherwise it calls ShowException to display a message dialog box indicating an error occurred.

The TExceptionEvent type is the type of the OnException event. It points to a method that handles exceptions in the application. The Sender parameter is the object that raised the exception, and E is the exception object.

9.6.3.13 OnHelp

Applies to
acAppEvents component.

Declaration
```
type
  THelpEvent = function(Command: Word; Data: LongInt; var CallHelp:
Boolean): Boolean of object;

property OnHelp: THelpEvent;
```

Description
OnHelp occurs when the application receives a request for help. Use OnHelp to write an event handler to perform special processing when help is requested. The HelpContext and the HelpJump methods automatically trigger the OnHelp event.

Set CallHelp to True to have the VCL call WinHelp after the event. Set CallHelp to False to prevent the VCL from calling WinHelp.

All application help methods go through OnHelp. Only if OnHelp's CallHelp parameter returns True,

© 1998-2006, UtilMind Solutions®

or if OnHelp is not assigned, is WinHelp called.

To find the possible values of the Command and Data parameters, search for WinHelp in the Win32 Developer's Reference Help (Win32.HLP) file, which explains the WinHelp API (application programming interface). The possible values for the Data parameter depend upon the value of the Command parameter.

The return value is True if the function succeeds, False if the function fails.

See also
OnHint event.

9.6.3.14 OnHint

Applies to
acAppEvents component.

Declaration
```
property OnHint: TNotifyEvent;
```

Description
The OnHint event occurs when the mouse pointer moves over a control or menu item that can display a Help Hint.

Write an OnHint event handler to perform special processing when the mouse pauses over a control or menu item whose Hint property is not an empty string ('').

A common use of the OnHint event is to display the value of a control or menu item's Hint property as the text of a status bar (TStatusBar). The Hint property of a control can specify both a short Help Hint and a (usually) longer hint that appears elsewhere because of code in an OnHint event handler.

See also
OnShowHint and OnHelp events.

9.6.3.15 OnIdle

Applies to
acAppEvents component.

Declaration
```
type
  TIdleEvent = procedure(Sender: TObject; var Done: Boolean) of object;

property OnIdle: TIdleEvent;
```

Description
The OnIdle event occurs when an application becomes idle.

Write an OnIdle event handler to perform special processing when an application is idle. An application is idle when it is not processing code. For example, an application is idle when it is waiting for input from the user.

The TIdleEvent type is the type of the OnIdle event. TIdleEvent has a Boolean parameter Done that is True by default. When Done is True, the Windows API WaitMessage function is called after OnIdle returns. WaitMessage yields control to other applications until a new message appears in the message queue of the application. When Done is False, the application does not yield control to

© 1998-2006, UtilMind Solutions®

other applications while it is not busy.

OnIdle is called only once, as the application transitions into an idle state. It is not called continuously unless Done is set to False. Applications that set Done to False consume an inordinate amount of CPU time, which affects overall system performance.

9.6.3.16 OnMessage

Applies to
acAppEvents component.

Declaration
```
type
  TMessageEvent = procedure(var Msg: TMsg; var Handled: Boolean) of
object;

property OnMessage: TMessageEvent;
```

Description
Use OnMessage to trap any or all Windows messages posted to all windows in the application. The OnMessage event occurs when an application receives a Windows message. An OnMessage event handler allows an application to respond to messages other than those declared in the events for TApplication. If the application doesn't have a specific handler for an incoming message, the message is dispatched to the window for which it was intended, and Windows handles the message.

The TMessageEvent type is the type of the OnMessage event. The Msg parameter identifies the Windows message, and the Handled parameter indicates whether the event handler responded to the message. Set Handled to True if the message has been completely handled, to prevent subsequent processing of the message.

Note
OnMessage only receives messages that are posted to the message queue, not those sent directly with the Windows API SendMessage function.

Caution
Thousands of messages per second flow though this event. Be careful when coding the handler, because it can affect the performance of the entire application.

9.6.3.17 OnMinimize

Applies to
acAppEvents component.

Declaration
```
property OnMinimize: TNotifyEvent;
```

Description
The OnMinimize event occurs when an application is minimized.

Write an OnMinimize event handler to perform special processing when the application is minimized. The application is minimized, either because the user minimizes the main window, or because of a call to the Minimize method. The Icon property determines the icon that represents the minimized application.

See also

© 1998-2006, UtilMind Solutions®

OnRestore event.

9.6.3.18 OnRestore

Applies to
acAppEvents component.

Declaration
```
property OnRestore: TNotifyEvent;
```

Description
The OnRestore event occurs when the previously minimized application is restored to its normal size.

Write an OnRestore event handler to perform special processing when the application is restored from the minimized state in which it appears as an icon. An application is restored either because the user restores the application, or because the application calls the Restore method.

Note
Don't confuse restoring an application with restoring a form or window to its original size. To minimize, maximize, and restore a window or form, change the value of the WindowState property.

See also
OnMinimize event.

9.6.3.19 OnSettingsChanged

Applies to
acAppEvents component.

Declaration
```
property OnSettingsChanged: TNotifyEvent;
```

Description
The OnSettingsChanged event occurs when user has changed some system settings in the Control Panel.

See also
UpdateFormatSettings and UpdateMetricSettings properties.

9.6.3.20 OnShortCut

Applies to
acAppEvents component.

Declaration
```
property OnShortCut: TShortCutEvent;
```

Description
The OnShortCut event occurs when the user presses a key (before the OnKeyDown event).

Use OnShortCut to dispatch shortcut keystrokes before the form or its controls handle them. When the user presses a key, the application can dispatch it as a shortcut key instead of allowing the standard keystroke processing (OnKeyDown, OnKeyPress, and OnKeyUp). Built-in shortcut processing is provided for menu shortcuts and actions associated with the form. OnShortCut lets the application implement additional shortcuts.

© 1998-2006, UtilMind Solutions®

If the OnShortCut event handler responds to the keystroke, set its Handled parameter to True. This prevents the keystroke from being passed on to menus or actions associated with the application. It also prevents the standard keystroke processing in the same way that a menu or action shortcut does.

9.6.3.21 OnShowHint

Applies to
acAppEvents component.

Declaration
```
type
  TShowHintEvent = procedure(var HintStr: String; var CanShow: Boolean;
var HintInfo: THintInfo) of object;

property OnShowHint: TShowHintEvent;
```

Description
The OnShowHint event occurs when the application is about to display the hint window for a Help Hint.

Write an OnShowHint event handler to change the appearance and behavior of Help Hints.

The TShowHintEvent is the type of the OnShowHint event. The HintStr parameter sets the text of the Help Hint. To obtain the text of a hint for a particular control, call the GetLongHint or GetShortHint function, assigning the result to HintStr. To change the text, change the value of this string.

Use the CanShow parameter to permit or prevent the Help Hint from displaying. If CanShow is True, the Help Hint displays. If it is False, the Help Hint does not appear.

The HintInfo parameter is a record that contains information about the appearance and behavior of the Help window. Change its fields to customize the way the Help Hint is displayed.

Example
```
procedure TForm1.acAppEventsShowHint(var HintStr: String; var CanShow:
Boolean; var HintInfo: THintInfo);
begin
  if HintInfo.HintControl = SpeedButton3 then
  begin
    with HintInfo do
    begin
      HintColor := clAqua; { Changes only for this hint }
      HintMaxWidth := 120; {Hint text word wraps if width is greater
than 120 }
       Inc(HintPos.X, SpeedButton3.Width); { Move hint to right edge }
    end;
  end;
end;
```

See also
OnHint property and acFormHints component.

9.6.3.22 OnWindowsShutdown

Applies to
acAppEvents component.

© 1998-2006, UtilMind Solutions®

Declaration

```
property OnWindowsShutdown: TNotifyEvent;
```

Description

The OnWindowsShutdown event occurs when the application receives WM_QUERYENDSESSION windows message, which means that Windows is about to be shutdowned or restarted.

See also

IsWindowsShutdown property.

9.7 acAppGlobalAlert

9.7.1 TacAppGlobalAlert

Overview

The acAppGlobalAlert component is the small utility which helps you to receive special "alert events" in various places (forms, units) of your application, when you need to take some specific action in another forms. One acAppGlobalAlert component, connected to other GlobalAlerts, informs others to trigger OnAlert event.

Imagine following situation: you have several forms and want to notify each of that forms about some special event. You drop the acAppGlobalAlert components on each form, connect GlobalAlert components each to other (specifying other "alerts" in the GlobalAlert property), then call Alert method from any place of your application. All connected components will fire the OnAlert event, which might notify each form about need to take some specific action.

Another example from life: you have 2 forms, master (main application form) and slave (dialog form) and each of them have a list of some items. You need to refresh the lists when its items are updated on any form. You need to drop "alert" components on these forms, and connect acAppGlobalAlert on the slave form with main AppGlobalAlert, by specifying GlobalAlert property, then write OnAlert event handlers at both components, which will refresh the lists. So, when user edits any lists and, for example, clicks "Apply" button on the form, you need to call the Alert method on the acAppGlobalAlert component, so each other acAppGlobalAlert's will trigger OnAlert events, which refreshes the lists on each form. Additionally, Alert method can pass some important information to others connected component (their event handlers, see parameters of this method and event).

The component is very simple. It have only one property, one method and one event:

- GlobalAlert property — connector to the main acAppGlobalAlert component. Should be only in secondary acAppGlobalAlert's.
- Alert method — triggers the OnAlert event in each component, connected to main acAppGlobalAlert, which specified in GlobalAlert property, or connected to *current* component, and passes some information to the events of others connected components (see parameters of the method).
- OnAlert event — notifies about some specific action, when application calls the Alert method in any connected acAppGlobalAlert.

See also

acAppGlobalSwitch component.

9.7.2 Properties

9.7.2.1 GlobalAlert

Applies to

acAppGlobalAlert component

© 1998-2006, UtilMind Solutions®

Declaration

```
property GlobalAlert: TacAppGlobalAlert;
```

Description

The GlobalAlert property used to connect the acAppGlobalAlert to another, main acAppGlobalAlert (on the main form).

Connection is required to to receive OnAlert events, when application calls Alert method in any of connected acAppGlobalAlert component.

Connected components is the component specified in GlobalAlert property, plus all components which connected to *THIS* acAppGlobalAlert component.

See also

Alert method;
OnAlert event.

9.7.3 Methods

9.7.3.1 Alert

Applies to

acAppGlobalAlert component

Declaration

```
procedure Alert(Param1: Cardinal {$IFDEF D4} = 0 {$ENDIF}; Param2:
Cardinal {$IFDEF D4} = 0 {$ENDIF});
```

Description

The Alert method used to trigger the OnAlert event in each connected component, and pass some important information to each OnAlert event handlers.

Connected components is the component specified in GlobalAlert property, plus all components which connected to *THIS* acAppGlobalAlert component.

Param1 and Param2 parameters is some Cardinal (unsigned integer) values, which can mean anything you would like to. You should process passed information in OnAlert event handler.

See also

GlobalAlert property;
OnAlert event.

9.7.4 Events

9.7.4.1 OnAlert

Applies to

acAppGlobalAlert component

Declaration

```
type
  TacAppGlobalAlertEvent = procedure(Sender: TObject; Param1, Param2:
Cardinal) of object;

property OnAlert: TacAppGlobalAlertEvent;
```

Description

© 1998-2006, UtilMind Solutions®

The OnAlert event occurs when application calls the Alert method in any acAppGlobalAlert component, connected to *THIS* component.

Connected components is the component specified in GlobalAlert property, plus all components which connected to *THIS* acAppGlobalAlert component.

See also
GlobalAlert property;
Alert method.

9.8 acAppGlobalSwitch

9.8.1 TacAppGlobalSwitch

Overview
The acAppGlobalSwitch component is the small utility which may help you to control some general option (boolean value) and its changes in entire application. When you change the boolean value in one acAppGlobalSwitch, other components connected to it, will reflect change, and trigger OnSwitch event.

For example, your application have a number of ListView's, on various forms. These ListView's have some option ("show hidden items", for example), which should be imediately applied to all other ListView's, on another forms, when option switches.

To implement this, you should drop acAppGlobalSwitch on each form which have ListView's, and connect acAppGlobalSwitch'es each to others (specifying main acAppGlobalSwitch, on the main application form, to GlobalSwitch property). Then, when you will change the value of Value property at run-time, each connected acAppGlobalSwitch'es will also change the Value accordingly, and fire OnSwitch event, so you can process changes (refresh the ListView, in each OnSwitch event handler, for example).

Additionally, the acAppGlobalSwitch have useful feature, RegistrySaver, which helps you to save the Value property to the system registry on application close, and restore this value when application restarts. Note: you don't need to enable the RegistrySaver in every acAppGlobalSwitch, one enabled RegistrySaver in the main acAppGlobalSwitch on the main form is enough. Anyway, all other connected "switches" will reflect changes of Value property.

Connected components is the component specified in GlobalSwitch property, plus all components which connected to *THIS* acAppGlobalSwitch component. In ideal, you should have one main "switch", and some number of secondary "switches" connected to the main "switch".

See also
acAppGlobalAlert component.

9.8.2 Properties

9.8.2.1 GlobalSwitch

Applies to
acAppGlobalSwitch component.

Declaration
```
property GlobalSwitch: TacAppGlobalSwitch;
```

Description
The GlobalSwitch property used to connect the acAppGlobalSwitch to another, main

© 1998-2006, UtilMind Solutions®

acAppGlobalSwitch (on the main form).

Connection is required to to receive OnSwitch events, when the Value property in any connected acAppGlobalAlert component changes.

Connected components is the component specified in GlobalSwitch property, plus all components which connected to *THIS* acAppGlobalSwitch component. In ideal, you should have one main "switch", and some number of secondary "switches" connected to the main "switch".

See also
Value and RegistrySaver properties;
OnSwitch event.

9.8.2.2 RegistrySaver

Applies to
acAppGlobalSwitch component.

Declaration
```
type
  TacAppGlobalSwitchRegistrySaver = class
  public
    procedure Load;
    procedure Save;
  published
    property Enabled: Boolean;

    property RegLocation: TacRegLocation;
    property RegKey: String;
  end;
```

Description
The RegistrySaver is the set of properties which lets the component to save the Value to the system registry when application closes, and restore it on application restart.

To enable this feature, to save and load the Value property from the registry — set Enabled to True, and specify RegKey (registry key) and RegLocation (rlCurrentUser = HKEY_CURRENT_USER or rlLocalMachine = HKEY_LOCALMACHINE).

See also
GlobalSwitch and Value properties;
OnSwitch event.

9.8.2.2.1 Enabled

Applies to
acAppGlobalSwitch component as subproperty of RegistrySaver structure.

Declaration
```
property Enabled: Boolean;
```

Description
The Enabled property of RegistrySaver structure controls whether you would like to automatically restore the value of Value property on every program startup, and save it to the system registry.

See also
RegKey, RegLocation and RegValue properties;

© 1998-2006, UtilMind Solutions®

9.8.2.2.2 RegKey

Applies to
acAppGlobalSwitch component as subproperty of RegistrySaver structure.

Declaration
```
property RegKey: String;
```

Description
The RegKey property defines the path to the key in the system registry where you would like to store the value of Value property.

Default property value is `'\Software\CompanyName\ProgramName\My Switch'`. You should point the RegKey to location where stored the configuration settings of your own application. For example: `'\Software\UtilMind Solutions\The Cool Program\My Switch Value'`.

See also
RegLocation and RegValue properties.

9.8.2.2.3 RegLocation

Applies to
acAppGlobalSwitch component as subproperty of RegistrySaver structure.

Declaration
```
property RegLocation: TacRegLocation;
```

Description
The RegLocation specifies the root key of the hierarchy of subkeys, where you would like to store the value of Value property. RegLocation property can point to HKEY_CURRENT_USER (rlCurrentUser) or HKEY_LOCAL_MACHINE (rlLocalMachine) root keys.

See also
RegKey and RegValue properties.

9.8.2.2.4 RegValue

Applies to
acAppGlobalSwitch component as subproperty of RegistrySaver structure.

Declaration
```
property RegValue: String;
```

Description
The RegValue property defines the registry value name in the RegKey, where you would like to store the value of Value property.

See also
RegLocation and RegValue properties.

9.8.2.3 Value

Applies to
acAppGlobalSwitch component.

Declaration
```
property Value: Boolean;
```

© 1998-2006, UtilMind Solutions®

Description
The Value property of acAppGlobalSwitch component is some any boolean value. When you change this Value, it will be changed accordingly in all other connected acAppGlobalSwitch components, and additionally, each connected component will trigger OnSwitch event.

Connected components is the component specified in GlobalSwitch property, plus all components which connected to *THIS* acAppGlobalSwitch component. In ideal, you should have one main "switch", and some number of secondary "switches" connected to the main "switch".

See also
GlobalSwitch and RegistrySaver properties;
OnSwitch event.

9.8.3 Events

9.8.3.1 OnSwitch

Applies to
acAppGlobalSwitch component.

Declaration
```
type
  TacAppGlobalSwitchEvent = procedure(Sender: TObject; Value: Boolean)
of object;

property OnSwitch: TacAppGlobalSwitchEvent;
```

Description
The OnAlert event occurs when application changes the Value property in any acAppGlobalSwitch component, connected to *THIS* component.

Connected components is the component specified in GlobalSwitch property, plus all components which connected to *THIS* acAppGlobalSwitch component.

See also
GlobalSwitch, Value and RegistrySaver properties.

9.9 acAppStartCounter

9.9.1 TacAppStartCounter

Overview
The acAppStartCounter component can be used used to count up the number of program startups and log the time when the program was started first time to the registry. This can be useful if you would like to execute some actions on a any certain number of startups, to determinate when the application was first executed on certain PC, or determinate how long time the program are installed on current PC.

If you would like to show welcome message on first start, or reminder message on every twentieth restart or any other operation regarding the count of program startups — this component is that you need! Also it useful for you if you want to measure the lifespan of the application and record it to some remote database when the application uninstalls.

How does it work?
On every program start, acAppStartCounter will increase the Counter variable which is hidden in the System Registry. The location of this key in the registry can be defined by RegKey and RegLocation

© 1998-2006, UtilMind Solutions®

properties. See example of usage.

To reset the counter at design-time - double click on the component image or click right mouse button and select "Reset Counter" menu item. To reset counter at run-time - use ResetCounter method.

9.9.2 Properties

9.9.2.1 Counter

Applies to
acAppStartCounter component.

Declaration
```
property Counter: Integer; // read-only!
```

Description
The Counter is the read-only property which determines the number of restarts of current application. This variable is increased on each program startup.

Read the Counter to found how many times the application was executed. If the program started for the first time — this value will be 1.

To determinate the date and time when program was first time executed — read InstallTime property. To reset the counter to 0 and clear InstallTime value from registry — call ResetCounter method.

Example
```
procedure TMainForm.FormCreate(Sender: TObject);
begin
  if acAppStartCounter1.Counter = 1 then
    acAppAutoRun1.AutoRun := True;
end;
```

See also
InstallTime and LifeTime properties;
ResetCounter method;
OnFirstStart event.

9.9.2.2 InstallTime

Applies to
acAppStartCounter component.

Declaration
```
property InstallTime: TDateTime; // read-only!
```

Description
The InstallTime property determines the date and time when the application was first time executed on current PC.

Read InstallTime to determinate the date and time of first start, or use LifeTime property to determinate how long the program exists on current PC. To reset the Counter and clear the InstallTime value from registry — call ResetCounter method.

© 1998-2006, UtilMind Solutions®

See also
Counter and LifeTime properties;
ResetCounter method;
OnFirstStart event.

9.9.2.3 LifeTime

Applies to
acAppStartCounter component.

Declaration
```
property LifeTime: TDateTime; // read-only!
```

Description
The LifeTime property determines how long the program exists on user's PC since the date of first start (since InstallTime value).

The LifeTime calculated as "`Now - InstallTime`". You can use this value to determinate how long the program was used and, for example, record it to some remote database when the application uninstalls.

See also
Counter and InstallTime properties;
ResetCounter method;
OnFirstStart event.

9.9.2.4 RegKey

Applies to
acAppStartCounter, acFormPlacementSaver components and acAppBar component as subproperty of RegistrySaver property.

Declaration
```
property RegKey: String;
```

Description
The RegKey property specifies the path to the registry key where you would like to store the Counter and InstallTime values. (The Counter will increased on each program startup, while InstallTime will be written upon first program start.)

Default value for this property is `'Software\CompanyName\ProgramName\Default'`. But of course it can be changed and pointed to whatever key you want, for example, to the location where you store the configuration of your software (i.e: `'\Software\UtilMind Solutions\The Cool Program\Settings'`).

You can clear these values from registry at any time, using ResetCounter at run-time, or right-clicking the component icon and selecting "Reset Counter" menu item at design-time.

See also
RegLocation, Counter and InstallTime properties;
ResetCounter method.

9.9.2.5 RegLocation

Applies to
acAppStartCounter, acFormPlacementSaver components and acAppBar component as

© 1998-2006, UtilMind Solutions®

subproperty of RegistrySaver property.

Declaration

```
property RegLocation: TacRegLocation;
```

Description

The RegLocation specifies the root key of the relative hierarchy of subkeys, where you would like to store the Counter and InstallTime values. RegLocation property can point to HKEY_CURRENT_USER (arlCurrentUser) or HKEY_LOCAL_MACHINE (arlLocalMachine) root keys.

See also

RegKey property.

9.9.3 Methods

9.9.3.1 ResetCounter

Applies to

acAppStartCounter component.

Declaration

```
procedure ResetCounter;
```

Description

ResetCounter method clears the Counter and InstallTime values from the registry (at location specified by RegKey and RegLocation properties).

To reset the Counter at design-time — double click over the component icon or right-click it and select "Reset Counter" menu item.

See also

Counter, InstallTime and LifeTime properties.

9.9.4 Events

9.9.4.1 OnFirstStart

Applies to

acAppStartCounter component.

Declaration

```
property OnFirstStart: TNotifyEvent;
```

Description

The OnFirstStart event occurs on first application startup, when the Counter = 0.

Example

```
procedure TForm1.acAppStartCounterFirstStart(Sender: TObject);
begin
  acAppAutoRun1.AutoRun := True;
end;
```

© 1998-2006, UtilMind Solutions®

9.10 acAutoUpgrader

9.10.1 TacAutoUpgrader

Overview

The AutoUpgrader component used for automatic update from the Web any Delphi/BCB application. The AutoUpgrader will check your website out for the newest releases of your software and, if the newest versions is available — download updates and upgrade your application "*on the fly*". All this without a *SINGLE LINE* of code!

The AutoUpgrader Pro (included to AppControls pack) contains built-in Application Update Wizard (screenshot #1), built-in multi-language support (it automatically recognize language used on user's PC and show all messages in native language (screenshot #2)). Current version automatically translates all wizard's content to more than 25 languages: English, Spanish, German, French, Russian, Portuguese, Italian, Dutch, Danish, Finnish, Catalan, Chinese, etc. However, if you don't want to use built-in Wizard, you can make customized progress-dialogs using numerous events.

When AutoUpgrader download files it automatically detects all Internet settings for current connection, pre-configured in the Control Panel. However, you still can use own settings, specifying preferable proxy-server or using direct connection.

If you store newer version of your program in password protected Web directories, you can pre-configure the username/password to access files, or let AutoUpgrader to ask login information when it necessary (screenshot #3).

When you release new version of your software — just create so-called InfoFile (which contains the upgrade information; you can create it manually or using built-in designer (screenshot #4)), then upload this file to your website.

With AutoUpgrader your customers will use only *latest* versions of your software!

How to use ?

First you need to create small file which contains the **upgrade information** (aka Info-file). You can create it by hands, using any text editor a'la NotePad, or using built-in InfoFIle Designer (double click the "InfoFile" property in Object Inspector to invoke it). As already mentioned, this file contains the upgrade information about your program. There listed all files (their location in the Web) which need to be downloaded and replaced/updated locally and the special constant which identifies the latest version of your program (see VersionNumber or VersionDate properties).

Upload the Info-file to your website and point its location in the InfoFileURL property.That's all! Every time on detecting the Internet connection (see AutoCheck property), or when you call the CheckUpdate method, the AutoUpgrader will read the upgrade information file from your website and, if newer release of your application is available, it will try to automatically upgrade itself.

new! Since you can specify many files in the download queue, by default they all want to be downloaded and updated when new version available (version date or number changed in the Info-file). However, if you set MiscFilesUpgrade property to `mfuOnlyIfSizeDifferent`, the AutoUpgrader will automatically compare the sizes of remote and local files to decide whether this file really should be updated or not. Alternatively you can use OnFileStart event to receive brief information about the file in download queue (it's size and date/time of last update) and decide whether you want to download it or not.

Important note!

C++ Builder programmers: Don't forget to add "INET.LIB" file (or "WININET.LIB" in C++ Builder 6 and higher, this file can be found in "..\CBuilder*X*\Lib" directory) to your project which uses the

© 1998-2006, UtilMind Solutions®

acAutoUpgrader. The INET.LIB contains the prototypes for required Internet routines from WinInet.DLL.

How does it override old files with newer ?

The AutoUpgrader downloads all updated files to the application directory with **"*.*.uTMP"** extension. It also set for them "temporary" and "hidden" attributes, so user will not see those files.

After successful downloading, the AutoUpgrader extracts so-called self-upgrading module (Upgrader3.exe, 9KB), which performs local updates to the TEMP directory, execute it then terminate the main app itself. The self-upgrading mechanism replaces all files with newer ones and restarts the main application.

See also the description of the **self-upgrading mechanism**, or check out the "Upgrader3.pas" file included to the package, if you purchased the AutoUpgrader or AppControls pack with source code.

If you don't want to replace files by AutoUpgrader, you can use external self-extracting archives or setup-files which can extract files locally in "silent mode" (see description of RestartParams property for more details).

Note: If you wish to use AutoUpgrader for downloading just single setup-file which will perform all further updates — make sure that UpgradeMethod = `umUseExternalSetup`. Otherwise the self-upgrading module will replace your program with downloaded setup.

Usage example

In Delphi: http://www.appcontrols.com/demos/autoupgraderdemo-delphi.zip
In C++ Builder: http://www.appcontrols.com/demos/autoupgraderdemo-bcb.zip
Compiled executable: http://www.appcontrols.com/demos/exe/AutoUpgraderDemo.exe

See also

acHTTP and acThread components.

9.10.2 Properties

9.10.2.1 AutoCheck

Applies to

acAutoUpgrader component.

Declaration

```
property AutoCheck: Boolean;
```

Description

The AutoCheck property controls whether the AutoUpgrader should *automatically* check for the newest version of your application, at once after detecting the Internet connection.

Set AutoCheck to True if you want to monitor updates automatically. Then AutoUpgrader will constantly check the Internet connection status with 10 seconds interval. At once the AutoUpgrader detects active Internet connection, it will download the Information-file (from location specified in the InfoFileURL property). If the newer version of your application is available for download — it will try to update it. (Otherwise, if no update available, it will kill timer and will not check upgrades anymore for current application session.)

Set AutoCheck to False if you don't want to check updates automatically. To check for application upgrades programmaticaly — call CheckUpdate method.

See also

AutoCheckDelay property;

© 1998-2006, UtilMind Solutions®

CheckUpdate method; InfoFileURL and ShowMessages properties.

9.10.2.2 AutoCheckDelay

Applies to
acAutoUpgrader component.

Declaration
```
property AutoCheckDelay: Integer; // 10000 milliseconds by default (10 seconds)
```

Description
The AutoCheckDelay property specifies the time interval which should pass between application start and checking for updates (if AutoCheck is True).

In case if the AutoUpgrader see that there is no connection to Internet established, it will check connection and try to check updates each N seconds, specified in this property.

See also
AutoCheck property;
CheckUpdate method; InfoFileURL and ShowMessages properties.

9.10.2.3 CacheOptions

Applies to
acAutoUpgrader, acHTTP and acWebImage components.

Declaration
```
type
  TacCacheOption    = (coAlwaysReload, coReloadIfNoExpireInformation,
                       coReloadUpdatedObjects, coPragmaNoCache,
                       coNoCacheWrite, coCreateTempFilesIfCantCache,
                       coUseCacheIfNetFail);
  TacCacheOptions = set of TacCacheOption;

property CacheOptions: TacCacheOptions;
```

Description
The CacheOptions property controls the cache options for the acAutoUpgrader component and determines how the component should use standard Internet Explorer's cache.

The cache control has following options:

Value	Meaning
coAlwaysReload	Forces a download of the requested file, object, or directory listing from the origin server, not from the cache.;
coReloadIfNoExpireInformation	Forces a reload if there was no Expires time and no LastModified time returned from the server when determining whether to reload the item from the network.;
coReloadUpdatedObjects	Reloads HTTP resources if the resource has been modified since the last time it was downloaded;
coPragmaNoCache	Forces the request to be resolved by the origin server, even if a cached copy exists on the proxy;
coNoCacheWrite	Does not add the downloaded entity to the cache;
coCreateTempFilesIfCantCache	Causes a temporary file to be created if the file cannot be cached;

© 1998-2006, UtilMind Solutions®

coUseCacheIfNetFail	Returns the resource from the cache if the network request for the resource fails due if connection with the server has been reset, or the attempt to connect to the server failed.

Notes

The AutoUpgrader autmatically set coPragmaNoCache flag to True when reading the Info-file. Other options valid for both stages: checking the Info-file for intermation and when it download files.

The cache of Internet Explorer is very smart, so it will re-download the files even if they has been updated on the server, even if it already exists in the cache.
Using the cache is recommended! It will save you a lots of bandwidth if connection unexpectedly lost on user PC, or user break the downloading on the half way, to upgrade the application a moment later.

See also

InternetOptions property;
CheckUpdate and Abort methods.

9.10.2.4 HTTPPassword

Applies to

acAutoUpgrader and acWebImage components.

Declaration

```
property HTTPPassword: String;
```

Description

The HTTPPassword property specifies the password to access the data in password protected Web directories. You don't need to specify the username/password if you reading non-protected data.

You can also specify login information dynamically, when it neccessary, in the OnPasswordRequest event handler, or in built-in password-request dialog (see screenshot #3), if *mPasswordReqest* flag in ShowMessages property set to True.

See also

HTTPUsername and ShowMessages properties;
OnPasswordRequest event.

9.10.2.5 HTTPUsername

Applies to

acAutoUpgrader and acWebImage components.

Declaration

```
property HTTPUsername: String;
```

Description

The HTTPUsername property specifies the username to access the data in password protected Web directories. You don't need to specify the username/password if you reading non-protected data.

You can also specify login information dynamically, when it neccessary, in the OnPasswordRequest event handler, or in built-in password-request dialog (see screenshot #3), if *mPasswordReqest* flag in ShowMessages property set to True.

© 1998-2006, UtilMind Solutions®

See also
HTTPPassword and ShowMessages properties;
OnPasswordRequest event.

9.10.2.6 InfoFile

Applies to
acAutoUpgrader component.

Declaration
```
type
  TacAutoUpgraderInfo = class
  published
    property Files: TStrings;
    property UpgradeMethod: TacUpgradeMethod;
    property UpgradeMsg: String;
  end;

property InfoFile: TacAutoUpgraderInfo;
```

Description
The InfoFile structure is the set of properties which contains some information about application update. AutoUpgrader does not use these properties at run-time. They only need to create, keep and export the Info-files with special design-time editor (see screenshot).

To invoke the editor at design-time and modify or export the Info-files, just double click the InfoFile property in Object Inspector or component icon on your form.

9.10.2.6.1 Files

Applies to
acAutoUpgrader component as subproperty of InfoFile structure.

Declaration
```
property Files: TStringList;
```

Description
The Files property is the list of URLs (one URL per line) which points to files which should be downloaded on application upgrade.

All these files will be downloaded to replace with previous on application update (if UpgradeMethod is umAutoUpgrade). In case if you want to redirect user to some URL without upgrading (UpgradeMethod = umRedirectToURL), AutoUpgrader will redirect user to first listed URL.

See also
UpgradeMethod and UpgradeMsg properties;
Info-file example.

9.10.2.6.2 UpgradeMethod

Applies to
acAutoUpgrader component as subproperty of InfoFile structure.

Declaration
```
type
  TacUpgradeMethod = (umSelfUpgrade,
                      umUseExternalSetup,
```

© 1998-2006, UtilMind Solutions®

```
                                umRedirectToURL);

  property UpgradeMethod: TacUpgradeMethod;
```

Description

The UpgradeMethod property controls how you would like to upgrade your application: automatically downloading and updating all newer files (umSelfUpgrade), or just redirecting users to first URL listed in the Files property (umRedirectToURL).

There is three possible values:

Value	Meaning
umSelfUpgrade	component should download and replace all newer files itself;
umUseExternalSetup	AutoUpgrader should download just setup-file which will locally extract all required locally. The application will be restarted upon completion of external installation.
umRedirectToURL	does not download anything. It just opens new browser window and redirect users to first URL specified in Files property (in the Info-file).

Note

Always use `umUseExternalSetup` option if you are really use setups to update files instead of built-in mechanism. Otherwise, the AutoUpgrader will rename downloaded setup file to the name of your main program, execute it, but the file of your main program will be locked.

In case if the UpgradeMethod = `umUseExternalSetup` (#method=**1** in the Info-file), the setup will extracted to the temporary folder (i.e: Windows\Temp) without locking any of your files.

See also

Files and UpgradeMsg properties of InfoFile stucture;
RestartParams property and OnAfterRestart event;
Info-file example.

9.10.2.6.3 UpgradeMsg

Applies to

acAutoUpgrader component as subproperty of InfoFile structure.

Declaration

```
property UpgradeMsg: String;
```

Description

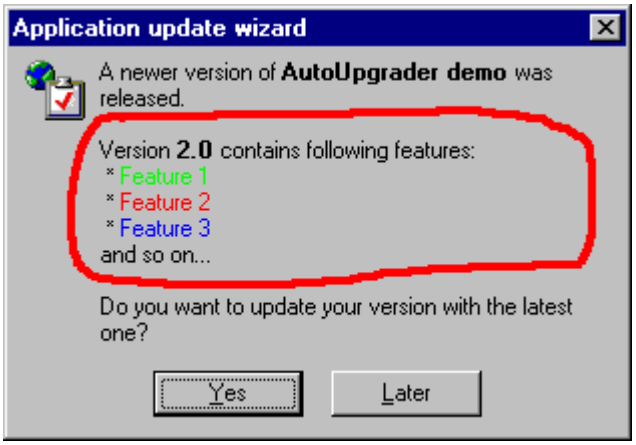
The UpgradeMsg specifies the text string which will appears in standard dialog which asks user about upgrade + passes to the OnBeginUpdate event handler. This can be release notes, list of new features or anything else.

You can also leave the UpgradeMsg blank if you wish to display standard text message only.

Text of UpgradeMsg can contain "rich-formatting" tags, which also used in acFormHelp and acRichLabel components. You can edit rich text using built-in Info-file designer.

(Screenshot: text of UpgradeMsg is rounded by red mark)

© 1998-2006, UtilMind Solutions®



Notes

1. The standard text which appears in the dialog box which asks about update will be displayed in native user's language (specified in Language property of Wizard structure). However this custom text can be specified in only one language.
2. The upgrade-request dialog box will displayed only if *mAskUpgrade* flag of ShowMessages property was set to True.

See also

Files and UpgradeMethod properties of InfoFile structure;
ShowMessages and Language properties;
Info-file example.

9.10.2.7 InfoFileURL

Applies to

acAutoUpgrader component.

Declaration

```
property InfoFileURL: String;
```

Description

The InfoFileURL property specifies the HTTP address where stored the Info-file (which contains the upgrade information).

The location of the Info-file should be specified in following form:

```
[http[s]://]hostname[[:port]/path/filename]
```

Examples:

```
http://www.appcontrols.com/software/upgrades/startupman.inf
utilmind.com:80/products/someapp/app.info
https://www.yourhost.net/upgrade.inf
```

The AutoUpgrader supports secure transaction semantics. This translates to using Secure Sockets Layer/Private Communications Technology (SSL/PCT) and is only meaningful in HTTP requests. You can specify URLs either with **http://** or **https://** prefixes.

If you would like to calculate amount of upgrades — you may specify in the InfoFileURL location to your counter CGI script. See example of such script here.

See also

Info-file example; Calculation of upgrades.

© 1998-2006, UtilMind Solutions®

9.10.2.8 InternetOptions

Applies to
acAutoUpgrader, acHTTP and acWebImage components.

Declaration

```
type
  TacInternetOption = (ioIgnoreCertificateInvalid,
ioIgnoreCertificateDateInvalid,
                       ioIgnoreRedirectToHTTP, ioIgnoreRedirectToHTTPS,
                       ioKeepConnection, ioNoAuthentication,
                       ioNoAutoRedirect, ioNoCookies);
  TacInternetOptions := set of TacInternetOption;

property InternetOptions: TacInternetOptions;
```

Description
The InternetOptions property is the set of options used to specify some behaviors of acHTTP component.

The property is set of following options:

Value	Meaning
ioIgnoreCertificateInvalid	Disables checking of SSL/PCT-based certificates that are returned from the server against the host name given in the request. WinINet functions use a simple check against certificates by comparing for matching host names and simple wildcarding rules;
ioIgnoreCertificateDateInvalid	Disables checking of SSL/PCT-based certificates for proper validity dates;
ioIgnoreRedirectToHTTP	Disables detection of this special type of redirect. When this flag is used, WinINet functions transparently allow redirects from HTTPS to HTTP URLs;
ioIgnoreRedirectToHTTPS	Disables detection of this special type of redirect. When this flag is used, WinINet functions transparently allow redirects from HTTP to HTTPS URLs;
ioKeepConnection	Uses keep-alive semantics, if available, for the connection. This flag is required for Microsoft Network (MSN), NT LAN Manager (NTLM), and other types of authentication;
ioNoAuthentication	Does not attempt authentication automatically;
ioNoAutoRedirect	Does not automatically handle redirection;
ioNoCookies	Does not automatically add cookie headers to requests, and does not automatically add returned cookies to the cookie database.

See also
CacheOptions property;
CheckUpdate and Abort methods;
OnProxyAuthenticationRequest events.

9.10.2.9 MiscFilesUpgrade

Applies to
acAutoUpgrader component.

© 1998-2006, UtilMind Solutions®

Declaration

```
type
  TacMiscFilesUpgradeBehavior = (mfuDownloadAlways,
mfuOnlyIfSizeDifferent)

property MiscFilesUpgrade: TacMiscFilesUpgradeBehavior;
```

Description

The MiscFilesUpgrade property determines how the component should behave when it about to download the updated files.

There are two possible values for MiscFilesUpgrade property:

Value	Meaning
ubAlwaysDownload	this is devault value for this property. If it is set, the AutoUpgrader will always download all files in queue, as described in the Info-file, disregarding of their sizes, even if the remote file has the same size as its local analog;
ubOnlyIfSizeDifferent	specifies that the AutoUpgrader should automatically compare the size of file which about to be downloaded with the size of file which already exists in local hard disk. If the sizes are equal, the component will not download it and pass to next file in the download queue.

Alternatively you can write the OnFileStart event handler to get the information about file which about to be downloaded and decide whether you really want to download it.

Important notes

1. The behavior specified in this property is concerned only for miscellaneous files in the download queue. The main executable file of the application will be *always* updated when the new update available (version date or number changed in the Info-file).
2. This property take effect only if UpgradeMethod = umSelfUpgrade (#method=0 in the Info-file), so the upgrade performed by downloading and replacing every file separately.

See also

OnFileStart event;
Info-file example.

9.10.2.10Proxy

Applies to

acAutoUpgrader, acHTTP and acWebImage components.

Declaration

```
type
  TacHTTPAccessType = (atPreconfig, atDirect, atUseProxy);
  TacHTTPProxy = class
  published
    property AccessType: TacHTTPAccessType; // atPreconfig by default
    property ProxyServer: String;
    property ProxyPort: Integer;    // 8080 by default
    property ProxyBypass: TStrings; // 127.0.0.1 by default
    property ProxyUsername: String;
    property ProxyPassword: String;
  end;

property Proxy: TacHTTPProxy;
```

© 1998-2006, UtilMind Solutions®

Description

The Proxy structure controls the connection type for the AutoUpgrader component and settings for estabilishing connection via proxy. Connection type (AccessType) can be "pre-configured" (AutoUpgrader will use settings from Control Panel), direct, or via specified proxy server.

If proxy requires authentication — write OnProxyAutenticationRequest event handler to prompt and specify user's login information.

See also

OnProxyAuthenticationRequest event.

9.10.2.10.1 AccessType

Applies to

acAutoUpgrader, acHTTP and acWebImage components as subproperty of Proxy structure.

Declaration

```
type
  TacHTTPAccessType = (atPreconfig, atDirect, atProxy);

property AccessType: TacHTTPAccessType;
```

Description

The AccessType property controls how the acAutoUpgrader components should access the remote server to request data. The AccessType can be *direct* or via the *proxy* server. If you'd like to use the access type previously configured in the Control Panel — leave the AccessType = atPreconfig.

Values	Meaning
atPreconfig	retrieves the proxy or direct configuration from the registry (user can configure the access type in the Control Panel);
atDirect	uses the direct connection and resolves all host names locally;
atProxy	access the remote data via the proxy server. Passes all requests to the proxy, unless a proxy bypass list is not empty and the name to be resolved bypasses the proxy. To specify the proxy server and port — use ProxyServer and ProxyPort properties. To configure bypass list — use ProxyBypass property.

See also

ProxyServer, ProxyPort and ProxyBypass properties.

9.10.2.10.2 ProxyBypass

Applies to

acAutoUpgrader, acHTTP and acWebImage components as subproperty of Proxy structure.

Declaration

```
property ProxyBypass: String;
```

Description

The ProxyBypass contains the list of host names or IP addresses, or both, that should not be routed through the proxy. The list can contain wildcards. If the ProxyBypass is empty, the component reads the bypass list from the registry.

You can use wild cards to match domain and host names or addresses — for example, www.*.com;128.*;240.*;*.mygroup.*;*x* and so on. Use semicolon (;) to separate entries.

© 1998-2006, UtilMind Solutions®

See also
ProxyServer and ProxyPort properties.

9.10.2.10.3 ProxyPassword

Applies to
acAutoUpgrader, acHTTP and acWebImage components as subproperty of Proxy structure.

Declaration
```
property ProxyPassword: String;
```

Description
The ProxyPassword property specifies the password required for connection via proxy server, if it's require secure authentication.

Alternativaly you can write OnProxyAuthenticationRequest event handler and specify proxy username and password dynamically.

Note
If your proxy server uses some "challenge-response" authentication scheme — please make sure that `ioKeepConnection` option of the InternetOptions is set to True. Otherwise, connection with proxy could be dropped.

See also
ProxyUsername and InternetOptions properties;
OnProxyAuthenticationRequest event.

9.10.2.10.4 ProxyPort

Applies to
acAutoUpgrader, acHTTP and acWebImage components as subproperty of Proxy structure.

Declaration
```
property ProxyPort: Integer;
```

Description
The ProxyPort property specifies the port number for the proxy server. The default value for proxy ports is **8080**.

See also
ProxyServer and ProxyBypass properties.

9.10.2.10.5 ProxyServer

Applies to
acAutoUpgrader, acHTTP and acWebImage components as subproperty of Proxy structure.

Declaration
```
property ProxyServer: String;
```

Description
The ProxyServer property specifies the hostname or IP address of the proxy server to use if the proxy access was specified in the AccessType property.

See also
ProxyPort and ProxyBypass properties.

© 1998-2006, UtilMind Solutions®

9.10.2.10.6 ProxyUsername

Applies to
acAutoUpgrader, acHTTP and acWebImage components as subproperty of Proxy structure.

Declaration
```
property ProxyUsername: String;
```

Description
The ProxyUsername property specifies the username required for connection via proxy server, if it's require secure authentication.

Alternativaly you can write OnProxyAuthenticationRequest event handler and specify proxy username and password dynamically.

Note
If your proxy server uses some "challenge-response" authentication scheme — please make sure that `ioKeepConnection` option of the InternetOptions is set to True. Otherwise, connection with proxy could be dropped.

See also
ProxyPassword and InternetOptions properties;
OnProxyAuthenticationRequest event.

9.10.2.11 RestartParams

Applies to
acAutoUpgrader component.

Declaration
```
property RestartParams: String;
```

Description
The RestartParams property specifies the command-line arguments which should be used on restarting the updated application. When the application receives these command-line parameters (on restarting after update) — it triggers OnAfterRestart event.

The RestartParams property can be also used to specify command-line options for another programs which will secretly extract your files locally on user's PC. This can be the setup program or self-extracting archive, which may require some command line options to extract files in "silent mode".

For example, you think that local update is better done by a setup program. So you just need to download the setup file and execute it with "silent installation" command line option. The setup will locally extract all required files then run the main program upon completion.

Let suppose you are using Inno Setup (http://www.jrsoftware.org/isinfo.htm). Then just set the RestartParams property to "/silent" or "/verysilent" and point the #url in the InfoFile to your setup file.

Important note!
If you wish to use AutoUpgrader just for downloading the external setup program which will locally extract all required files, make sure that UpgradeMethod property set to `umUseExternalSetup` (or, another words, `#method=1` in the InfoFile).

See also

© 1998-2006, UtilMind Solutions®

OnAfterRestart event, UpgradeMethod property;
InfoFile example.

9.10.2.12ShowMessages

Applies to
acAutoUpgrader component.

Declaration

```
type
  TacAutoUpgraderShowMessages = set of (mAskUpgrade, mConnLost,
mHostUnreachable, mLostFile, mNoInfoFile, mNoUpdateAvailable,
mPasswordRequest);

property ShowMessages: TacAutoUpgraderShowMessages;
```

Description
The ShowMessages is the set of flags which determines which standard notification messages and request dialogs should or should not be automatically displayed / handled on some AutoUpgrader's events.

These settings let's you to use AutoUpgrader without additional coding of the event handers. Text for messages translated accordingly to settings specified in Language property of Wizard structure.

There is the list of standard flags. Set some of them to True to enable automatic handling and displaying of these notifications, or set to False to disable them. Alternatively you may use numerous events which duplicates standard messages and allows to write custom processing.

Flags	Meaning
mAskUpgrade	when *mAskUpgrade* is True, AutoUpgrader will notify user that newer version is available for download and whether user would like to upgrade it. When *mAskUpgrade* is False it will try to upgrade the application without confirmation. To hook this event — write the OnBeginUpgrade event handler;
mConnLost	notifies user about disconnection, when it True. If you'd like to use custom message box — set *mConnLost* flag to False and write your own OnConnLost event handler;
mHostUnreachable	notifies user that requested remote host is unreachable. Most common reasons is: (a) user currently disconnected from Internet; (b) remote host is down; (c) wrong hostname in the InfoFileURL property or in the list of Files which should be downloaded and updated. By default this flag is disabled (False). You can catch this event using OnHostUnreachable handler.
mLostFile	the requested file was not found in the Internet addres specified in remote Info-file. This dialog asks user whether he(she) like to skip lost file and try to continue upgrade without it. Set this flag to False and write OnLostFile event handler to impelement custom dialog or to Abort the upgrading when this happends;
mNoInfoFile	the file which contains the upgrade information was is inaccessible or was not found by address specified in the InfoFileURL. This flag disabled by default. Use OnNoInfoFile event to make some actions if the Info-file was not found;
mNoUpdateAvailable	notifies user that update is not available and he(she) currently using most recent version of the application. Disabled by default. Set *mNoUpdateAvailable* it to True or write OnNoUpdateAvailable event handler;
mPasswordRequest	ensures whether AutoUpgrader should automatically asks the username and password, if some file placed in password protected directory. Standard

© 1998-2006, UtilMind Solutions®

password-request dialog looks like **this**. To disable it and/or use custom password-request dialog — set *mPasswordRequest* flag to False and write OnPasswordRequest event handler.

Recomendations

If your program looking for its updates automatically at once the AutoUpgrader detects the Internet connection (AutoCheck = True), we strongly recomend to turn off some standard error messages like `mHostUnreachable`, `mNoInfoFile`, `mNoUpdate` to not confuse user with unexpected error messages. By default these options is turned off.

See also

InfoFileURL property and InfoFile structure;
OnConnLost, OnHostUnreachable, OnLostFile, OnNoInfoFile, OnNoUpdateAvailable and OnPasswordRequest events;
Example of server-based file which contains upgrade information (Info-file).

9.10.2.13ThreadPriority

Applies to

acAutoUpgrader, acHTTP, acHostnameResolver and acSendMail components.

Declaration

```
property ThreadPriority: TThreadPriority;
```

Description

ThreadPriority indicates the priority used when scheduling the thread. Adjust the priority higher or lower as needed.

TThreadPriority type defines the possible values for the Priority property of the acThread component, as defined in the following table. The system schedules CPU cycles to each thread based on a priority scale; the Priority property adjusts a thread's priority higher or lower on the scale.

Values	Meaning
tpIdle	The thread executes only when the system is idle. The system will not interrupt other threads to execute a thread with tpIdle priority.
tpLowest	The thread's priority is two points below normal.
tpLower	The thread's priority is one point below normal.
tpNormal	The thread has normal priority.
tpHigher	The thread's priority is one point above normal.
tpHighest	The thread's priority is two points above normal.
tpTimeCritical	The thread gets highest priority.

Warning

Boosting the thread priority of a CPU intensive operation may "starve" the other threads in the application. Only apply priority boosts to threads that spend most of their time waiting for external events.

See also

acThread component.

9.10.2.14TransferBufferSize

Applies to

acAutoUpgrader and acHTTP components.

© 1998-2006, UtilMind Solutions®

Declaration

```
type
  TacBufferSize = 256..MaxInt; // 2147483647 bytes maximum

property TransferBufferSize: TacBufferSize; // 4096 bytes by default
```

Description

The TransferBufferSize property specifies the size of buffer (in bytes) for writing or reading data from the Web.

The OnProgress event will be triggered after downloading each block with size specified in this property.

See also

OnProgress event.

9.10.2.15VersionControl

Applies to

acAutoUpgrader component.

Declaration

```
type
  TacAutoUpgraderVersionControl = (byDate, byNumber);

property VersionControl: TacAutoUpgraderVersionControl;
```

Description

The VersionControl property determines which property will used for version control — VersionDate (`#date` variable in the Info-file) or VersionNumber (`#version` variable in the Info-file).

If VersionControl is *byDate*, then after reading the Info-file (from location specified in InfoFileURL), AutoUpgrader will compare `#date` variable of Info-file with value specified in VersionDate property. If VersionDate lower than date stated in `#date` variable, AutoUpgrader will think that newest version of your application already available on your Web site and will try to upgrade the software on user's PC. Value of the VersionNumber property will be ignored.

If VersionControl is *byNumber*, then after reading the Info-file, AutoUpgrader will compare `#number` variable with value of VersionNumber property. If VersionNumber is not equal to `#number` variable, AutoUpgrader will try to upgrade your application on user's machine, guessing that newest release version available for download. VersionDate will be ignored.

See also

VersionDate, VersionNumber and InfoFileURL properties.

9.10.2.16VersionDate

Applies to

acAutoUpgrader component.

Declaration

```
property VersionDate: String;
```

Description

The VersionDate property specifies the release date of current version of your application. This value can be always set automatically at design-time if VersionDateAutoSet property is True.

© 1998-2006, UtilMind Solutions®

The value of VersionDate property must have following format:
MM/DD/YYYY (for example, **02/24/2000**, that is 24 February 2000). (This is standard US short date style, little bit inconvenient for Europeans though<g>)

If VersionControl is *byDate*, then after reading the Info-file from the Web, AutoUpgrader will compare `#date` variable of Info-file with value in VersionDate property. If VersionDate lower than date specified in `#date` variable, AutoUpgrader will guess that newer version already available on your Web site and will try to upgrade your application on user's PC. VersionNumber property will be ignored.

See also
VersionControl, VersionDateAutoSet and VersionNumber properties.

9.10.2.17VersionDateAutoSet

Applies to
acAutoUpgrader component.

Declaration
```
property VersionDateAutoSet: Boolean;
```

Description
The VersionDateAutoSet property controls whether the value of VersionDate property should be automatically updated on every application rebuild.

When VersionDateAutoSet is True, VersionDate property will contain only current date (date of last recompiling).

See also
VersionDate property.

9.10.2.18VersionNumber

Applies to
acAutoUpgrader component.

Declaration
```
property VersionNumber: String;
```

Description
The VersionNumber property specifies current version number of your application. Use this property to specify the version number (or any text string which identifies current version) if you perform VersionControl byNumber.

If VersionControl is *byNumber*, then after reading the Info-file from the Web, AutoUpgrader will compare `#number` variable with value in VersionNumber property. If VersionNumber is not equal to value specified in `#number` variable, AutoUpgrader will try to upgrade your application on user's PC, guessing that newest version already available for download. VersionDate property will be ignored.

See also
VersionControl and VersionDate properties;
Info-file example.

© 1998-2006, UtilMind Solutions®

9.10.2.19Wizard

Applies to
acAutoUpgrader component.

Declaration
```
type
  TacAutoUpgraderWizard = class
  published
    property Enabled: Boolean default True;
    property HideFileLocation: Boolean;
    property Pic118x218: TBitmap;
    property Language: TacAutoUpgraderLanguage;
    property StayOnTop: Boolean;
  end;

property Wizard: TacAutoUpgraderWizard;
```

Description
The Wizard structure allows to specify some preferences for "Application update wizard" dialog.

Purpose of the "Application update wizard" destined allows you to implement auto-upgrade mechanism to your software with just single or without any line of code, don't wasting time to programming the event handers and progress dialogs.

The AutoUpgrader component automatically supports number of various languages

See also
ShowMessages property.

9.10.2.19.1 Enabled

Applies to
acAutoUpgrader component as subproperty of Wizard structure.

Declaration
```
property Enabled: Boolean; // True by default
```

Description
The Enabled property of Wizard structure controls whether you would like to use built-in "Application update wizard" dialog in your auto-upgradable application.

By default, wizard is enabled. However, if you wish to implement your own, custom progress dialog using numerous events, just set Enabled to False and wizard will not be used.

See also
HideFileLocation, Language, Pic118x218 and StayOnTop properties.

9.10.2.19.2 HideFileLocation

Applies to
acAutoUpgrader component as subproperty of Wizard structure.

Declaration
```
property HideFileLocation: Boolean; // False by default
```

Description

© 1998-2006, UtilMind Solutions®

The HideFileLocation property determines whether the location of downloading files should be hidden in the Application Update Wizard window.

By default users can see from where the file are downloading, but if you wish to hide the file name and location — just set this property to True.

See also

Enabled, Language, Pic118x218 and StayOnTop properties.

9.10.2.19.3 Language

Applies to

acAutoUpgrader component as subproperty of Wizard structure.

Declaration

```
type
{ Newer versions of AutoUpgrader component may contain much more
languages! Check out the version history for AppControls library at:
http://www.appcontrols.com/appcontrols/history.html }
  TacAutoUpgraderLanguage = (wlAuto, wlEnglish, wlRussian,
                             wlGerman, wlFrench, wlSpanish,
                             wlPortuguese, wlPortugueseBrazilian,
                             wlItalian, wlDutch, wlDanish,
                             wlSwedish, wlFinnish, wlEstonian,
                             wlUkrainian, wlPolish, wlCzech,
                             wlHungarian, wlRomanian, wlCatalan,
                             wlBasque, wlAfrikaans,
wlChineseTraditional,
                             wlChineseSimplified, wlIndonesian, wlMalay,
                            wlIcelandic);

property Language: TacAutoUpgraderLanguage; // wlAuto by default
```

Description

The Language property specifies the language which will be used to display all standard text messages within the AutoUpgrader component.

When you specifying the Language, all text in notification messages and the "Appliction update wizard" will be translated accordingly (see screenshot #2).

By default Language = `wlAuto`, that means that AutoUpgrader should check out current regional settings on user PC and use his native language, it's possible. If you wish to test this feature and temporary change your regional settings — go to the "Control Panel" and doubleclick "Regional Settings" icon.

See also

ShowMessages property;
Enabled, HideFileLocation, Pic118x218 and StayOnTop properties of Wizard structure.

9.10.2.19.4 Pic118x218

Applies to

acAutoUpgrader component as subproperty of Wizard structure.

Declaration

```
property Pic118x218: TPicture;
```

© 1998-2006, UtilMind Solutions®

Description
The Pic118x218 property specifies a Bitmap or JPEG image which will be displayed at the left side of Application Update Wizard.

You can specify picture with ANY size, however it will *stretched* to fit bounds of the panel where Width = 118 pixels and Height = 218 pixels. If picture is not specified then AutoUpgrader will display default bitmap (see screenshot).

See also
Enabled, Language and StayOnTop properties.

9.10.2.19.5 StayOnTop

Applies to
acAutoUpgrader component as subproperty of Wizard structure.

Declaration
```
property StayOnTop: Boolean; // True by default
```

Description
The StayOnTop property specifies whether the Application Update Wizard form should stay over other windows of your application, so user will always see the update progress.

Note
The StayOnTop property controls the topmost position only within single application. Setting the StayOnTop to True will NOT make the Wizard stay over other application's windows.

See also
Enabled, HideFileLocation, Language and Pic118x218 properties.

9.10.3 Methods

9.10.3.1 Abort

Applies to
acAutoUpgrader, acHTTP, acWebImage, acSendMail and acSQLQueryThread components.

Declaration
```
procedure Abort;
```

Description
The Abort method terminates the upgrade process on any stage. Call it to imediately stop the upgrading.

After successful termination the OnAborted event occurs.

AutoUpgrader can automatically resume incomplete download, because it download all files to cache, in case if you have specified to use it in CacheOptions. To resume the upgrade — just call CheckUpdate method again.

See also
CheckUpdate method;
OnAborted event;
CacheOptions property.

© 1998-2006, UtilMind Solutions®

9.10.3.2 CheckUpdate

Applies to
acAutoUpgrader component.

Declaration
```
function CheckUpdate(JustCheckIsUpdateAvailable: Boolean = False):
Boolean;
```

Description
Call CheckUpdate method to check for the application updates and initiate the upgrade process (if newer version are really exists, and JustCheckIsUpdateAvailable parameter is False on function call).

In case if you just want to check whether update available, without any message boxes — call this method as `CheckUpdate(True)`, and check whether update are available in function result.

How does it works?
When you call CheckUpdate method, the AutoUpgrader will download the file which contains the upgrade information (Info-file) from the URL specified in InfoFileURL property. If AutoUpgrader can not read remote information file, the OnNoInfoFile event occurs (or OnHostUnreachable, in case if user currently disconnected from Internet).

If file which contains the upgrade information has successfully downloaded, AutoUpgrader parses it to check whether the newer version of application already available for download. If compares the **#date** variable in Info-file with VersionDate property, or **#number** variable with VersionNumber property (this depends on method which you using for VersionContorl).

If new version are really exists, it fires OnBeginUpgrade event, then initiate downloading of all updated files to replace them locally.

Example
Delphi:
```
procedure TForm1.CheckUpdateBtnClick(Sender: TObject);
begin
  acAutoUpgrader1.CheckUpdate;
end;
```

C++ Builder:
```
void __fastcall TForm1::CheckUpdateBtn1Click(TObject *Sender)
{
  acAutoUpgrader1->CheckUpdate();
}
```

See also
Abort methods; Info-file example;
AutoCheck, InfoFileURL and VersionControl properties;
OnBeginUpgrade, OnNoUpdateAvailable and OnNoInfoFile events.

9.10.3.3 RestartApplication

Applies to
acAutoUpgrader component.

Declaration

© 1998-2006, UtilMind Solutions®

```
procedure RestartApplication(const Params: String {$IFDEF D4} = ''
{$ENDIF});
```

Description

The RestartApplication used to restart the application which calls this method, with command-line parameters specified in *Params* parameter.

If *Params* is not specified (`''`, empty text), the application will be restarted with parameters specified in RestartParams property.

If the RestartApplication method is called after checking the the application for updates (CheckUpdate method), it will try to upgrade your application (replace old files with new, which was downloaded with *.uTMP extension), using the upgrade method specified in the Info-file.

You can also use this method to restart and upgrade your application "manually", instead of automatic restart which can be hooked and disabled in OnEndUpgrade event handler, but anyway, do not worry to call this method if you want to restart your application for ANY reason.

See also

RestartParams property;
CheckUpdate and Abort methods;
OnEndUpgrade event;
Info-file example.

9.10.4 Events

9.10.4.1 OnAborted

Applies to

acAutoUpgrader, acHTTP, acWebImage and acSendMail components.

Declaration

```
property OnAborted: TNotifyEvent;
```

Description

The OnAborted event occurs when user canceled the upgrade process in the "Application update wizard" (see screenshot), or after calling the Abort method.

See also

Abort method.

9.10.4.2 OnAfterRestart

Applies to

acAutoUpgrader component.

Declaration

```
property OnAfterRestart: TNotifyEvent;
```

Description

The OnAfterRestart event occurs on first start of upgraded application. Write this event handler to do some specific actions after successful program update and after restart (i.e: show some "history" notes, etc).

When application restarts after its upgrade, it execute itself with command-line parameter specified in RestartParams property. The AutoUpgrader simply checks whether this argument exists

© 1998-2006, UtilMind Solutions®

```
if ParamStr(1) = RestartParams then
```
then, in case of need, OnAfterRestart event will triggered.

Remark

The AutoUpgrader could not trigger this event if you use some setup program or self-extracting archive to replace files from archive after downloading, instead of updating them with built-in updating mechanism. See description of RestartParams property for more details.

See also

RestartParams property; OnEndUpgrade event.

9.10.4.3 OnBeginUpgrade

Applies to

acAutoUpgrader component.

Declaration

```
type
  TacAUBeginUpgradeEvent = procedure(Sender: TObject;
     const UpgradeMsg: String; UpgradeMethod: TacUpgradeMethod;
     Files: TStringList; var CanUpgrade: Boolean) of object;

property OnBeginUpgrade: TacAUBeginUpgradeEvent;
```

Description

The OnBeginUpgrade event occurs when newer version of your application is available for download and AutoUpgrader are ready to upgrade it.

The OnBeginUpdate event occurs after successful downloading of the Info-file (which contains upgrade information), from location specified in InfoFileURL property. AutoUpgrader parses the info-file and retrieve the information required for upgrade. Some of this information passes to the OnBeginUpgrade event handler as parameters:

Parameter	Meaning
UpgradeMsg	this is the text string, specified in the downloaded Info-file, which describes release notes or new features in recent version. See UpgradeMsg property of the InfoFile structure for more information;
UpgradeMethod	determines how the application should be updated. Value can be either `umAutoUpgrade` (updated files will be automatically downloaded and replaced) or `umRedirectToURL` (user will redirected to first URL listed in *Files* parameter). See UpgradeMethod property for more information;
Files	the list of URLs to files which should be downloaded and updated locally;
CanUpgrade	set this variable to False if you do NOT want to continue the upgrade. To decide whether you want to download each file separately — use OnFileStart event.

At once after this event the AutoUpgrader will show the "Application update wizard" (if it is Enabled).

Example 1 (can be used if you want to implement custom dialog box which informs user about new version and prompts upgrade)

```
procedure TForm1.acAutoUpgrader1BeginUpgrade(Sender: TObject;
  const UpgradeMsg: string; UpgradeMethod: TacUpgradeMethod;
  Files: TStringList; var CanUpgrade: Boolean);
```

© 1998-2006, UtilMind Solutions®

```
begin
  CanUpgrade := Application.MessageBox('New version available. Would you
like to upgrade?',
      PChar(Application.Title), MB_YESNO or MB_ICONQUESTION) = ID_YES;
end;
```

Example 2 (awaiting while user click Next button, then do upgrade)

```
// This example demonstrates how to:
// 1. Check application for its updates;
// 2. Display the update information in some custom wizard (in
OnBeginUpgrade event handler);
// 3. Do not upgrade imediately, wait until user press "Next" button
(but don't smoke
//    the CPU with constant Application.ProcessMessages loop)
type
  TCustomWizardForm = class(TForm)
  public
    WaitEvent: THandle; // handle for Wait and StopWait routines
    ContinueUpgrade: Boolean;
  end;

implementation

uses acUtils;

procedure TCustomWizardForm.FormShow(Sender: TObject);
begin
  acAutoUpgrader1.CheckUpdate;
end;

procedure TCustomWizardForm.acAutoUpgrader1BeginUpgrade(Sender: TObject;
const UpgradeMsg: String;
  UpgradeMethod: TacUpgradeMethod; Files: TStringList; var CanUpgrade:
Boolean);
begin
  ContinueUpgrade := False;  // by default
  Wait(WaitEvent, INFINITE); // wait for some action (until StopWait is
called or application terminated)
  CanUpgrade := ContinueUpgrade;
end;

// this will stop waiting and switch to next page of wizard
procedure TCustomWizardForm.WizardNextButtonClick(Sender: TObject);
begin
  ContinueUpgrade := True;
  StopWait(WaitEvent);
end;
```

See also

OnEndUpgrade, OnProgress, OnFileStart and OnFileDone events;
Wizard structure.

9.10.4.4 OnConnLost

Applies to

acAutoUpgrader component.

© 1998-2006, UtilMind Solutions®

Declaration

```
property OnConnLost: TNotifyEvent;
```

Description

The OnConnLost event occurs when the connection with remote server lost for some reason, at the moment of downloading the data.

Write the OnConnLost event hander to implement custom message box which notify user about disconnection from Internet, or to perform some specific actions when the connection lost on downloading the data.

The AutoUpgrader have the standard dialog box, destineddestined to notify user about this event. If you wish to show this message automatically (in native user's language, specified in Language property), set `mConnLost` flag of ShowMessages property to True, or otherwise, set it to False if don't want to display this message automatically.

See also

ShowMessages and Language properties;
OnHostUnreachable and OnLostFile events.

9.10.4.5 OnDoOwnCloseAppMethod

Applies to

acAutoUpgrader component.

Declaration

```
property OnDoOwnCloseAppMethod: TNotifyEvent;
```

Description

The DoOwnCloseAppMethod can be used to write your own method which will close the application before upgrade and restart of the updated application.

Remarks

If you write this method, the AutoUpgrader will NOT execute its own methods which automatically close the application. When you override AutoUpgrader's default close method, you should handle it programmatically in this event handler.

By default, if this event is not assigned, the AutoUpgrader will execute following code to shutdown the application:

```
procedure ShutdownApplication;
begin
  if Assigned(Application.MainForm) then
    Application.MainForm.Close;
  Application.Terminate;
end;
```

See also

OnEndUpgrade and OnAfterRestart events.

9.10.4.6 OnEndUpgrade

Applies to

acAutoUpgrader component.

Declaration

```
type
```

© 1998-2006, UtilMind Solutions®

```
  TacAUEndUpgradeEvent = procedure(Sender: TObject; var
RestartImediately: Boolean) of object;

property OnEndUpgrade: TacAUEndUpgradeEvent;
```

Description

The OnEndUpgrade event occurs when application has successfully updated and about to be restarted.

If you are using the built-in Application Update Wizard, this event will occur when user press OK button on Wizard form.

Set *RestartImediately* parameter to False, if you don't want to restart your application imediately after processing this event handler, and prefer to call RestartApplication method later, to restart and upgrade the application "manually".

Example *// overrides restarting the application, awaiting until user click some "Finish" button to restart the app*

```
procedure TForm2.acAutoUpgrader1EndUpgrade(Sender: TObject);
begin
  RestartImediately := False; // restart it later when user click
"Finish" button
end;

procedure TForm2.FinishButtonClick(Sender: TObject);
begin
  RestartApplication(''); // restart with default parameters
                          // (RestartParams + use upgrade method
specified in the Info-file)
end;
```

See also

RestartParams property and RestartApplication method;
OnDoOwnCloseAppMethod, OnAfterRestart and OnBeginUpgrade events.

9.10.4.7 OnFileDone

Applies to

acAutoUpgrader component.

Declaration

```
type
  TacAUFileDoneEvent = procedure(Sender: TObject;
    const FileName: String) of object;

property OnFileDone: TacAUFileDoneEvent;
```

Description

The OnFileDone event occurs when AutoUpgrader successfully downloaded one file from the queue. The *FileName* parameter is the path and the file name where the file has been stored after downloading.

To get all files which should to be updated — use *Files* parameter in the OnBeginUpdate event handler. If you wish to hook the event when the file is about to be downloaded and decide whether you really want to download it — use OnFileStart event.

© 1998-2006, UtilMind Solutions®

See also
OnFileStart, OnBeginUpgrade, OnEndUpgrade and OnProgress events.

9.10.4.8 OnFileStart

Applies to
acAutoUpgrader component.

Declaration
```
type
  TacAUFileStartEvent = procedure(Sender: TObject;
     const FileURL: String; FileSize: Integer; FileTime: TDateTime;
     var CanUpgrade: Boolean) of object;

property OnFileStart: TacAUFileStartEvent;
```

Description
The OnFileStart event occurs after receiving the headers of the file which is about to be downloaded.

Write the OnFileStart event handler to get the name and size of file which about to be downloaded, plus the date/time of its last modification (time when it has been uploaded to your website). Also, you can use this event to decide, whether you really want to download this file. If your program think that this file should already updated — set *CanUpgrade* parameter to False and AutoUpgrader will not download it from your website, and will pass to next file in the download queue.

Alternatively, if you don't want to download files only in case if their sizes are different, you can set MiscFilesUpgrade property to `mfuOnlyIfSizeDifferent`, and the AutoUpgrader will automatically compare the size of remote and local files to decide whether they should be upgraded or not.

See also
MiscFilesUpgrade property;
OnFileDone, OnBeginUpgrade, OnEndUpgrade and OnProgress events;
OnHeaderInfo event of acHTTP component.

9.10.4.9 OnHostUnreachable

Applies to
acAutoUpgrader, acHTTP and acSendMail components.

Declaration
```
type
  TacAUHostUnreachableEvent = procedure(Sender: TObject;
     const URL, Hostname: String) of object;

property OnHostUnreachable: TacAUHostUnreachableEvent;
```

Description
The OnHostUnreachable event occurs if the AutoUpgrader can not connect to some remote host. The unreachable host passes in *Hostname* parameter and full path to inaccessible file in the *URL* parameter.

Possible reasons of this problem is:
1. User currently not connected to the Internet;
2. Requested host name is unknown. Please check spelling of domain name in the InfoFileURL property and in URLs listed in the file which contains upgrade information, or in the Files list of

© 1998-2006, UtilMind Solutions®

InfoFile structure;
3. Remote server is down (disconnected from Internet).

The AutoUpgrader have the standard dialog box, destined to notify user about this event. If you wish to show this message automatically (in native user's language, specified in Language property), set `mHostUnreachable` flag of ShowMessages property to True, or otherwise, set it to False if don't want to display this message.

Example

Delphi:

```
procedure TForm1.acAutoUpgrader1HostUnreachable(Sender: TObject;
  URL, Hostname: string);
begin
   Application.MessageBox(PChar('Host http://' + Hostname + ' is
unreachable.'#13#13'Please check your Internet connection and'#13'try to
upgrade this software again later.'),
                          PChar(Application.Title),
                          MB_OK or MB_ICONSTOP);
end;
```

C++ Builder:

```
void __fastcall TForm1::acAutoUpgrader1HostUnreachable(TObject *Sender,
AnsiString URL, AnsiString Hostname)
{
  AnsiString Msg =
      "Host http://" + Hostname + " is unreachable.\n\n"
      "Please check your Internet connection and\n"
      "try to upgrade this software again later.";
   Application->MessageBox(Msg.c_str(),
                           Application->Title.c_str(),
                           MB_OK | MB_ICONSTOP);
}
```

See also

InfoFIleURL and ShowMessages properties;
Files property in the InfoFile structure; Info-file example;
Hostname property of acHTTP component.

9.10.4.10OnLaterUpgrade

Applies to

acAutoUpgrader component.

Declaration

```
property OnLaterUpgrade: TNotifyEvent;
```

Description

The OnLaterUpgrade event occurs if user press "Later" button in standard dialog box which asks about upgrade.

The standard dialog box which asks user about upgrade will be displayed only if `mAskUpgrade` flag is enabled in ShowMessages property. (If you don't want to ask about upgrade at all, for example to start upgrade imediately, just set `mAskUpgrade` flag to False.)

Screenshot

© 1998-2006, UtilMind Solutions®

See also
ShowMessages property.

9.10.4.11 OnLostFile

Applies to
acAutoUpgrader component.

Declaration

```
var
  TacAULostFileEvent = procedure(Sender: TObject; const FileURL: String;
ErrorCode: Integer; var ContinueUpgrade: Boolean) of object;

property OnLostFile: TacAULostFileEvent;
```

Description
The OnLostFile event occurs if some file, of that which listed in the downloaded Info-file, can not be downloaded by specified URL (error code has received in the response header from HTTP server).

Most common error reason is HTTP error #404 (document not found), but please check out all HTTP status codes for more information about HTTP errors.

The AutoUpgrader passes following parameters to the OnLostFile event handler:

Parameter	Meaning
FileURL	the Internet address of the file which we just tryed to download;
ErrorCode	integer number which identifies the HTTP error (see the list of HTTP Status Codes to recognize an error).;
ContinueUpgrade	set this flag variable to True if you wish to skipinaccessible file and still continue the upgrade.

The AutoUpgrader have the standard dialog box, destined to notify user about this event. If you wish to show this message automatically (in native user's language, specified in Language property), set `mLostFile` flag of ShowMessages property to True, or otherwise, set it to False if don't want to display this message automatically.

Example
Delphi:

```
procedure TForm1.acAutoUpgrader1LostFile(Sender: TObject;
  FileURL: string; ErrorCode: Integer;
  var ContinueUpgrade: Boolean);
begin
  ContinueUpgrade := Application.MessageBox(PChar('File "' + FileURL +
'" not found or inaccessible.'#13#13'Try to skip this file and continue
upgrade?'),
```

© 1998-2006, UtilMind Solutions®

```
    PChar('Error #' + IntToStr(ErrorCode)),
    MB_YESNO or MB_ICONWARNING) = ID_YES;
end;
```

C++ Builder:

```
void __fastcall TForm1::acAutoUpgrader1LostFile(TObject *Sender,
      AnsiString FileURL, int ErrorCode, bool &ContinueUpgrade)
{
  AnsiString Msg  =
      "File \'" + FileURL + "\' not found or inaccessible.\n\n"
      "Try to skip this file and continue upgrade?";
  AnsiString Capt =
      "Error #" + IntToStr(ErrorCode);

  ContinueUpgrade = Application->MessageBox(Msg.c_str(),
        Capt.c_str(), MB_YESNO | MB_ICONWARNING) == ID_YES;
}
```

See also

ShowMessages property;
Info-file example; HTTP Status Codes.

9.10.4.12OnNoInfoFile

Applies to

acAutoUpgrader component.

Declaration

```
property OnNoInfoFile: TNotifyEvent;
```

Description

The OnNoInfoFile event occurs if the file which contain the upgrade information was not found in location specified in the InfoFileURL property.

The AutoUpgrader have the standard dialog box, destined to notify user about this event. If you wish to show this message automatically (in native user's language, specified in Language property), set `mNoInfoFile` flag of ShowMessages property to True, or otherwise, set it to False if don't want to display this message.

Remarks

This event may also occur if you have pointed the InfoFileURL to buggy CGI program, which execution failed (AutoUpgrader received HTTP error #500 in the header of response from server).

See also

InfoFileURL, ShowMessages and Language properties;
OnNoUpdateAvailable event.

9.10.4.13OnNoUpdateAvailable

Applies to

acAutoUpgrader component.

Declaration

```
property OnNoUpdateAvailable: TNotifyEvent;
```

© 1998-2006, UtilMind Solutions®

Description

The OnNoUpdateAvailable event occurs if no update avaliable for download (newer version not yet released).

The OnNoUpdateAvailable occurs if AutoUpgrader successfully downloaded some content from specified InfoFileURL (remote server returned status code 200 OK) but it appears that no update available (after comparision of the release date or version number (depending on value of VersionControl property) it ensures that current version is latest) or upgrade information was not found among downloaded data.

If you wish to automatically display standard message which notify user that "there is no update available and he actually using most recent version", set `mNoUpdateAvailable` flag in ShowMessages property to True or, otherwise, set it to False to disable this message box.

See also

InfoFileURL, VersionControl and ShowMessages properties;
Info-file example; OnNoInfoFile event.

9.10.4.14OnPasswordRequest

Applies to

acAutoUpgrader, acHTTP and acWebImage components.

Declaration

```
type
  TacHTTPPasswordRequestEvent = procedure(Sender: TObject;
    var TryAgain: Boolean) of object;

property OnPasswordRequest: TacHTTPPasswordRequestEvent;
```

Description

The OnPasswordRequest event can be used to implement custom dialog box which asks user for his username and password to get access to the file which stored in the password protected Web area.

Usually you don't need to use this event. For example, your files can be open for everyone, or you wish to use standard, built-in password-request dialog (see screenshot) which will displayed if `mPasswordRequest` flag of ShowMessages property set to True.

However, if you still want to use your own password-request dialog, then set *TryAgain* parameter to True to retry the HTTP query (attempt to download the file), and specify correct login information to the HTTPUsername and HTTPassword properties.

Example

Delphi:

```
procedure TacAutoUpgrader.AutoUpgrader1PasswordRequest(Sender: TObject;
   var TryAgain: Boolean);
begin
  { PassForm is any form with two edit boxes used for entering the login
information (username and password) }
  if PassForm.ShowModal = ID_OK then
  begin
    acAutoUpgrader1.HTTPUsername := PassForm.UsernameEdit.Text;
    acAutoUpgrader1.HTTPPassword := PassForm.PasswordEdit.Text;
    TryAgain := True; // Retry download attempt
```

© 1998-2006, UtilMind Solutions®

```
  end;
end;
```

C++ Builder:

```
void __fastcall TForm1::acAutoUpgrader1PasswordRequest(TObject *Sender,
   bool &TryAgain)
{
  if (PassForm->ShowModal() == ID_OK)
  {
    acAutoUpgrader1->HTTPUsername = PassForm->UsernameEdit->Text;
    acAutoUpgrader1->HTTPPassword = PassForm->PasswordEdit->Text;
    TryAgain = True; // Retry HTTP query (download attempt)
  };
}
```

See also

HTTPUsername, HTTPPassword and ShowMessages properties;
Password request (screenshot).

9.10.4.15 OnProgress

Applies to

acAutoUpgrader, acHTTP and acSendMail components.

Declaration

```
type
  TacAUProgressEvent = procedure(Sender: TObject;
    const FileURL: String; FileSize, BytesRead, ElapsedTime,
    EstimatedTimeLeft: Integer;
    PercentsDone, TotalPercentsDone: Byte;
    TransferRate: Single) of object;

property OnProgress: TacAUProgressEvent;
```

Description

The OnProgress event occurs every time when component has successfully downloaded *part* of some file which should be updated.

Write the OnProgress event handler to show the download progress and implement your own, custom progess box instead of standard Application update wizard (then set Enabled property of Wizard structure to False), or use it as an addition to standard wizard.

There are following parameters which passes to the event handler:

Parameter	Meaning
FileURL	location of the file which currently downloading. (This file is one of that that listed in remote Info-file, from which we have gotten an upgrade information.);
FileSize	total size of file which we currently downloading, in bytes (if possible to determinate). **Note:** usually server does not provide information about content-length for non-binary data (i.e: for "text/html" or "text/plain" MIME-types);
BytesRead	size of already received data, in bytes;
ElapsedTime	time elapsed from beginning of download (in seconds);
EstimatedTimeLeft	estimated time left before finishing of download. (Formula: `X := FileSize / BytesRead * ElapsedTime - ElapsedTime`)

© 1998-2006, UtilMind Solutions®

PercentsDone progress for current file in percents (0%..100%);
TotalPercentsDone progress for whole download queue in percents (0%..100%);
TransferRate speed of data transfer (in Kb/s).

The OnProgress event triggered after downloading each block of data with size specified in TransferBufferSize property.

Example

Delphi:

```
procedure TForm1.acAutoUpgrader1Progress(Sender: TObject;
  FileURL: string; FileSize, BytesRead, ElapsedTime,
  EstimatedTimeLeft: Integer; PercentsDone,
  TotalPercentsDone: Byte; TransferRate: Single);
begin
  // two progress bars
  ProgressCurrentFile.Position := PercentsDone;
  ProgressAllFiles.Position := TotalPercentsDone;

  // file size label (in Kb)
  FileSizeLabel.Caption :=
       Format('File size: %.1f Kb', [FileSize / 1024]);

  // downloaded size label (in Kb)
  DownloadedLabel.Caption :=
       Format('Downloaded: %.1f Kb', [BytesRead / 1024]);

  // transfer rate label (in Kb per second)
  TransRateLabel.Caption :=
       Format('Transfer rate: %.1f Kb/s', [TransferRate]);

  // estimated time left label
  EstTimeLabel.Caption :=
       Format('Estimated time left: %d:%.2d:%.2d',
        [EstimatedTimeLeft div 60 div 60,   // hours
         EstimatedTimeLeft div 60 mod 60,   // minutes
         EstimatedTimeLeft mod 60 mod 60]); // seconds
end;
```

C++ Builder:

```
void __fastcall TForm1::acAutoUpgrader1Progress(TObject *Sender,
      AnsiString FileURL, int FileSize, int BytesRead,
      int ElapsedTime, int EstimatedTimeLeft,
      BYTE PercentsDone, BYTE TotalPercentsDone,
      float TransferRate)
{
  // two progress bars
  ProgressCurrentFile->Position = PercentsDone;
  ProgressAllFiles->Position = TotalPercentsDone;

  // file size label (in Kb)
  FileSizeLabel->Caption = Format("File size: %.1f Kb",
ARRAYOFCONST(((float)FileSize / 1024)));

  // downloaded size label (in Kb)
```

© 1998-2006, UtilMind Solutions®

```
  DownloadedLabel->Caption = Format("Downloaded: %.1f Kb",
ARRAYOFCONST(((float)BytesRead / 1024)));

  // transfer rate label (in Kb per second)
  TransRateLabel->Caption = Format("Transfer rate: %.1f Kb/s",
ARRAYOFCONST(((float)TransferRate)));

  // estimated time left label
  EstTimeLabel->Caption =
       Format("Estimated time left: %d:%.2d:%.2d",
       ARRAYOFCONST((EstimatedTimeLeft / 60 / 60,    // hours
                     EstimatedTimeLeft / 60 % 60,    // minutes
                     EstimatedTimeLeft % 60 % 60)));// seconds
}
```

See also

Wizard structure; Info-file example;
OnBeginUpgrade, OnEndUpgrade and OnFileDone events.

9.10.4.16 OnProxyAuthenticationRequest

Applies to

acAutoUpgrader, acHTTP and acWebImage components.

Declaration

```
type
  TacHTTPProxyAuthenticationRequestEvent = procedure(Sender: TObject;
    var ProxyUsername, ProxyPassword: String;
    var TryAgain: Boolean) of object;

property OnProxyAuthenticationRequest:
TacHTTPProxyAuthenticationRequestEvent;
```

Description

The OnProxyAuthenticationRequest event should be used to prompt users for their username/password to access the Web via secure proxy server which requires authentication.

Write this event to prompt and specify the *ProxyUsername* and *ProxyPassword* parameters, required for the proxy authentication, and set *TryAgain* parameter to True, to retry the HTTP query with provided login information.

Alternatively you can specify ProxyUsername and ProxyPassword properties in the Proxy structure, before the request. In case if specified username and password is okay, the OnProxyAuthenticationRequest event will not occur.

Notes

If your proxy server uses some "challenge-response" authentication scheme — please make sure that `ioKeepConnection` option of the InternetOptions is set to True. Otherwise, connection with proxy could be dropped.

If you leave this event unhandled (if you set TryAgain parameter to False), the component will generate error code #407 (Proxy Authentication Required).

See also

Proxy and InternetOptions properties;
HTTP Status Codes.

© 1998-2006, UtilMind Solutions®

9.10.5 Upgrade Information File

What is the information file (InfoFile) and what for we need it?

This file contains the information about latest version of your application. This is the server-side part of AutoUpgrader component, small text-file, stored on your website, and InfoFileURL property must point to its location.

On detecting the Internet connection (if AutoCheck = True), or after calling the CheckUpdate method, the AutoUpgrader downloads this file from your website to check for an application updates.

If VersionControl property is *byDate* — it compares the date specified in VersionDate property with the date of *latest* version, specified in `#date` variable of the .Info-file (see example of real file below). If value of `#date` variable higher than date specified in VersionDate property, then AutoUpgrader will show the dialog which notify user that newer version already available and whether he(she) would like to upgrade the software (see screenshot).

The optional text for this dialog box (release notes, new features etc) specified in `#message` variable (see example below). `#message` is not obligatory parameter, feel free to leave it empty.

If user press "Yes" button, then AutoUpgrader will attempt to download the updated files from locations specified in `#urlX` variables, for further updating of these files locally. However there is two alternative options: 1) if you wish to download just setup-file for further manual installation — set `#method` value to **1** or, 2) if you wish to just redirect user to download site — set `#method` to **2** and the AutoUpgrader will just redirect user to first listed listed `#url` (see UpgradeMethod property for more details).

In case if you are using VersionNumber for version control, then before upgrading, the AutoUpgrader will compare value in `#version` variable with value specified in VersionNumber property. If values is equal then AutoUpgrader will try to auto-upgrade software same way.

Example of information file

```
# --- Begin
#date=09/19/2001
#version=2.0.1b
#message={Version [B]2.0[] contains following features:
 * [LIME]Feature 1[DEF]
 * [RED]Feature 2[DEF]
 * [BLUE]Feature 3[DEF]
and so on...}
#url1=http://www.yourdomain.com/download/cool/CoolProgram.exe
#method=0 (self-upgrade)
# ---  End
```

Info-file Format

```
#date=MM/DD/YYYY
#version=any string (i.e: 1.1 or 2.13.04.7b)

// download queue
#url1=http://www.mydomain.com/download/MyApp.exe
#url2=http://www.mydomain.com/download/MyApp.dll
...
#urlN=http://www.mydomain.com/download/ReleaseNotes.txt

#method=0, 1 or 2 (see UpgradeMethod property)
```

© 1998-2006, UtilMind Solutions®

```
(#message begins with "{" and ends with "}" characters. Can contain
rich-style tags like in acFormHelp or acRichLabel components)
#message={long message
another line
line 3}
```

How to create Info-files?

Double-click the component icon on your form, or…
Right-click the icon and select "Edit Info-file…" menu item", or…
Click "…" button on InfoFile property in Object Inspector, or…
Open "NotePad" or any other text-editor and create this file manually<g>.

Some real Info-files on our website

http://www.appcontrols.com/download/exe/AutoUpgraderPro.inf
http://www.appcontrols.com/software/upgrades/colorpicker.inf
http://www.appcontrols.com/software/upgrades/startupman.inf
http://www.appcontrols.com/software/upgrades/virtualdrives.inf
http://www.appcontrols.com/software/upgrades/wallpaperman.inf

9.10.6 Self-upgrading mechansim

Self-upgrading mechanism, how does it works?

When the AutoUpgrader downloads required files from the Web it stores them to the working directory with **"*.*.uTMP"** extensions. It also set for them "temporary" and "hidden" attributes, so user will not see these files.

After successful downloading, the AutoUpgrader extracts so-called self-upgrading module (Upgrader3.exe, 9KB), which performs local updates to the TEMP directory, execute it then terminate the main app itself. The self-upgrading mechanism replaces all files with newer ones and restart the main application.

The self-upgrading module (Upgrader3.exe) performs the local updates with following steps:

```
1. Locate program directory by exe-name transmitted as command-line parameter
1.1. Check whether the AutoUpgrader uses the external setup program (#method=1)
1.2. Split the file name and parameters
1.3. Split the file name and path
2. Test whether an exe-file exists + get the filename with correct charcase
3. Awaiting for termination of the main program
4. Test whether newer (downloaded) exe-file exists
5. Find all downloaded (.uTMP) files
5.1. If this is executable AND newest exe still not found -- trying to
     use another suitable EXE. (This may happends if program was renamed
     by user, or local update should be done by external setup.)
5.2. Rename .uTMP file to its normal name
6. Re-execute main program (or setup file).
New steps, added in v3.1.1:
7. IF the local update should be done by external setup THEN (else terminate)
7.1. Wait until termination of setup program
7.2. Delete the setup (we don't need it anymore after installation?)
7.3. Restart the original main app
```

Note

If you wish to use AutoUpgrader for downloading just single setup-file which will perform all further updates — make sure that UpgradeMethod = `umUseExternalSetup` (`#method=1` in the Info-file). Otherwise the self-upgrading module will replace your program with downloaded setup.

9.10.7 Calculation of upgrades

Would like to calculate the number of upgrades? Then point the InfoFileURL property to the CGI script which will calculate the program upgrades, instead of real Info-file.

© 1998-2006, UtilMind Solutions®

Here is an example of such CGI script, in Perl. If you would like to use it — just cut and paste this script from here, or download it from http://www.appcontrols.com/misc/autoupgrade-cgi.zip

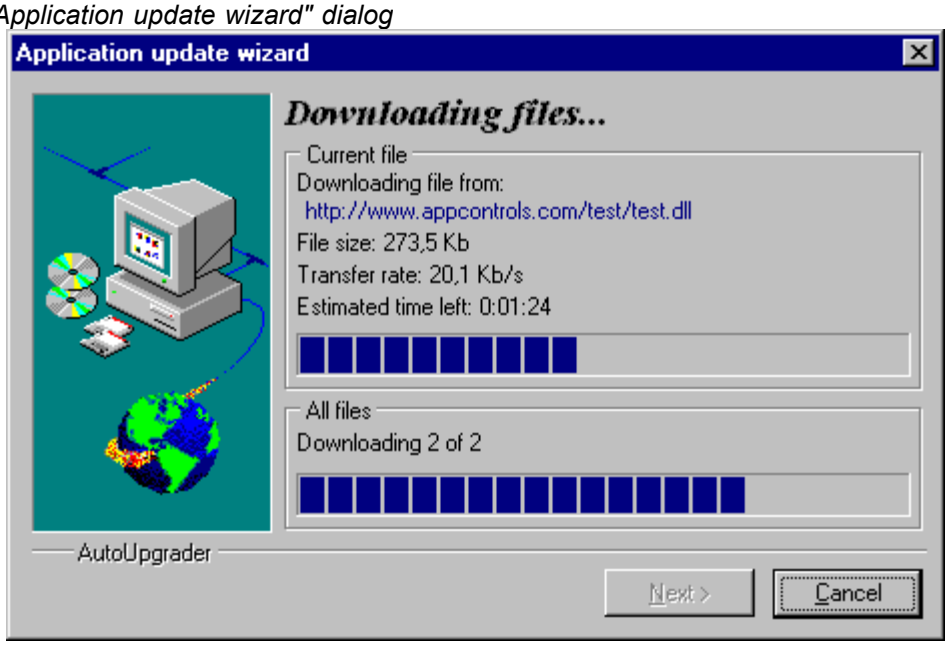
```
#!/usr/bin/perl
##############################################################################
# Auto-Upgrade counter script       Version 1.0                              #
# (c) 2000 UtilMind Solutions       info@utilmind.com                        #
#                                  http://www.utilmind.com                   #
# Created Jan 18, 1999              Last Modified Jan 18, 1999               #
##############################################################################
#  COPYRIGHT NOTICE                                                          #
# Copyright (c) 1999 by UtilMind Solutions. All Rights Reserved.             #
#                                                                            #
# This program distributed as server side part of AutoUpgrader component.    #
##############################################################################
#  INSTALLATION TIPS                                                         #
# Put this script to your CGI-BIN directory and chmod it to 755 (executable) #
# Create flat database file (users_served.txt) and chmod it to               #
# 666 (read-write permission) and install it to CGI-BIN too (see $database   #
# variable)                                                                  #
##############################################################################

####################
# Configuration

$database  = "users_served.txt"; # You may set any other name to this flat database

$info_file = "../autoupgrade.inf"; # !IMPORTANT: This is local path to REAL .Info-file

$use_flock = 1; # 1 - yes, 0 - no. Should be used but won't work Win95/98


####################
# Executable

print "Content-Type: text/html\n\n";

open(FILE, $database) || &error("Can't locate database file");
$count = <FILE>;
close(FILE);

$count++; # increasing the counter

open(FILE, ">$database") || &error("Can't update database file");
if ($use_flock eq 1) {
  flock(FILE, 2); # locking write accessing
}
print FILE $count;
if ($use_flock eq 1) {
  flock(FILE, 8); # unlocking
}
close(FILE);


open(FILE, "$info_file") || &error("Can't locate info-file");
@DATA = <FILE>;
close(FILE);

print @DATA;
print "#served=$count\n";


sub error {
  print "Error: $_[0]<br>\n";
  print "Reason: $!";
  exit;
}
```

© 1998-2006, UtilMind Solutions®

9.10.8 Screenshots

9.10.8.1 Application update wizard

Screenshot #1
"Application update wizard" dialog

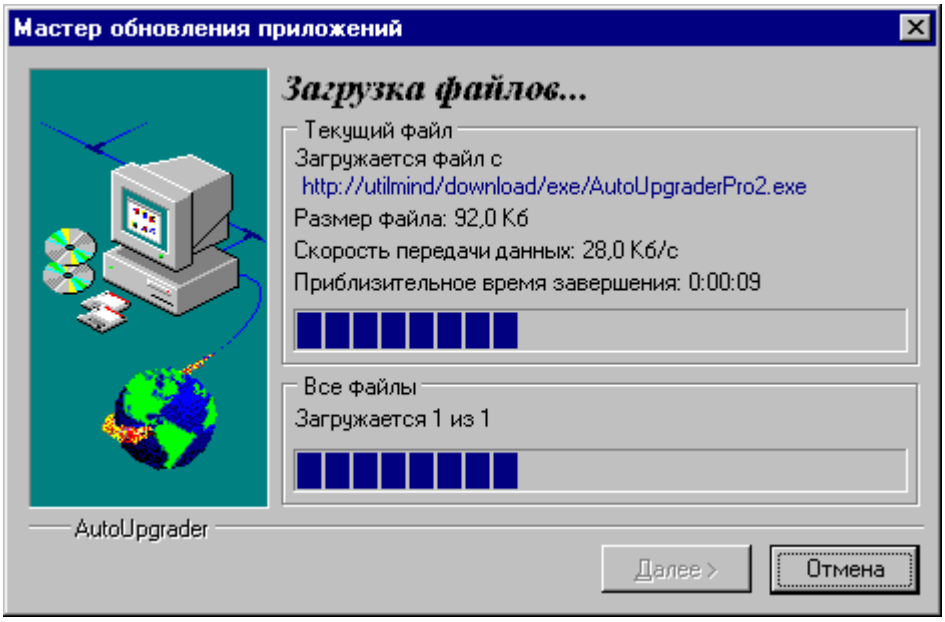


9.10.8.2 Multi-language support

Screenshot #2
"Application update wizard" in German

...and in Russian

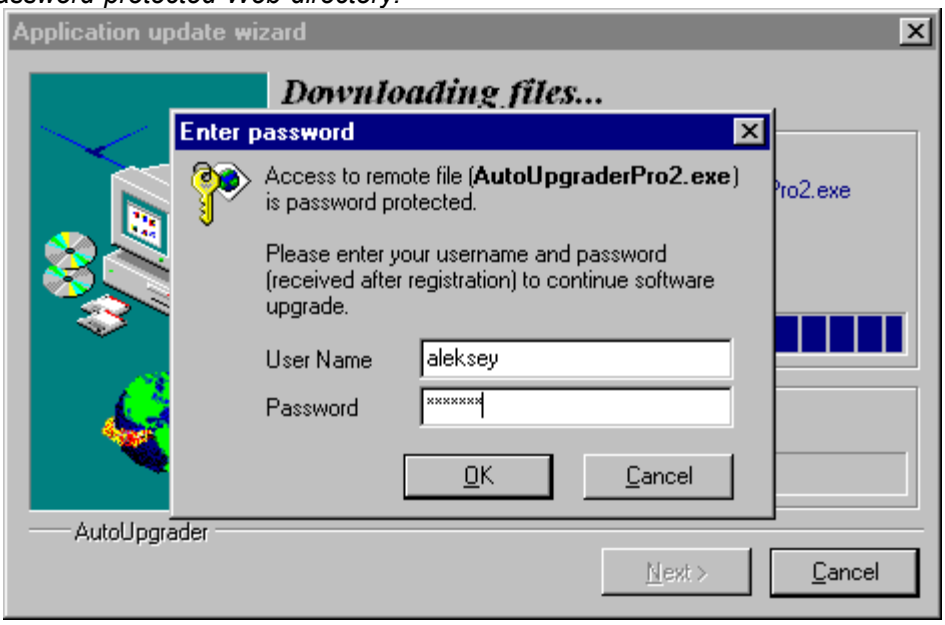
© 1998-2006, UtilMind Solutions®



BTW, all other messages (not only in Wizard) will be translated as well.

9.10.8.3 Access to protected area

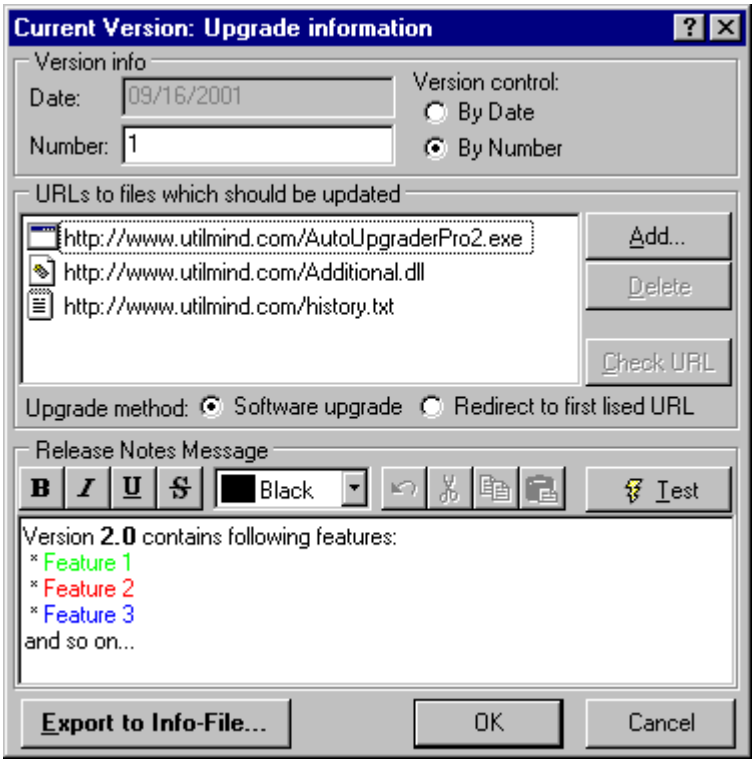
Screenshot #3
AutoUpgrader will ask user his username and password on necessity to download files from password protected Web directory.



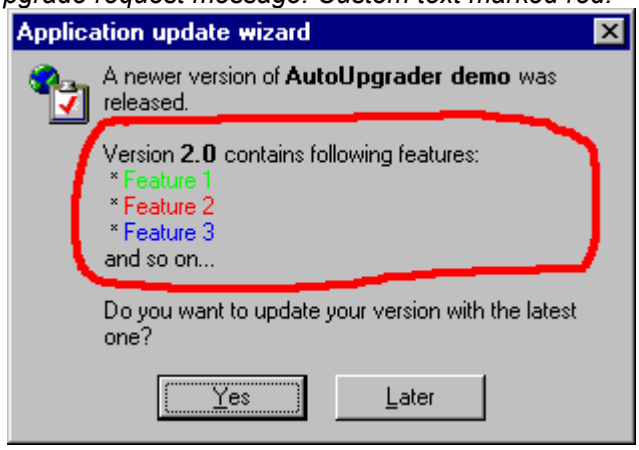
9.10.8.4 Info-file designer

Screenshot #4
Built-in Info-file designer will help you to create files with upgrade information.

© 1998-2006, UtilMind Solutions®



Upgrade request message. Custom text marked red:



9.11 acAwayTimer

9.11.1 TacAwayTimer

Overview

The acAwayTimer component monitors user activity and increase its time counter when keyboard and mouse are inactive (user distracted from PC or goes away). Also it triggers special event when system is about to start the screensaver, or after some defined period of user inactivity.

Advantages ?

You can execute some specific actions when user away from PC. For example, automatically save file, re-index the database etc. If you are writing Internet communication tool, you can switch the status of presence to "inactive" (Sleep/Away mode, like it does most Inernet Conferencing and

© 1998-2006, UtilMind Solutions®

Instant Messenger tools like ICQ, AOL, MSN or WinJabber), to inform user on the other side that user is temporary away.

How does it works and how to use ?

When the Active property is True, the acAwayTimer checks the mouse movements and keyboard presses with time interval specified in MonitorInterval property (in milliseconds).

When component does not detects any activity (user doesn't use mouse and keyboard for some time), it increases special time counter which can be used to determinate how much time user was inactive (temporary away from PC, went to smoke, have break for coffee etc). You can determinate the time of inactivity reading AwayTime property. This is read-only property, however if you want to reset the AwayTime property programmatically — use ResetAway method.

If you want to perform some specific actions after some time of inactivity — specify required time period (in units of milliseconds) to SleepInterval property and write OnSleep event handler. To be notified when user became active again (for example to return "Away" status to previous) — write OnAwake even handler.

Also, the acAwayTimer detects when the screen saver starts and become deactivated. To be notified about screen saver presence — write OnScreenSaverStart and OnScreenSaverEnd events.

See also

acAccurateTimer and acCronJob components.

9.11.2 Properties

9.11.2.1 Active

Applies to

acAwayTimer, acAccurateTimer, acCronJob, acCursorLocker, acFormBackground, acFormHelp, acFormMagnet, acFormSystemMenu and acIESniffer components.

Declaration

```
property Active: Boolean; // True by default
```

Description

The Active property controls whether the acAwayTimer is active to monitor the system for user activity.

Set Active property to True to enable the timer and check system for mouse movements and keyboard state, othwise set it to False to disable the monitoring timer.

The detection are performed with intervals specified in MonitorInterval property.

See also

MonitorInterval property.

9.11.2.2 AwayTime

Applies to

acAwayTimer component.

Declaration

```
property AwayTime: Cardinal; // read-only. time in milliseconds
```

Description

© 1998-2006, UtilMind Solutions®

The AwayTime is the read-only property used to determinate the time interval (in milliseconds) passed after last user activity (i.e: number of milliseconds passed after last mouse movement or last key press).

See also
MonitorInterval property and ResetAway method.

9.11.2.3 MonitorInterval

Applies to
acAwayTimer and acIESniffer components.

Declaration
```
property MonitorInterval: Integer; // 100 milliseconds by default
```

Description
The MonitorInterval property specifies the time intervals, in units of milliseconds, between monitoring the system for mouse movements and keyboard state, if Active property is True.

By default, the MonitorInterval set to 100, that means that component is checking the user activity every 100 milliseconds (10 times per second).

See also
Active property.

9.11.2.4 SleepInterval

Applies to
acAwayTimer component.

Declaration
```
property SleepInterval: Cardinal; // 5 minutes by default
```

Description
The SleepInterval property specifies the time interval (in milliseconds), which should pass since last user activity till execution of the OnSleep event.

For example, the OnSleep event will occur in 5 minutes of user inactivity (if user did not moved mouse and did not pressed keyboard buttons), if SleepInterval is 300000 (5 minutes = 300 seconds = 300000 milliseconds).

See also
OnSleep event.

9.11.3 Methods

9.11.3.1 ResetAway

Applies to
acAwayTimer component.

Declaration
```
procedure ResetAway;
```

Description
The ResetAway method reduces the AwayTime variable to zero.

© 1998-2006, UtilMind Solutions®

See also
AwayTime property.

9.11.4 Events

9.11.4.1 OnActiveTimer

Applies to
acAwayTimer component.

Declaration
```
property OnActiveTimer: TNotifyEvent;
```

Description
The OnActiveTimer event is like OnTimer event of usual TTimer component, but occurs only when user actively using his PC (until acAwayTimer component not detected that user "sleeping").

This event allows to perform some specific actions, with interval specified in MonitorInterval property, but ONLY when user IS ACTIVE (for example you can show him some popup ads or something :-)). This event never occurs when computer is inactive.

See also
Active and MonitorInterval property.

9.11.4.2 OnScreenSaverStart

Applies to
acAwayTimer component.

Declaration
```
property OnScreenSaverStart: TNotifyEvent;
```

Description
The OnScreenSaverStart event occurs when the component detects that screen saver program became active.

Example *(this will play sound on starting the screen saver)*
```
procedure TForm1.acAwayTimer1ScreenSaverStart(Sender: TObject);
begin
  acWavPlayer1.Play;
end;
```

See also
OnScreenSaverEnd event.

9.11.4.3 OnScreenSaverEnd

Applies to
acAwayTimer component.

Declaration
```
property OnScreenSaverEnd: TNotifyEvent;
```

Description
The OnScreenSaverEnd event occurs when the component detects that screen saver program was deactivated by user.

© 1998-2006, UtilMind Solutions®

See also
OnScreenSaverStart event.

9.11.4.4 OnSleep

Applies to
acAwayTimer component.

Declaration
```
property OnSleep: TNotifyEvent;
```

Description
The OnSleep event occurs after some period of user inactivity, specified in SleepInterval property.

For example, the OnSleep event will occur in 5 minutes of user inactivity (if user did not moved mouse and did not pressed keyboard buttons), if SleepInterval property is 300000 (5 minutes = 300 seconds = 300000 milliseconds).

You can use OnSleep event to execute some specific actions when user away from PC. For example, automatically save file, re-index the database etc. If you are writing Internet communication tool, you can switch the status of presence to "inactive" (Sleep/Away mode, like it does most Inernet Conferencing and Instant Messenger tools like ICQ, AOL, MSN or WinJabber), to inform user on the other side that user is temporary away. If you wish to return the presence status to previous state when user become active — use OnAwake event handler.

See also
SleepInterval property and OnAwake event.

9.11.4.5 OnAwake

Applies to
acAwayTimer component.

Declaration
```
property OnAwake: TNotifyEvent;
```

Description
The OnAwake event occurs when user awake (become active) after some period of inactivity, specified in SleepInterval property.

Use OnAwake event to return the application status to previous state after "sleeping" (Away mode). To be notified when user inactive after some time period — write OnSleep event handler.

See also
SleepInterval property and OnSleep event.

9.12 acCaptionButton

9.12.1 TacCaptionButton

Overview
The acCaptionButton component applies an additional custom button to the title bars of your forms + special menu item associated with this caption button to the system menu.

Additional caption buttons is completely customizable and stand in close integration with the SystemMenu and, as any standard button on the title bar have built-in tool tips (Hint property).

© 1998-2006, UtilMind Solutions®

However, unlike the standard buttons which built-in to any window, acCaptionButton have a lot of incredible additional features! See description and snapshots below.

Flexible Customization

The order of button on the title bar specified by BtnOrder property, but you can also separate buttons by any width of blank space specified in SeparatorWidth property.

The image on caption button can be either Sign (you may specify the SignFont) or Glyph bitmap image. You can also specify the glyph image for button in pressed state (GlyphDown property).

If you would like to allow user press the caption button - make AllowPress property True. If you would like to assign to the caption button any popup menu - welcome!, the required menu item will be available also in system menu at run-time.

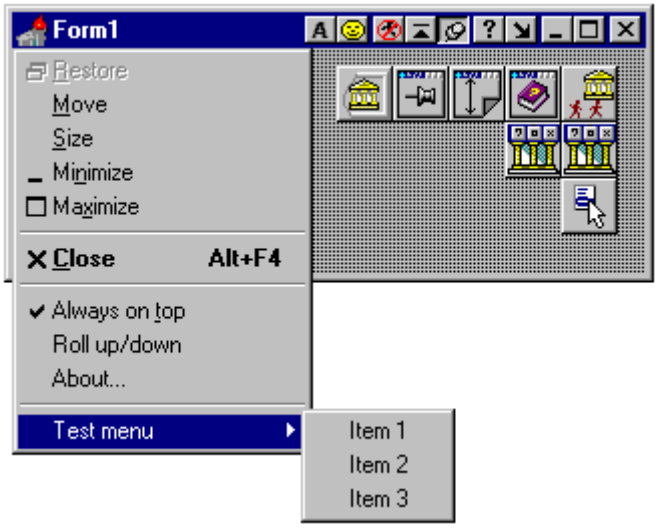
Any button can be in enabled (usual state) or disabled (user could not press it) state - see Enabled property, or invisible on the title bar at all - see Visible property.

All these and much more cool features could be specified at design-time and does not require ANY line of additional code!

See also CaptionButton's events for handling them.

Snapshot 1 *(tool tips)*:

Snapshot 2 *(integration with system menu)*:



Snapshot 3 *(buttons with popup menu)*:

© 1998-2006, UtilMind Solutions®

9.12.2 Properties

9.12.2.1 AllowPress

Applies to
acCaptionButton component and other components that uses TacCustomCaptionButton as ancestor class - acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller.

Declaration
```
property AllowPress: Boolean;
```

Description
The AllowPress property controls whether CaptionButton could be pressed (if button pressed, Down property will be True).

If AllowPress is True, the button will have two states - pressed and unpressed (up and down state), which can be controlled by Down property. Also two additional events will occurs - OnPressed on button pressing and OnReleased on releasing.

See also
Down property and OnClick, OnPressed, OnReleased events.

9.12.2.2 BtnOrder

Applies to
acCaptionButton component and other components that uses TacCustomCaptionButton as ancestor class - acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller.

Declaration
```
type
 TacBtnOrder = -1..32767;

property BtnOrder: TacBtnOrder;
```

Description
The BtnOrder property indicates position of caption button, the order in which buttons will be displayed. BtnOrder calculates from right side of title bar to left side (see snapshot below).

Initially, the button order is always the order in which the components were added to the form, but you can change this by changing the BtnOrder property. The value of the BtnOrder property is unique for each ancestor of the acCaptionButton on the form (like TabOrder property of ancestors of TWinControl). The first caption button added to the form has a BtnOrder value of 0, the second is 1, the third is 2, and so on. These values determine where a control is on the title bar.

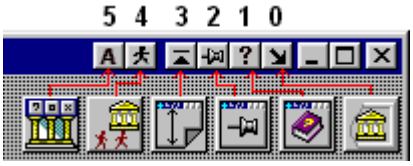
If you attempt to give a button a BtnOrder value greater than the number of caption buttons on the form minus one (because numbering starts with 0), AppControls won't accept the new value, but will enter the value that assures the component will be the last in the button order.

The caption button with the BtnOrder value of 0 is the custom button which placed on right side of title bar, after regular system buttons.

To separate the buttons by additional empty space — use SeparatorWidth property. For example snapshot below shows that acAppAutoRun component have BtnOrder = 4 (fifth position from the left of system buttons) and SeparatorWidth = 4 pixels.

Snapshot (orders of buttons)

© 1998-2006, UtilMind Solutions®



See also
SeparatorWidth property.

9.12.2.3 Cursor

Applies to
acCaptionButton component and other components that uses TacCustomCaptionButton as ancestor class - acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller.

Declaration
```
property Cursor: TCursor;
```

Description
The Cursor property controls the mouse cursor shape used when the mouse moves over CaptionButton in released state.

See also
CursorDown property.

9.12.2.4 CursorDown

Applies to
acCaptionButton component and other components that uses TacCustomCaptionButton as ancestor class - acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller.

Declaration
```
property CursorDown: TCursor;
```

Description
The CursorDown property controls the mouse cursor shape used when the mouse moves over CaptionButton in pressed state (Down is True).

See also
Cursor, Down and AllowPress properties.

9.12.2.5 Down

Applies to
acCaptionButton component and other components that uses TacCustomCaptionButton as ancestor class - acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller.

Declaration
```
property Down: Boolean;
```

Description
The Down property controls whether the button is actually pressed (if True) or unpressed (False).

However, if CaptionButton does not allow pressing, you will unable to change this property, and Down will always False.

© 1998-2006, UtilMind Solutions®

See also
AllowPress property.
OnClick, OnPressed, OnReleased events.

9.12.2.6 Enabled

Applies to
acCaptionButton component and other components that uses TacCustomCaptionButton as ancestor class - acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller.

Declaration
```
property Enabled: Boolean;
```

Description
The Enabled property controls whether the button on the title bar responds to mouse and keyboard messages. If Enabled is True, button responds normally. If Enabled is False, button become disabled and user can not press that button.

See also
Visible property.

9.12.2.7 Glyph

Applies to
acCaptionButton component and other components that uses TacCustomCaptionButton as ancestor class - acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller.

Declaration
```
property Glyph: TBitmap; // 12x11 pixels optionally
```

Description
The Glyph property specifies the bitmap that appears on the selected CaptionButton.

If Glyph image is not specified (bitmap is empty), then CaptionButton will show the character, specified in Sign property to show it as button image.

Use the Open dialog box that appears as an editor in the Object Inspector to choose a bitmap file (with a .BMP extension) to use on the button, or specify a bitmap image at run-time.

Note
The size of bitmap usually does not matter because the component will stretch image so it exactly fits the bounds of the button, however, optionally, please use bitmap images with **12x11** pixels resolution.

See also
GlyphDown property.

9.12.2.8 GlyphDown

Applies to
acCaptionButton component and other components that uses TacCustomCaptionButton as ancestor class - acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller.

Declaration
```
property GlyphDown: TBitmap; // 12x11 pixels optionally
```

© 1998-2006, UtilMind Solutions®

Description

The GlyphDown property specifies the bitmap that appears on the selected CaptionButton in pressed state (when Down property is True or when the user clicks this caption button).

If GlyphDown property is not specified (bitmap is empty), Glyph image will used for both button states (pressed and released). If Glyph image is not specified too, then will show the character, specified in Sign property to show it as button image.

Use the Open dialog box that appears as an editor in the Object Inspector to choose a bitmap file (with a .BMP extension) to use on the button, or specify a bitmap image at run-time.

Note

The size of bitmap usually does not matter because the component will stretch image so it exactly fits the bounds of the button, however, optionally, please use bitmap images with **12x11** pixels resolution.

See also

AllowPress, Down, Glyph properties.
OnPressed, OnReleased events.

9.12.2.9 GlyphTransparent

Applies to

acCaptionButton component and other components that uses TacCustomCaptionButton as ancestor class - acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller.

Declaration

```
property GlyphTransparent: Boolean;
```

Description

The GlyphTransparent property controls whether the background of the Glyph images should be transparent (if True). Set GlyphTransparent to False if you would like to show the Glyph with original colors, including the background.

See also

Glyph and GlyphDown properties.

9.12.2.10 Hint

Applies to

acCaptionButton component and other components that uses TacCustomCaptionButton as ancestor class - acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller.

Declaration

```
property Hint: String;
```

Description

The Hint property is the text string that can appear when mouse pointer moves over current CaptionButton.

If the CaptionButton's ShowHint property is False, the Help Hint won't appear, but the other hints still will.

Snapshot

© 1998-2006, UtilMind Solutions®

See also
ShowHint property.

9.12.2.11PopupMenu

Applies to
acCaptionButton component and other components that uses TacCustomCaptionButton as ancestor class - acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller.

Declaration
```
property PopupMenu: TPopupMenu;
```

Description
The PopupMenu property identifies the pop-up menu that appears when the user press the button on the form's title bar. When PopupMenu specified, same menu items will appears in the form's system menu.

If PopupMenu is not specified, button will behave as usual button without popup menu.

Snapshot *(button with popup menu)*

See also
SystemMenu properties.

9.12.2.12SeparatorWidth

Applies to
acCaptionButton component and other components that uses TacCustomCaptionButton as ancestor class - acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller.

Declaration
```
property SeparatorWidth: Byte; // in screen pixels
```

Description
The SeparatorWidth property controls the space between current and previous by order caption button in screen pixels.

See also
BtnOrder property.

9.12.2.13ShowHint

Applies to
acCaptionButton component and other components that uses TacCustomCaptionButton as ancestor class - acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller.

© 1998-2006, UtilMind Solutions®

Declaration

```
property ShowHint: Boolean;
```

Description

The ShowHint property controls whether acCaptionButton can show the Hint string when mouse pointer moves over button. If ShowHint is True, Hint will appears.

See also

Hint property.

9.12.2.14Sign

Applies to

acCaptionButton component and other components that uses TacCustomCaptionButton as ancestor class - acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller.

Declaration

```
property Sign: Char;
```

Description

The Sign property contains the character that will be shown as button image. You may choose custom font for this sign (SignFont property) and ever shift the sign accordingly to SignLeft and SignTop values.

Note

In case if the bitmap image is assigned to Glyph property, the CaptionButton will show the Glyph image instead of the sign. In this case, values assigned to Sign, SignFont, SignLeft and SignTop properties will be NOT used.

See also

SignFont, SignLeft, SignTop properties.

9.12.2.15SignFont

Applies to

acCaptionButton component and other components that uses TacCustomCaptionButton as ancestor class - acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller.

Declaration

```
property SignFont: TFont;
```

Description

The SignFont property is a font object that controls the attributes of the button's Sign, if you don't using glyph images.

See also

Sign property.

9.12.2.16SignLeft

Applies to

acCaptionButton component and other components that uses TacCustomCaptionButton as ancestor class - acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller.

Declaration

```
property SignLeft: Integer;
```

© 1998-2006, UtilMind Solutions®

Description

The SignLeft property controls the horizontal shifting for the Sign character, relatively to the button center. For example, SignLeft = **1** means that Sign character will be shifted to the right side of button at one pixel. SignLeft = **-1** means that Sign will be shifted at one pixel to the left side.

See also

SignTop, Sign and SignFont properties.

9.12.2.17SignTop

Applies to

acCaptionButton component and other components that uses TacCustomCaptionButton as ancestor class - acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller.

Declaration

```
property SignTop: Integer;
```

Description

The SignTop property controls the vertical shifting for the Sign character, relatively to the button center. For example, SignTop = **1** means that Sign character will be shifted to the right side of button at one pixel. SignTop = **-1** means that Sign will be shifted at one pixel to the left side.

See also

SignLeft, Sign and SignFont properties.

9.12.2.18SystemMenu

Applies to

acCaptionButton component and other components that uses TacCustomCaptionButton as ancestor class - acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller.

Declaration

```
property SystemMenu: TacSystemMenu;
```

Description

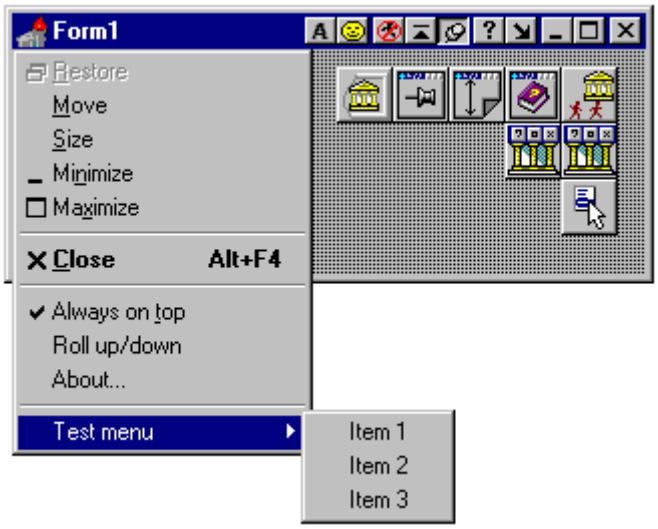
The System Menu is the popup menu that appears when you click on the program icon on the upper-left corner of form's title bar. The SystemMenu property is the list of properties that manages the menu item associated with current button on the form's title bar.

Properties

ApplyToMenu	applies or removes the menu item associated with current button from the system menu;
Caption	text for the title of menu item;
Position	position of current menu item in the system menu;
Separators	specifies the separators for menu item, to separatate it from previous and / or next menu item of the system menu.

Snapshot

© 1998-2006, UtilMind Solutions®



See also
acFormSystemMenu component.

9.12.2.18.1 ApplyToMenu

Applies to
acCaptionButton component and all other components which uses CaptionButton as base ancestor class - acAppAutoRun, acTrayIcon, acFormTopmost, acFormRoller. ApplyToMenu is subproperty of SystemMenu.

Declaration
```
property ApplyToMenu: Boolean;
```

Description
The ApplyToMenu property controls whether the CaptionButton currently have the menu item assiciated with this button. Set ApplyToMenu property to True to add according menu item (with text specified by Caption property, in position specified by Position property) to the form's system menu and False to remove.

See also
Caption property.

9.12.2.18.2 Caption

Applies to
acCaptionButton component and all other components which uses CaptionButton as base ancestor class - acAppAutoRun, acTrayIcon, acFormTopmost, acFormRoller. Caption is subproperty of SystemMenu.

Declaration
```
property Caption: String;
```

Description
The Caption property specifies the text for menu item in the form's system menu, associated with current caption button on the title bar. If Caption is not specified, text from Hint property will be taken as item title.

© 1998-2006, UtilMind Solutions®

See also
Hint property.

9.12.2.18.3 Position

Applies to
acCaptionButton component and all other components which uses CaptionButton as base ancestor class - acAppAutoRun, acTrayIcon, acFormTopmost, acFormRoller. Position is subproperty of SystemMenu.

Declaration
```
property Position: Word;
```

Description
The Position property determines the position for menu item in the system menu associated with current caption button on the form's title bar. Change the Position value to move current menu item by system menu.

Maximum value for Position property is the current maximum number of items in the system menu (since orders for menu items starts with 0).

See also
Separators property.

9.12.2.18.4 Separators

Applies to
acCaptionButton component and all other components which uses CaptionButton as base ancestor class - acAppAutoRun, acTrayIcon, acFormTopmost, acFormRoller. Separators is subproperty of SystemMenu.

Declaration
```
type
  TacMenuSeparators = set of (seBefore, seAfter);

property Separators: TacMenuSeparators;
```

Description
The Separators property specifies the separators for current menu item to separate it from previous and / or next menu items of the system menu. To set separator before menu item — set `seBefore` to True. To set separator after menu item — set `seAfter` to True.

Specify Separators to separate current menu item in the system menu from another menu items.

See also
Position property.

9.12.2.19Visible

Applies to
acCaptionButton component and other components that uses TacCustomCaptionButton as ancestor class - acAppAutoRun, acFormHelp, acFormTopmost, acFormRoller.

Declaration
```
property Visible: Boolean;
```

Description

© 1998-2006, UtilMind Solutions®

The Visible property determines whether the caption button currently visible on the title bar. Make Visible property True if you would like to show the button on title bar, or False if you would like to hide the button.

When you changing the Visible property, placements of others button will automacially recalculated and other buttons with higher BtnOrder will be shifted to the right side of the title bar, filling the empty space.

See also
Enabled property.

9.12.3 Methods

9.12.3.1 RefreshButton

Applies to
acCaptionButton component and all successors of TacCustomCaptionButton (acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost and acFormRoller).

Declaration
```
procedure RefreshButton;
```

Description
The RefreshButton method repaints the custom caption button on the title bar. For example, RefreshButton method automatically executes when caption text changes.

See also
RefreshNCArea method.

9.12.3.2 RefreshNCArea

Applies to
acCaptionButton component and all successors of TacFormHookComponent class (i.e: acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost and acFormRoller).

Declaration
```
procedure RefreshNCArea;
```

Description
The RefreshNCArea method repaints all non-client parts of window (i.e. frames, caption buttons etc). RefreshNCArea method automatically executes when buttons should recalculate their placement after changing of BtnOrder or SeparatorWidth values.

See also
RefreshButton method.

9.12.3.3 ShowParentForm

Applies to
acCaptionButton component and all successors of TacComponent class (i.e: acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost and acFormRoller).

Declaration
```
procedure ShowParentForm;
```

Description
The ShowParentForm method brings the parent form to front over other windows, and restores it

© 1998-2006, UtilMind Solutions®

from task bar if it was minimized.

9.12.4 Events

9.12.4.1 OnCaptionDblClick

Applies to
acCaptionButton and acTrayIcon components.

Declaration
```
property OnCaptionDblClick: TNotifyEvent;
```

Description
The OnCaptionDblClick event notifies when user double clicks the caption of the form (not button).

See also
OnClick event.

9.12.4.2 OnClick

Applies to
acCaptionButton component.

Declaration
```
property OnClick: TNotifyEvent;
```

Description
The OnClick event occurs when the user clicks the button on title bar **or** selects the menu item associated with this caption button in the SystemMenu.

Use OnButtonClick and OnMenuClick events to identify the source of mouse click, and OnPressed / OnReleased events to make some specific operations when user press and release button (if AllowPress = True).

See also
OnButtonClick, OnMenuClick, OnPressed and OnReleased events;

9.12.4.3 OnButtonClick

Applies to
acCaptionButton component.

Declaration
```
property OnButtonClick: TNotifyEvent;
```

Description
The OnButtonClick event occurs when the user clicks the button on title bar.

If you would like to hook the event when user either clicks the button or selects menu item in the system menu — use OnClick event instead of this.

See also
OnClick, OnMenuClick, OnPressed and OnReleased events.

© 1998-2006, UtilMind Solutions®

9.12.4.4 OnMenuClick

Applies to
acCaptionButton component.

Declaration
```
property OnMenuClick: TNotifyEvent;
```

Description
The OnMenuClick event occurs when the user selects the menu item, associated with this caption button, in the system menu.

If you would like to hook the event when user either clicks the button or selects menu item in the system menu — use OnClick event instead of this.

See also
OnClick, OnButtonClick, OnPressed and OnReleased events.

9.12.4.5 OnPressed

Applies to
acCaptionButton component.

Declaration
```
property OnPressed: TNotifyEvent;
```

Description
The OnPressed event occurs when the user clicks the the unpressed CaptionButton on title bar or selects in the SystemMenu the unchecked menu item associated with this button.

When user clicks the button which could be pressed (this controlled by AllowPress property), the button state will changed to down, Down property will be True and associated menu item in the system menu will be checked (if SystemMenu.ApplyToMenu is True).

See also
AllowPress, Down, SystemMenu properties.
OnClick and OnReleased events.

9.12.4.6 OnReleased

Applies to
acCaptionButton component.

Declaration
```
property OnReleased: TNotifyEvent;
```

Description
The OnReleased event occurs when the user clicks the the pressed CaptionButton on title bar or selects in the SystemMenu the checked menu item associated with this button.

When user clicks (unpress) the button which could be pressed (this controlled by AllowPress property), the button state will changed to up, Down property will be False and associated menu item in the system menu will be unchecked (if SystemMenu.ApplyToMenu is True).

See also
AllowPress, Down, SystemMenu properties.

© 1998-2006, UtilMind Solutions®

OnClick and OnPressed events.

9.13 acClipboard

9.13.1 acClipboard

Overview

The acClipboard component monitors the clipboard for changes and triggers special event when the content of the clipboard changes. Also it provides with super easy way to retrieve and put to clipboard the Bitmap images, plain or unicode text and list of files (the bitmap, plain text or file list are usual properties, so the clipboard content can be modified even at design-time!). In case if you are working with some more complicated or custom clipboard formats, you can use several special methods which can help you to access the clipboard just like you working with usual strings or streams.

How to use?

To monitor the clipboard for changes — drop the component to your form and write OnChange event handler (Active property should be True for monitoring).

The component provides easy access to the bitmap image, plain text and list of files in the clipboard. You can work with bitmap, text and file list just as usual properties. To retrieve or put these values to clipboard — use ClipboardBitmap, ClipboardText and ClipboardFiles properties.

To check which formats currently are available in the clipboard — you can use out AvailableFormats property or check whether necessary format available using HasFormat function.

To retrieve the raw data from clipboard as string or stream — use GetString or GetStream methods. To put the data to clipboard — use PutString or PutStream. To erase all the content of clipboard — call Clear method.

Example

```
// Demonstrates how to be notified when the content of the clipboard
changes
// and refresh the "Paste" menu items when some format appears or
disappears from the clipboard.
procedure TacFormHelpEditor.acClipboard1Change(Sender: TObject);
begin
  PasteTextItem.Enabled := acClipboard1.HasFormat(CF_TEXT);
  // here is an alternative
  PasteBitmapItem := acClipboard1.HasFormat('BITMAP'); // <-- you can
specify format name as string
end;
```

9.13.2 Properties

9.13.2.1 Active

Applies to

acClipboard

Declaration

```
property Active: Boolean;
```

Description

The Active property determines whether the component are monitoring the clipboard for changes.

© 1998-2006, UtilMind Solutions®

Set Active to True if you wish to monitor the clipboard for changes, or set it to False to disable monitoring. When Active is True and some data in the clipboard changes, the component triggers the OnChange event.

See also
AvailableFormats property;
OnChange event.

9.13.2.2 AvailableFormats

Applies to
acClipboard

Declaration
```
property AvailableFormats: TStrings;
```

Description
The AvailableFormat property retrieves the list of formats (format names) which currently are available at the clipboard.

To convert the format name to format index — use GetFormatIndex method. You can also retrieve the raw data of any format to string or stream using GetString or GetStream methods.

Example
```
// retrieving data from clipboard format by format name
SomeDataFromClipboard := GetString(AvailableFormats[0]);
```

See also
ClipboardBitmap, ClipboardFiles and ClipboardText properties;
HasFormat, Clear, GetString, GetStream, PutString, PutStream, GetFormatIndex and GetFormatName methods;
OnChange event.

9.13.2.3 ClipboardBitmap

Applies to
acClipboard

Declaration
```
property ClipboardBitmap: TBitmap;
```

Description
The ClipboardBitmap property used to retrieve or put the bitmap image to the clipboard.

You can read or modify the ClipboardBitmap as usual TBitmap object (you can even paint by canvas of ClipboardBitmap at run-time, or load image at design-time) and the clipboard will imediately reflect all changes (also triggering OnChange event, if Active property is True).

Note
The bitmap images have standard clipboard format, registered by 'BITMAP' name, and have CF_BITMAP format index, plus have the palette, usually stored to clipboard together with image. The bitmap palette is also standard format, named 'PALETTE', and have CF_PALETTE index.

See also
Active, AvailableFormats, ClipboardFiles and ClipboardText properties;
HasFormat, Clear, GetString and PutString methods;

© 1998-2006, UtilMind Solutions®

OnChange event.

9.13.2.4 ClipboardFiles

Applies to
acClipboard

Declaration
```
property ClipboardFiles: TStrings;
```

Description
The ClipboardFiles property used to retrieve or put the file name or list of files/folders which can be copied by user from Windows Explorer and ready to be pasted somewhere.

You can read or modify the ClipboardFiles as usual string list (even at design-time) and the clipboard will imediately reflect all changes (also triggering OnChange event, if Active property is True).

Note
The file list is standard clipboard format registered by 'HDROP' name, and have CF_HDROP format index. The Windows shell uses this clipboard format when copying the files from one location to another, using content menu of folder or drag and drop method.

See also
Active, AvailableFormats, ClipboardBitmap and ClipboardText properties;
HasFormat, Clear, GetString and PutString methods;
OnChange event.

9.13.2.5 ClipboardText

Applies to
acClipboard

Declaration
```
property ClipboardText: String;
```

Description
The ClipboardText property used to retrieve or put the plain text to the clipboard.

You can read or modify the ClipboardText as usual string property (even at design-time) and the clipboard will imediately reflect all changes (also triggering OnChange event, if Active property is True).

Alternatively you can retrieve the plain text from clipboard using the GetString method. Here is an example:
```
TextFromClipboard := GetString(CF_TEXT);
```

Note
The plain text is standard clipboard format registered by 'TEXT' name, and have CF_TEXT format index.

See also
Active, AvailableFormats, ClipboardBitmap and ClipboardFiles properties;
HasFormat, Clear, GetString and PutString methods;
OnChange event.

© 1998-2006, UtilMind Solutions®

9.13.3 Methods

9.13.3.1 Clear

Applies to
acClipboard

Declaration
```
procedure Clear;
```

Description
The Clear method empties the clipboard and frees handles in data in the clipboard.

After calling the Clear method, the AvailableFormats property will show the empty list, so all data has been erased from the clipboard. You can put to clipboard new data using PutString or PutStream methods…

See also
AvailableFormats, ClipboardBitmap, ClipboardFiles and ClipboardText properties;
HasFormat, Clear, GetString and PutString methods;
OnChange event.

9.13.3.2 HasFormat

Applies to
acClipboard

Declaration
```
function HasFormat(const FormatName: String): Boolean; overload;
function HasFormat(FormatIndex: Integer): Boolean; overload;
```

Description
The HasFormat method determines if the clipboard contains a particular format. If HasFormat is True, the format is present; if False, the format is absent.

The acClipboard component keeps a list of available format names in the AvailableFormats property. If you need to get the format index by name you can use GetFormatIndex method, or GetFormatName to retrieve the name by index.

Some standard clipboard formats are listed below. However, there are many more clipboard formats provided by Windows, and custom formats can be registered. All are supported by HasFormat.

Value	Meaning
CF_TEXT	Plain text, with each line ending with a CR-LF combination. A null character identifies the end of the text.
CF_BITMAP	A Windows bitmap graphic
CF_METAFILEPICT	Windows metafile
CF_PICTURE	An object of type TPicture (VCL only)
CF_SYLK	Symbolic link
CF_DIF	Data interchange format
CF_TIFF	Tiff image
CF_OEMTEXT	OEM text
CF_DIB	DIB image

© 1998-2006, UtilMind Solutions®

CF_PALETTE	Palette data
CF_PENDATA	Pen data
CF_RIFF	Riff audio data
CF_WAVE	Wav audio data
CF_UNICODETEXT	Unicode text
CF_ENHMETAFILE	Enhanced metafile image
CF_HDROP	File name(s)
CF_LOCALE	Locale descriptor
CF_OBJECT	Any persistent object

Examples

```
procedure TacFormHelpEditor.acClipboard1Change(Sender: TObject);
begin
   PasteTextItem.Enabled := acClipboard1.HasFormat(CF_TEXT);
   // here is an alternative
   PasteBitmapItem := acClipboard1.HasFormat('BITMAP'); // <-- you can
specify format name as string
end;

procedure TForm1.GetUnicodeExample;
var
  Str: String;
begin
  with acClipboard1 do
   if HasFormat(CF_UNICODETEXT) then
     Str := GetString(CF_UNICODETEXT);
   else
     Str := '';
end;
```

See also

Active, AvailableFormats, ClipboardBitmap, ClipboardFiles and ClipboardText properties;
Clear, GetString, PutString, GetStream, PutStream, GetFormatIndex and GetFormatName methods;
OnChange event.

9.13.3.3 GetFormatIndex

Applies to

acClipboard

Declaration

```
function GetFormatIndex(const FormatName: String;
RegisterIfNotAvailable: Boolean = False): Integer;
```

Description

The GetFormatIndex property returns the index of the clipboard format by its name.

If the specified format name is not registered in the system, and *RegisterIfNotAvailable* parameter is True, the function registers new format and returns the index of registered format. Otherwise, if the specified FormatName does not exists, the function returns 0.

To receive the list of all formats that currently are available in the clipboard — read

© 1998-2006, UtilMind Solutions®

AvailableFormats property.

See also
AvailableFormats property;
HasFormat and GetFormatName methods.

9.13.3.4 GetFormatName

Applies to
acClipboard

Declaration
```
function GetFormatName(FormatIndex: Integer): String;
```

Description
The GetFormatName property retrieves the name of the clipboard format by its index.

Function may return the empty string if no format has been registered by specified *FormatIndex*.

To receive the list of all formats that currently are available in the clipboard — read AvailableFormats property.

See also
AvailableFormats property;
HasFormat and GetFormatIndex methods.

9.13.3.5 GetStream

Applies to
acClipboard

Declaration
```
function GetStream(const FormatName: String; Stream: TStream): Boolean;
overload;
function GetStream(FormatIndex: Integer; Stream: TStream): Boolean;
overload;
```

Description
The GetStream method retrieves the data from the clipboard to the *Stream* (the Stream object must be previously created in the caller object or method). The format name or format index must be specified in *FormatName*, or *FormatIndex* parameter.

See also
Active, AvailableFormats, ClipboardBitmap, ClipboardFiles and ClipboardText properties;
HasFormat, Clear, GetString, PutString, GetStream, PutStream, GetFormatIndex and GetFormatName methods;
OnChange event.

9.13.3.6 GetString

Applies to
acClipboard

Declaration
```
function GetString(const FormatName: String): String; overload;
function GetString(FormatIndex: Integer): String; overload;
```

© 1998-2006, UtilMind Solutions®

Description
The GetStream method retrieves the data from the clipboard to the *Stream*. The format name or format index must be specified in *FormatName*, or *FormatIndex* parameter.

Example
```
TextFromClipboard := GetString(CF_TEXT);
```

See also
Active, AvailableFormats, ClipboardBitmap, ClipboardFiles and ClipboardText properties;
HasFormat, Clear, GetString, PutString, GetStream, PutStream, GetFormatIndex and GetFormatName methods;
OnChange event.

9.13.3.7 PutStream

Applies to
acClipboard

Declaration
```
procedure PutStream(const FormatName: String; Stream: TStream);
overload;
procedure PutStream(FormatIndex: Integer; Stream: TStream); overload;
```

Description
The PutStream method copies the data from *Stream* to the clipboard. The format name or format index must be specified in *FormatName*, or *FormatIndex* parameter.

See also
Active, AvailableFormats, ClipboardBitmap, ClipboardFiles and ClipboardText properties;
HasFormat, Clear, GetString, PutString, GetStream, PutStream, GetFormatIndex and GetFormatName methods;
OnChange event.

9.13.3.8 PutString

Applies to
acClipboard

Declaration
```
procedure PutString(const FormatName, Data: String); overload;
procedure PutString(FormatIndex: Integer; const Data: String); overload;
```

Description
The PutString method copies the text specified in the *Data* parameter to the clipboard. The format name or format index must be specified in *FormatName*, or *FormatIndex* parameter.

See also
Active, AvailableFormats, ClipboardBitmap, ClipboardFiles and ClipboardText properties;
HasFormat, Clear, GetString, PutString, GetStream, PutStream, GetFormatIndex and GetFormatName methods;
OnChange event.

© 1998-2006, UtilMind Solutions®

9.13.4 Events

9.13.4.1 OnChange

Applies to
acClipboard

Declaration
```
property OnChange: TNotifyEvent;
```

Description
The OnChange event occurs when the content of the clipboard changes.

Write the OnChange event handler to display the new content in the clipboard viewer window, or refresh the "Paste" menu items and buttons when the data with compatible format appears or disappears from the clipboard.

Note
The OnChange event only occurs if Active property is True. Set Active to True if you wish to monitor the clipboard for changes, or set it to False to disable monitoring.

Example
```
procedure TacFormHelpEditor.acClipboard1Change(Sender: TObject);
begin
  PasteTextItem.Enabled := acClipboard1.HasFormat(CF_TEXT);

  // here is an alternative
  PasteBitmapItem := acClipboard1.HasFormat('BITMAP'); // <-- you can
specify format name as string
end;
```

See also
Active and AvailableFormats properties;
HasFormat, GetString and PutString methods;
OnChange event.

9.14 acConnectionList

9.14.1 acConnectionList

Overview
The acConnectionList retrieves the list of all active Internet connection and returns information in the form similar to **NetStat** tool. You can retrieve the list using Refresh method and get access to the information about each connection, using the component as usual TList, with objects which contains detailed information about connection (local and remote port and host name, protocol, and even possible software behind the protocol).

How does it works and how to use?
The component is the container of the list of **TacConnection** objects. Each TacConnection object contains brief information about the currently established TCP or UDP connection (it's local and remote IP address and port number, connection state, protocol type, etc).

The acConnectionList component automatically contains the list of currently available connections when your application starts. However, to keep the list fresh, you should call Refresh method, so the component will re-retreive the list of active connections.

© 1998-2006, UtilMind Solutions®

Following example demonstrates how to retrieve the list of processes and display them in the ListView:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  I: Integer;
  ListItem: TListItem;
begin
  acConnectionList1.Refresh; // refreshing the list of processes

  ListView1.Items.BeginUpdate;
  try
    ListView1.Items.Clear;
    I := acConnectionList1.Count;
    if I <> 0 then
     for I := 0 to I - 1 do
      begin
        ListItem := ListView1.Items.Add;
         ListItem.Caption := acConnectionList1[I].LocalIPStr +
IntToStr(LocalPort));
        ListItem.SubItems.Add(acConnectionList1[I].RemoteIPStr +
 IntToStr(RemotePort));

        // ...optionally, let's associate the Data pointer to
 TacConnection object
        // and Data pointer of Connection object to ListItem
        ListItem.Data := acConnectionList1[I];
        acConnectionList1[I].Data := ListItem;
      end;
  finally
    ListView1.Items.EndUpdate;
  end;
end;

// This code demonstrates how to display the Remote IP address and
port
// when user selects the list item of ListView
procedure TForm1.ListView1SelectItem(Sender: TObject; Item: TListItem;
  Selected: Boolean);
begin
  if Item <> nil then
   with TacConnection(Item.Data) do
    Label1.Caption := LocalIPStr + ':' + IntToStr(LocalPort);
end;
```

See also

acProcessList and acHostnameResolver components.

9.14.2 Properties

9.14.2.1 AutoResolve

Applies to

acConnectionList and acHostnameResolver components.

Declaration

```
property AutoResolve: Boolean;
```

© 1998-2006, UtilMind Solutions®

Description

The AutoResolve property determines whether the component should lookup for hostname automatically at once after specifying the IPList or adding new IP address to that list.

When AutoResolve is True, the thread which resolves hostnames will be started automatically and application can receive retrieved IPs in OnHostnameRetrieved event. When AutoResolve is False, you should programmatically start the thread using Execute method to start the lookup for hostnames.

See also

IPList property;
Execute and Abort methods;
OnHostnameRetrieved event.

9.14.2.2 CacheSize

Applies to

acConnectionList and acHostnameResolver components.

Declaration

```
property CacheSize: Integer;
```

Description

The CacheSize property specifies the maximum number of IPs and resolved hostnames which can contain the internal cache of acHostnameResolver component.

The internal cache is the special list with IPs and Hostnames which are already retrieved. The component use this cache to avoid double resolving of hostname by particular IP address.

If you don't want to use cache, just set this property to 0.

The cached list can be automatically saved in registry for further use after application restarts. If you wish to store cache in the registry — just set RegistrySaver.Enabled property to True and specify the registry key which should hold records in RegistrySaver.RegKey property.

The component contains special method which helps to resolve cached IPs without executing the lookup thread. Use ResolveCachedIP to get the hostname by specified IP address.

See also

RegistrySaver structure;
ResolveCachedIP method.

9.14.2.3 ConnectionList

Applies to

acConnectionList component.

Declaration

```
type
  TacConnections = class(TacObjectList)
  public
    function FindConnectionByLocalPort(LocalPort: Word; Proto:
TacConnectionProto = pAny; ListeningPortOnly: Boolean = False):
TacConnection;
    function FindConnectionByRemotePort(RemotePort: Word; Proto:
TacConnectionProto = pAny): TacConnection;
```

© 1998-2006, UtilMind Solutions®

```
    function FindConnectionByLocalIPStr(const LocalIPStr: String; Proto:
TacConnectionProto = pAny; ListeningPortOnly: Boolean = False):
TacConnection;
    function FindConnectionByRemoteIPStr(const RemoteIPStr: String;
Proto: TacConnectionProto = pAny): TacConnection;

    function FindLocalIPStrByLocalPort(LocalPort: Word; Proto:
TacConnectionProto = pAny): String;
    function FindLocalIPStrByRemotePort(RemotePort: Word; Proto:
TacConnectionProto = pAny): String;
    function FindRemoteIPStrByLocalPort(LocalPort: Word; Proto:
TacConnectionProto = pAny): String;
    function FindRemoteIPStrByRemotePort(RemotePort: Word; Proto:
TacConnectionProto = pAny): String;

    function IsPortListening(Port: Word; Proto: TacConnectionProto =
pAny): Boolean;

    property Items[Index: Integer]: TacConnection;
  end;

property ConnectionList: TacConnections;
```

Description

The ConnectionList is the internal list which contains the information about each TCP/UDP connection.

The ConnectionList is descendant of TList object, and contains only TacConnection objects.

The component does not refreshes this list automatically. To refresh this list — use Refresh method.

See also

TacConnection object;
Refresh method.

9.14.2.4 Count

Applies to

acConnectionList and acProcessList components.

Declaration

```
property Count: Integer; // read-only!
```

Description

The Count is the read-only property which determines the number of Items (TacConnection objects) which are available in the list.

To get access to the list items — use Items property.

See also

Items property.

9.14.2.5 IsOnline

Applies to

acConnectionList and acHostnameResolver components.

© 1998-2006, UtilMind Solutions®

Declaration

```
property IsOnline: Boolean; // read-only!
```

Description

The IsOnline is the read-only property which determines current online status of machine. It returns True if computer connected to Internet or False otherwise.

The component is able to constantly monitor the connection and notify the application when computer goes online or offline in OnOnlineStatusChanged event. The status monitoring performed between time intervals, specified in MonitorInterval property, if MonitorActive is True. To disable monitoring, set MonitorActive property to False.

Remark

This property is useless on machines with Ethernet cable connection. IsOnline always returns True (online), even if user cut cable off. Unfortunately there is no convinient way to determinate whether computer is online. However, for sure you may try to ping some host or something. This property mostly good for Dial-up modem connections only.

See also

IsOnline, MonitorActive and MonitorInterval properties;
OnOnlineStatusChanged event.

9.14.2.6 Items

Applies to

acConnectionList component.

Declaration

```
property Items[Index: Integer]: TacConnection read Get; default; //
read-only!
```

Description

The Items is the *run-time only* property, which can be used to get an access to each instance of TacConnection object (which contains the information about the connection) at particular position. *Index* gives the position of the TacConnection object, where 0 is the position of the first connection, 1 is the position of the second connection, and so on ("Count - 1" is the position of last process in the list).

Items is the default property of acConnectionList component. The Items identifier can be omitted when accessing the TacConnection objects. For example, the following two lines of code are both acceptable and do the same thing:

```
FirstConnection := acConnectionList1.Items[0];
FirstConnection := acConnectionList1[0];
```

To determinate the number of items which currently are available in the list — read Count property.

Example

```
// Demonstrates how to retrieve the list of processes and display them
in the ListView:
procedure TForm1.Button1Click(Sender: TObject);
var
  I: Integer;
  ListItem: TListItem;
begin
  acConnectionList1.Refresh; // refreshing the list of processes
```

© 1998-2006, UtilMind Solutions®

```
  ListView1.Items.BeginUpdate;
  try
    ListView1.Items.Clear;
    I := acConnectionList1.Count;
    if I <> 0 then
     for I := 0 to I - 1 do
      begin
        ListItem := ListView1.Items.Add;
        ListItem.Caption := acConnectionList1[I].LocalIPStr +
IntToStr(LocalPort));
        ListItem.SubItems.Add(acConnectionList1[I].RemoteIPStr +
IntToStr(RemotePort));

        // ...optionally, let's associate the Data pointer to TacProcess
object
        // and Data pointer of Process object to ListItem
        ListItem.Data := acConnectionList1[I];
        acConnectionList1[I].Data := ListItem;
      end;
  finally
    ListView1.Items.EndUpdate;
  end;
end;
```

See also
Count property.

9.14.2.7 LocalHostname

Applies to
acConnectionList and acHostnameResolver components.

Declaration
```
property LocalHostname: String; // read-only!
```

Description
The LocalHostname is read-only property which determines the hostname of local machine. Also you can determinate its current IP address reading LocalIP property.

See also
LocalIP property.

9.14.2.8 LocalIP

Applies to
acConnectionList and acHostnameResolver components.

Declaration
```
property LocalIP: String; // read-only!
```

Description
The LocalIP is read-only property which determines the IP address of local machine (represented by text string). Also you can determinate the hostname of current machine reading LocalHostname property.

See also
LocalHostname property.

© 1998-2006, UtilMind Solutions®

9.14.2.9 MonitorActive

Applies to
acConnectionList and acHostnameResolver components.

Declaration
```
property MonitorActive: Boolean;
```

Description
The MonitorActive property specifies whether the component should monitor the online status of current PC (whether it connected to Internet).

The time periods between checking the online status should be specified in MonitorInterval property. To be notified when online status changed — write OnOnlineStatusChanged event handler.

Remark
Unfortunately component is unable to detect whether computer is not connected to Internet, if user are use Ethernet cable connection. The IsOnline property will always return True even if user cut cable off.

See also
IsOnline, MonitorActive and MonitorInterval properties;
OnOnlineStatusChanged event.

9.14.2.10 MonitorInterval

Applies to
acConnectionList and acHostnameResolver components.

Declaration
```
property MonitorInterval: Integer; // 100 milliseconds by default
```

Description
The MonitorInterval property specifies the time periods, in milliseconds, between checking the online status of computer, if MonitorActive property already set to True.

To be notified when online status changed — write OnOnlineStatusChanged event handler.

Remark
Unfortunately component is unable to detect whether computer is not connected to Internet, if user are use Ethernet cable connection. The IsOnline property will always return True even if user cut cable off.

See also
IsOnline, MonitorActive and MonitorInterval properties;
OnOnlineStatusChanged event.

9.14.2.11 Priority

Applies to
acConnectionList and acHostnameResolver components.

Declaration
```
property Priority: TThreadPriority;
```

© 1998-2006, UtilMind Solutions®

Description

Priority indicates the priority used when scheduling the thread (which retrieves host names by IP addresses). Adjust the priority higher or lower as needed.

TThreadPriority type defines the possible values for the Priority property of the acThread component, as defined in the following table. The system schedules CPU cycles to each thread based on a priority scale; the Priority property adjusts a thread's priority higher or lower on the scale.

Values	Meaning
tpIdle	The thread executes only when the system is idle. The system will not interrupt other threads to execute a thread with tpIdle priority.
tpLowest	The thread's priority is two points below normal.
tpLower	The thread's priority is one point below normal.
tpNormal	The thread has normal priority.
tpHigher	The thread's priority is one point above normal.
tpHighest	The thread's priority is two points above normal.
tpTimeCritical	The thread gets highest priority.

Warning

Boosting the thread priority of a CPU intensive operation may "starve" the other threads in the application. Only apply priority boosts to threads that spend most of their time waiting for external events.

See also

acThread component.

9.14.2.12 RegistrySaver

Applies to

acConnectionList and acHostnameResolver components.

Declaration

```
type
  TacHostnameResolverRegistrySaver = class
  published
    property Enabled: Boolean;
    property RegKey: String;
    property RegLocation: TacRegLocation;
  end;

property RegistrySaver: TacHostnameResolverRegistrySaver;
```

Description

The RegistrySaver structure used to save and restore the cached hostnames to/from system registry.

If you wish to save the internal cache and restore it on next allication restart — just specify the RegLocation and RegKey where to store these settings, and set Enabled to True.

9.14.2.13 RetrieveTCP

Applies to

acConnectionList component.

Declaration

© 1998-2006, UtilMind Solutions®

```
property RetrieveTCP: Boolean; // read-only!
```

Description

The RetrieveTCP property specifies whether the component should retreive the TCP connections to its internal list of available connections.

The list can be refreshed using Refresh method.

See also

RetreiveUDP and ConnectionList properties;
Refresh method.

9.14.2.14 RetrieveUDP

Applies to

acConnectionList component.

Declaration

```
property RetrieveUDP: Boolean; // read-only!
```

Description

The RetrieveUDP property specifies whether the component should retreive the UDP connections to its internal list of available connections.

The list can be refreshed using Refresh method.

See also

RetreiveUDP and ConnectionList properties;
Refresh method.

9.14.3 Methods

9.14.3.1 Refresh

Applies to

acConnectionList, acProcessList and acIESniffer components (used to refresh the content of the internal data list).

Declaration

```
procedure Refresh;
```

Description

The Refresh method retrieves the list of all currently available connections fills the internal list of acProcessList component with new information about each system process. After refreshing of the list, you see current number of processes using Count property or get access to the the process information using Items property.

Use Refresh method to keep the list of processes always fresh. The component does NOT update its list automatically (it does it only once on startup).

See also

Count and Items properties.

9.14.3.2 FindConnectionByLocalPort

Applies to

acConnectionList component.

© 1998-2006, UtilMind Solutions®

Declaration

```
function FindConnectionByLocalPort(LocalPort: Word; Proto:
TacConnectionProto = pAny;
  ListeningPortOnly: Boolean = False): TacConnection;
```

Description

The FindConnectionByLocalPort method searches for the connection where the local port is the same as specified in *LocalPort* parameter and protocol is the same as specified in *Proto* parameter (or returns any connection, either TCP or UDP, if *Proto* is pAny). Also it can search for connection which where the *LocalPort* is listening port, if *ListeningPortOnly* parameter is True.

Function returns pointer to TacConnection object if the connection was found in the internal list of component, or **nil** if the connection with specified *LocalPort* not found.

See also

FindConnectionByRemotePort, FindConnectionByLocalIPStr, FindConnectionByRemoteIPStr, FindLocalIPStrByLocalPort, FindLocalIPStrByRemotePort, FindRemoteIPStrByLocalPort and FindRemoteIPStrByRemotePort methods.

9.14.3.3 FindConnectionByRemotePort

Applies to

acConnectionList component.

Declaration

```
function FindConnectionByRemotePort(RemotePort: Word; Proto:
TacConnectionProto = pAny): TacConnection;
```

Description

The FindConnectionByRemotePort function searches for the connection where the local port is the same as specified in *RemotePort* parameter and protocol is the same as specified in *Proto* parameter (or returns any connection, either TCP or UDP, if *Proto* is pAny).

Function returns pointer to TacConnection object if the connection was found in the internal list of component, or **nil** if the connection with specified *RemotePort* not found.

See also

FindConnectionByLocalPort, FindConnectionByLocalIPStr, FindConnectionByRemoteIPStr, FindLocalIPStrByLocalPort, FindLocalIPStrByRemotePort, FindRemoteIPStrByLocalPort and FindRemoteIPStrByRemotePort methods.

9.14.3.4 FindConnectionByLocalIPStr

Applies to

acConnectionList component.

Declaration

```
function FindConnectionByLocalIPStr(const LocalIPStr: String; Proto:
TacConnectionProto = pAny;
  ListeningPortOnly: Boolean = False): TacConnection;
```

Description

The FindConnectionByLocalIPStr method searches for the connection where the local IP address is the same as specified in *LocalIPStr* parameter and protocol is the same as specified in *Proto* parameter (or returns any connection, either TCP or UDP, if *Proto* is pAny). Also it can search for connection which where the *LocalPort* is listening port, if *ListeningPortOnly* parameter is True.

© 1998-2006, UtilMind Solutions®

Function returns pointer to TacConnection object if the connection was found in the internal list of component, or **nil** if the connection with specified *LocalIPStr* not found.

See also
FindConnectionByLocalPort, FindConnectionByRemotePort, FindConnectionByRemoteIPStr, FindLocalIPStrByLocalPort, FindLocalIPStrByRemotePort, FindRemoteIPStrByLocalPort and FindRemoteIPStrByRemotePort methods.

9.14.3.5 FindConnectionByRemoteIPStr

Applies to
acConnectionList component.

Declaration
```
function FindConnectionByRemoteIPStr(const RemoteIPStr: String; Proto:
TacConnectionProto = pAny): TacConnection;
```

Description
The FindConnectionByRemoteIPStr method searches for the connection where the remote IP address is the same as specified in *RemoteIPStr* parameter and protocol is the same as specified in *Proto* parameter (or returns any connection, either TCP or UDP, if *Proto* is pAny).

Function returns pointer to TacConnection object if the connection was found in the internal list of component, or **nil** if the connection with specified *RemoteIPStr* not found.

See also
FindConnectionByLocalPort, FindConnectionByRemotePort, FindConnectionByLocalIPStr, FindLocalIPStrByLocalPort, FindLocalIPStrByRemotePort, FindRemoteIPStrByLocalPort and FindRemoteIPStrByRemotePort methods.

9.14.3.6 FindLocalIPStrByLocalPort

Applies to
acConnectionList component.

Declaration
```
function FindLocalIPStrByLocalPort(LocalPort: Word; Proto:
TacConnectionProto = pAny): String;
```

Description
The FindLocalIPStrByLocalPort method searches for the connection where the local port is the same as specified in *LocalPort* parameter and protocol is the same as specified in *Proto* parameter (or returns any connection, either TCP or UDP, if *Proto* is pAny).

Function returns the local IP address of connection (if such connection found in internal list), or empty string if connection with specified *LocalPort* doesn't exists.

See also
FindConnectionByLocalPort, FindConnectionByRemotePort, FindConnectionByLocalIPStr, FindConnectionByRemoteIPStr, FindLocalIPStrByRemotePort, FindRemoteIPStrByLocalPort and FindRemoteIPStrByRemotePort methods.

9.14.3.7 FindLocalIPStrByRemotePort

Applies to
acConnectionList component.

© 1998-2006, UtilMind Solutions®

Declaration

```
function FindLocalIPStrByRemotePort(RemotePort: Word; Proto:
TacConnectionProto = pAny): String;
```

Description

The FindLocalIPStrByRemotePort method searches for the connection where the remote port is the same as specified in *RemotePort* parameter and protocol is the same as specified in *Proto* parameter (or returns any connection, either TCP or UDP, if *Proto* is pAny).

Function returns the local IP address of connection (if such connection found in internal list), or empty string if connection with specified *RemotePort* doesn't exists.

See also

FindConnectionByLocalPort, FindConnectionByRemotePort, FindConnectionByLocalIPStr, FindConnectionByRemoteIPStr, FindLocalIPStrByLocalPort, FindRemoteIPStrByLocalPort and FindRemoteIPStrByRemotePort methods.

9.14.3.8 FindRemoteIPStrByLocalPort

Applies to

acConnectionList component.

Declaration

```
function FindRemoteIPStrByLocalPort(LocalPort: Word; Proto:
TacConnectionProto = pAny): String;
```

Description

The FindRemoteIPStrByLocalPort method searches for the connection where the local port is the same as specified in *LocalPort* parameter and protocol is the same as specified in *Proto* parameter (or returns any connection, either TCP or UDP, if *Proto* is pAny).

Function returns the remote IP address of connection (if such connection found in internal list), or empty string if connection with specified *LocalPort* doesn't exists.

See also

FindConnectionByLocalPort, FindConnectionByRemotePort, FindConnectionByLocalIPStr, FindConnectionByRemoteIPStr, FindLocalIPStrByLocalPort, FindLocalIPStrByRemotePort and FindRemoteIPStrByRemotePort methods.

9.14.3.9 FindRemoteIPStrByRemotePort

Applies to

acConnectionList component.

Declaration

```
function FindRemoteIPStrByRemotePort(RemotePort: Word; Proto:
TacConnectionProto = pAny): String;
```

Description

The FindRemoteIPStrByRemotePort method searches for the connection where the remote port is the same as specified in *RemotePort* parameter and protocol is the same as specified in *Proto* parameter (or returns any connection, either TCP or UDP, if *Proto* is pAny).

Function returns the remote IP address of connection (if such connection found in internal list), or

© 1998-2006, UtilMind Solutions®

empty string if connection with specified *RemotePort* doesn't exists.

See also
FindConnectionByLocalPort, FindConnectionByRemotePort, FindConnectionByLocalIPStr, FindConnectionByRemoteIPStr, FindLocalIPStrByLocalPort, FindLocalIPStrByRemotePort and FindRemoteIPStrByLocalPort methods.

9.14.3.10 IsPortListening

Applies to
acConnectionList component.

Declaration
```
function IsPortListening(Port: Word; Proto: TacConnectionProto = pAny):
Boolean;
```

Description
The IsPortListening method checks whether the specified *Port* currently in use. Optionally you can specify *Proto* parameter, if you wish to check some certain TCP or UDP port.

See also
FindConnectionByLocalPort, FindConnectionByRemotePort, FindConnectionByLocalIPStr, FindConnectionByRemoteIPStr, FindLocalIPStrByLocalPort, FindLocalIPStrByRemotePort, FindRemoteIPStrByLocalPort and FindRemoteIPStrByRemotePort methods.

9.14.4 Events

9.14.4.1 OnHostnameRetrieved

Applies to
acHostnameResolver and acConnectionList components.

Declaration
```
type
  TacHostnameRetrievedEvent = procedure(Sender: TObject; const IPStr,
Hostname: String) of object;

property OnHostnameRetrieved: TacHostnameRetrievedEvent;
```

Description
The OnHostnameRetrieved event occurs when component has retrieved some IP address from the list (queue) specified in IPList property.

The event passes to its handler following parameters:

Parameter	Meaning
IPStr	the original IP address which has been specified in IPList property, but already remoted from that queue because the host name was retrieved.
Hostname	the name of machine retrieved by IP address after lookup.

If you are use cache (CacheSize > 0), the retrieved hostname will be put to the cache, so next time when you will resolve it, the process will be much quicker. Also you can use ResolveCachedIP function to get it very quickly.

© 1998-2006, UtilMind Solutions®

See also
AutoResolve and IPList properties;
Execute, Abort and ResolveCachedIP methods;
OnHostnameRetrieved event.

9.14.4.2 OnOnlineStatusChanged

Applies to
acHostnameResolver and acConnectionList components.

Declaration
```
property OnOnlineStatusChanged: TNotifyEvent;
```

Description
The OnOnlineStatusChanged event occurs when the component detects that machine become connected to Internet or connection lost.

To get current online status at any time — read IsOnline property. True means that machine currently connected to Internet and False means that it disconnected.

Remarks
1. The online status monitoring only performed if MonitorActive property is True.
2. Component checks online status with intervals specified in MonitorInterval property.

See also
IsOnline, MonitorActive and MonitorInterval properties.

9.14.5 TacConnection

Unit
acConnectionList

Declaration
```
type
  TacConnectionProto = (pTCP, pUDP, pAny);
  TacConnection = class(TacListObject)
  public
    Data: Pointer;

    property LocalIP: TInAddr;
    property LocalIPStr: String;
    property LocalIPLong: LongInt;
    property LocalPort: Word;

    property RemoteIP: TInAddr;
    property RemoteIPStr: String;
    property RemoteIPLong: LongInt;
    property RemotePort: Word;

    property LocalHostname: String;
    property RemoteHostname: String;

    property State: DWord;
    property StateStr: String;

    property Proto: TacConnectionProto;
    property ProtoName: String;
```

© 1998-2006, UtilMind Solutions®

```
    property IsIncoming: Boolean;
  end;
```

Description

The TacConnection object is the item of acConnectionList component, which holds the information about the TCP/UDP connection.

9.14.5.1 Data

plies to

acConnection and acProcess objects.

Declaration

```
property Data: Pointer;
```

Description

The Data property specifies any application-specific data associated with the TacConnection object.

Use Data to associate arbitrary data structure with the TacConnection object.

Example

```
// Demonstrates how to fill the ListView with process entries
// and associate ListItem with TacProcess object and otherwise,
TacProcess with ListItem...
procedure TForm1.Button1Click(Sender: TObject);
var
  I: Integer;
  ListItem: TListItem;
begin
  ListView1.Items.BeginUpdate;
  try
    ListView1.Items.Clear;
    I := acConnectionList1.Count;
    if I <> 0 then
     for I := 0 to I - 1 do
      begin
        ListItem := ListView1.Items.Add;
        ListItem.Caption := acConnectionList1[I].LocalHostname;

        // associate the Data pointer to TacConnection object and Data
pointer of Connection object to ListItem
       ListItem.Data := acConnectionList1[I];
       acConnectionList1[I].Data := ListItem;
      end;
  finally
    ListView1.Items.EndUpdate;
  end;
end;
```

9.14.5.2 LocalIP

Applies to

acConnection object.

Declaration

© 1998-2006, UtilMind Solutions®

```
property LocalIP: TInAddr;
```

Description
The LocalIP property determines the local IP address of the connection.

See also
LocalIP, LocalIPLong, LocalIPStr, LocalHostname, LocalPort, RemoteIP, RemoteIPLong, RemoteIPStr, RemoteHostname, RemotePort properties.

9.14.5.3 LocalIPStr

Applies to
acConnection object.

Declaration
```
property LocalIPStr: String;
```

Description
The LocalIP property returns the string representation of the local IP address of the connection.

See also
LocalIP, LocalIPLong, LocalIPStr, LocalHostname, LocalPort, RemoteIP, RemoteIPLong, RemoteIPStr, RemoteHostname, RemotePort properties.

9.14.5.4 LocalIPLong

Applies to
acConnection object.

Declaration
```
property LocalIPLong: LongInt;
```

Description
The LocalIP property returns the integer value which represents the local IP address of the connection.

See also
LocalIP, LocalIPLong, LocalIPStr, LocalHostname, LocalPort, RemoteIP, RemoteIPLong, RemoteIPStr, RemoteHostname, RemotePort properties.

9.14.5.5 LocalHostname

Applies to
acConnection object.

Declaration
```
property LocalIP: String;
```

Description
The LocalHostname property returns the hostname of the local machine (resolved by LocalIP).

See also
LocalIP, LocalIPLong, LocalIPStr, LocalHostname, LocalPort, RemoteIP, RemoteIPLong, RemoteIPStr, RemoteHostname, RemotePort properties.

© 1998-2006, UtilMind Solutions®

9.14.5.6 LocalPort

Applies to
acConnection object.

Declaration
```
property LocalPort: Word;
```

Description
The LocalPort property determines the local port of the connection.

See also
LocalIP, LocalIPLong, LocalIPStr, LocalHostname, LocalPort, RemoteIP, RemoteIPLong, RemoteIPStr, RemoteHostname, RemotePort properties.

9.14.5.7 RemoteIP

Applies to
acConnection object.

Declaration
```
property RemoteIP: TInAddr;
```

Description
The RemoteIP property determines the remote IP address of the connection.

See also
LocalIP, LocalIPLong, LocalIPStr, LocalHostname, LocalPort, RemoteIP, RemoteIPLong, RemoteIPStr, RemoteHostname, RemotePort properties.

9.14.5.8 RemoteIPStr

Applies to
acConnection object.

Declaration
```
property RemoteIPStr: String;
```

Description
The RemoteIPStr property returns the string representation of the remote IP address of the connection.

See also
LocalIP, LocalIPLong, LocalIPStr, LocalHostname, LocalPort, RemoteIP, RemoteIPLong, RemoteIPStr, RemoteHostname, RemotePort properties.

9.14.5.9 RemoteIPLong

Applies to
acConnection object.

Declaration
```
property RemoteIPLong: LongInt;
```

Description
The RemoteIPLong property returns the integer value which represents the remote IP address of the connection.

© 1998-2006, UtilMind Solutions®

See also
LocalIP, LocalIPLong, LocalIPStr, LocalHostname, LocalPort, RemoteIP, RemoteIPLong, RemoteIPStr, RemoteHostname, RemotePort properties.

9.14.5.10 RemoteHostname

Applies to
acConnection object.

Declaration
```
property RemoteHostname: String;
```

Description
The RemoteHostname property returns the hostname of the local machine (resolved by RemoteIP).

See also
LocalIP, LocalIPLong, LocalIPStr, LocalHostname, LocalPort, RemoteIP, RemoteIPLong, RemoteIPStr, RemoteHostname, RemotePort properties.

9.14.5.11 RemotePort

Applies to
acConnection object.

Declaration
```
property RemotePort: Word;
```

Description
The RemotePort property determines the remote port of the connection.

See also
LocalIP, LocalIPLong, LocalIPStr, LocalHostname, LocalPort, RemoteIP, RemoteIPLong, RemoteIPStr, RemoteHostname, RemotePort properties.

9.14.5.12 State

Applies to
acConnection object.

Declaration
```
property State: DWord;
```

Description
The State property determines the state of the connection.

The possible values for the State property are following:
```
const
  CLOSED       = 1;
  LISTEN       = 2;
  SYN_SENT     = 3;
  SYN_RECEIVED = 4;
  ESTABLISHED  = 5;
  CLOSE_WAIT   = 6;
  FIN_WAIT_1   = 7;
  CLOSING      = 8;
  LAST_ACK     = 9;
  FIN_WAIT_2   = 10;
```

© 1998-2006, UtilMind Solutions®

```
    TIME_WAIT     = 11;
    TCB_DISCARD   = 12;
```

See also
StateStr property.

9.14.5.13StateStr

Applies to
acConnection object.

Declaration
```
property StateStr: String;
```

Description
The StateStr property returns the string representation of State property (in human readable text format).

See also
State property.

9.14.5.14Proto

Applies to
acConnection object.

Declaration
```
type
  TacConnectionProto = (pTCP, pUDP, pAny);

property Proto: TacConnectionProto;
```

Description
The Proto property determines the low level protocol of the connection. The possible values are *pTCP* or *pUDP*.

See also
ProtoName property.

9.14.5.15ProtoName

Applies to
acConnection object.

Declaration
```
property ProtoName: String;
```

Description
The ProtoName property returns the string representation of low level protocol of the connection.

The possible values of this property are `'TCP'` or `'UDP'`.

See also
Proto property.

© 1998-2006, UtilMind Solutions®

9.14.5.16 IsIncoming

Applies to
acConnection object.

Declaration
```
property IsIncoming: Boolean;
```

Description
The IsIncoming property determines whether the connection is incoming or outgoing.

Original code
```
function TacConnection.GetIsIncoming: Boolean;
begin
  Result := (Owner <> nil) and (Owner is TacConnections) and
            (TacConnections(Owner).IsPortListening(LocalPort, Proto));
end;
```

See also
LocalPort and Proto properties;
IsPortListening method.

9.15 acCronJob

9.15.1 acCronJob

Overview
The acCronJob component is the thread-based alarm timer implementation for Delphi and C++ Builder which acts like *cronjob* utility in Unix. The component produces periodical OnAlert events by schedule specified in CRON format (actually it uses extended CRON format with new "seconds" field, to let the event to be triggered even every second).

CRON format
To specify the moments when the component should trigger the OnAlert event, the component uses the **CRON** format (all who worked in Unix must be aware about this utility, this is daemon of scheduled, periodically executed tasks). The authors of CronJob component has very liked an idea of describing the periodical events of any complexity, tiered to astronomical time, in simple text string. We have used slightly modified CRON format, added seconds field and altered the rules of describing the lists of numbers.

The CronMask property presents itself of several "list of numbers", separated by space. Every list specifies enumeration of moments of time or date, in units, dependant of position of the list in the mask string.

The CronMask schedule should be specified in following maneer (list of numbers/enumerations separated by spaces):
```
Seconds Minutes Hours Days Months DaysOfWeek
```

Here is some examples of describing of schedule in the CRON format:

- Each minute at 0 seconds and 30 seconds:
```
0,30
0+30
+30
```
- Each second at 0, 8 and 16 hours:
```
0-59 0-59 0-16+8
* * 0+8
* * +8
```

© 1998-2006, UtilMind Solutions®

- The beginning of each hour, except noon and midnight: `0 0 1-11,13-23`
- Every 3 seconds of 1st day of each month: `0-59+3 * * 1` `+3 * * 1`
- 30 minutes, 0 seconds of each hour at Sunday: `0 30 * * * 0`
- Every 5 seconds: `+5`
- Every 2 hours since 8AM to 6PM, except weekends: `0 8-18+2 * * 1-6`
- At 10:15, 11:15, 13:15, 14:15, 17:15 except weekends: `0 15 10,11,13,14,17 * * 1-6`

Description of CRON lists

The string list with numbers without spaces is the group of numbers separated by comma "," (i.e: `0,30`). The group can contain one number or the range of numbers. The ranges should be specified by two numbers, separated by hyphen "-" (the beginning and ending of the range, i.e: `0-9`). Optionally, after the range you can also specify the value of *step,* separated by plus "+" (i.e: `0-59+5`). The default step value (if none specified) is 1. If the step specified, the ending value may be not specified, by default it will be the maximal value in context of purpose of the list. The beginning of the range is 0 by default (or 1 for days and monthes).

The asterisk sign "*" instead of group means all diapasone of possible values in the context.

The order of values or ranges in one list is not important. For example, the list like `0-5,12,8,20-30+2` is interpreted as following enumeration: `0,1,2,3,4,5,8,12,20,22,24,26,28,30`.

How to use

Drop the CronJob component on your form, specify the CronMask property and write the OnAlert event handler which will process scheduled events.

See also

acAccurateTimer and acAwayTimer components.

9.15.2 Properties

9.15.2.1 Active

Applies to

acCronJob, acAwayTimer and acAccurateTimer components.

Declaration

```
property Active: Boolean; // True by default
```

Description

The Active property controls whether the acCronJob is active and able to trigger OnAlert event at the time scheduled in CronMask property.

Set Active property to True to enable the cronjob and, othwise set it to False to disable it.

See also

CronMask property and OnAlert event.

9.15.2.2 AlertsCount

Applies to

acCronJob and acAccurateTimer components.

Declaration

```
property AlertsCount: Cardinal; // read-only!
```

© 1998-2006, UtilMind Solutions®

Description
The AlertsCount is the read-only property which determines the number triggered OnAlert events in current application session.

See also
AlertsCount property;
OnAlert event.

9.15.2.3 CronMask

Applies to
acCronJob component.

Declaration
```
property CronMask: String;
```

Description
The CronMask string specifies the string in CRON format, which describes scheduled time periods to trigger OnAlert events.

CRON format
To specify the moments when the component should trigger the OnAlert event, the component uses the **CRON** format (all who worked in Unix must be aware about this utility, this is daemon of scheduled, periodically executed tasks). The authors of CronJob component has very liked an idea of describing the periodical events of any complexity, tiered to astronomical time, in simple text string. We have used slightly modified CRON format, added seconds field and altered the rules of describing the lists of numbers.

The CronMask property presents itself of several "list of numbers", separated by space. Every list specifies enumeration of moments of time or date, in units, dependant of position of the list in the mask string.

The CronMask schedule should be specified in following maneer (list of numbers/enumerations separated by spaces):
```
Seconds Minutes Hours Days Months DaysOfWeek
```

Here is some examples of describing of schedule in the CRON format:

- Each minute at 0 seconds and 30 seconds:
```
0,30
0+30
+30
```
- Each second at 0, 8 and 16 hours:
```
0-59 0-59 0-16+8
* * 0+8
* * +8
```
- The beginning of each hour, except noon and midnight:
```
0 0 1-11,13-23
```
- Every 3 seconds of 1st day of each month:
```
0-59+3 * * 1
+3 * * 1
```
- 30 minutes, 0 seconds of each hour at Sunday:
```
0 30 * * * 0
```

Description of CRON lists
The string list with numbers without spaces is the group of numbers separated by comma "," (i.e: `0,30`). The group can contain one number or the range of numbers. The ranges should be specified by two numbers, separated by hyphen "-" (the beginning and ending of the range, i.e: `0-9`). Optionally, after the range you can also specify the value of *step,* separated by plus "+" (i.e: `0-59+5`). The default step value (if none specified) is 1. If the step specified, the ending value may be not specified, by default it will be the maximal value in context of purpose of the list. The beginning

© 1998-2006, UtilMind Solutions®

of the range is 0 by default (or 1 for days and monthes).

The asterisk sign "*" instead of group means all diapasone of possible values in the context.

The order of values or ranges in one list is not important. For example, the list like `0-5,12,8,20-30+2` is interpreted as following enumeration: `0,1,2,3,4,5,8,12,20,22,24,26,28,30`.

See also
Active property and OnAlert event.

9.15.2.4 LastTime

Applies to
acCronJob component.

Declaration
```
property LastTime: TDateTime; // read-only!
```

Description
The LastTime is read-only property which determines the date/time of last triggered OnAlert event.

See also
AlertsCount property.

9.15.3 Events

9.15.3.1 OnAlert

Applies to
acCronJob component.

Declaration
```
property OnAlert: TNotifyEvent;
```

Description
The OnAlert event occurs at the time described in CronMask property. Write this event handler to take some specific actions by specified schedule.

See also
Active and CronMask properties.

9.16 acCursorLocker

9.16.1 TacCursorLocker

Overview
The acCursorLocker component is able to lock the movements of a mouse cursor, within the bounds of the Delphi/BCB form or within any control on this form.

To lock the cursor withing the form - just drop on this form acCursorLocker component, set Active property to True then execute the project.

The similar locking mechanism used even in Delphi/BCB IDE. For example, try to select components of your form and move mouse pointer beyond the form bounds. As you can see, cursor will be locked while you pushing down a left mouse button.

© 1998-2006, UtilMind Solutions®

9.16.2 Properties

9.16.2.1 Active

Applies to
acCursorLocker, acAwayTimer, acFormBackground, acFormHelp, acFormMagnet, acFormSystemMenu and acIESniffer components

Declaration
```
property Active: Boolean;
```

Description
The Active property controls whether the acCursorLocker should lock the mouse pointer within the bounds of control specified by Control property.

9.16.2.2 Control

Applies to
acCursorLocker and acStatusControl components.

Declaration
```
property Control: TControl;
```

Description
The Control property specifies the control which will hook the mouse movements and lock within its bounds. Control property can be the name of form or name of any control on this form.

Controls are visual components: that is, they are components you can see when your application is running. All controls have properties in common that specify the visual attributes of controls, such as Left, Top, Height, Width.

9.17 acEmbeddedForm

9.17.1 TacEmbeddedForm

Overview
The acEmbeddedForm component used to insert the form and all its content to any successor of TWinControl (i.e: Panels, GroupBoxes, Labels, pages of PageControl etc).

This is better than usual frames that comes with Delphi/BCB beginning from v5 since:
1. You can *dynamically* change the container of the embedded form at run-time (dynamically insert or remove the form from any successor of TWinControl).
2. Still have all functionality of usual form (some controls can be placed only to form or its successor).

How to use?
Just select the control which should contain the form in the HolderControl property and specify the name of form to insert in the FormName property. You can change these propertis at run-time to "embed" or "un-embed" the forms from specified controls.

Notes
1. You can specify the source form and target control at design-time but the "embedding" take effect at run-time only!

2. Component do not raise any error or exception if the form specified in FormName could not be found. So please make sure that the form with such name are really exists.

© 1998-2006, UtilMind Solutions®

3. If you *dinamically creating the form* which you wish to embed to some container at run-time, don't forget to this form an unique name, to let the component to identify it withing the application. Example:

```
MyNewForm := TMyForm.Create(Application);
MyNewForm.Name := 'myform' + UniqueId; // <-- give it unique name
acEmbeddedForm1.FormName := MyNewForm.Name;
```

4. If you would like to embed the form to *ActiveX form*, please make sure that form which you would like to embed automatically created before the ActiveX form displayed. Unfortunately ActiveX projects does not create all VCL forms automatically, so you should add some of following code to your ActiveX project:

```
library YourActiveXFormLibrary;

uses
  Forms, // <-- this will allows to use an "Application" object
  ComServ,

// -- snip --

{$E ocx}

exports
  DllGetClassObject,
  DllCanUnloadNow,
  DllRegisterServer,
  DllUnregisterServer;

{$R *.TLB}

{$R *.RES}

begin
  Application.CreateForm(TForm1, Form1); // <-- this will auto-create
additional VCL forms
end.
```

9.17.2 Properties

9.17.2.1 FormName

Applies to

acEmbeddedForm component.

Declaration

```
property FormName: String;
```

Description

The FormName specifies the name of form which should be inserted to the control specified in the HolderControl property.

Note

Component do not raise any error or exception if the form specified in FormName could not be found. So please make sure that the form with such name are really exists.

See also

HolderControl property.

© 1998-2006, UtilMind Solutions®

9.17.2.2 HolderControl

Applies to
acEmbeddedForm component.

Declaration
```
property HolderControl: TWinControl;
```

Description
The HolderControl property specifies the control which should be used to hold the embedded form. You can select any successor of TWinControl.

See also
FormName property.

9.18 acExtAssociation

9.18.1 TacExtAssociation

Overview
The **acExtAssociation** component lets you to read, install or uninstall the file types (extension associations) to the system registry. Component provides with detalied information about any file types installed to the system (ie. pas, .wav, .doc, .gif, .html). It determinates the executable file, argument string, media-type, icon file, icon index and two icons (32x32 and 16x16) associated with specified file type. To get the information about any file extension — just specify the extension at Extension property.

Also, with **acExtAssociation** you may install new extensions or modify existing, changing the location of executable file or extension icon. For example, you can install ".yourdatafile" extension, specify the executable program for this file type and point the file icon associated with this file extension in the shell. After opening the any file with ".yourdatafile" extension (clicking the file icon), the application associated with this file type will be executed to process some data from ".yourdatafile".

How to read the information about the file type ?
Just specify any file extension (ie. pas, .wav, .doc, .gif, .html) to the EXTENSION property. All others properties will be filled with information about extension.
See also GetExecutableByExtension function of acUtils unit.

How to install the new file type (file association) ?
1. Switch the component to Read-Write mode (amReadWrite value of AccessMode property)
2. Assign the file extension to the Extension property.
3. Specify the type of file in the FileDescription property.
4. Point the executable file associated with this file type in the ExecutableFile property.
5. Point the file where stored the extension icon in the IconLocation property. You can point to any icon file — program (.exe), dynamic link library (.dll) or icon (.ico). Value can be same as assigned to the ExecutableFile property.
6. Specify the icon index optionally in the IconIndex property.

Since you don't need to call any method, just done some manipulations with a few properties, you can install the new file extension even at design-time. Please see the code sample for more details.

Alternatively you can use InstallExtension method.

How to uninstall the file extension ?

© 1998-2006, UtilMind Solutions®

Just call the UninstallExtension method and specify the extension in *Ext* parameter of UninstallExtension function.

9.18.2 Properties

9.18.2.1 AccessMode

Applies to
acExtAssociation component.

Declaration

```
type
  TacShellExtAccessMode = (amReadOnly, amReadWrite);

property AccessMode: TacShellExtAccessMode;
```

Description
The AccessMode property controls whether other properties of this component can be modified (whether the information about file type specified in the EXTENSION property can be modified).

By default, AccessMode is `amReadOnly`. This means that you can only read the information but can not change the information for file type specified in EXTENSION property.

Set AccessMode to `amReadWrite` to install new, or modify existing file type.

Caution
⚠ Please be careful on modifying the information of existing file types!! Your changes will override previous settings in the Windows shell!

See also
EXTENSION property.

9.18.2.2 DefCommandName

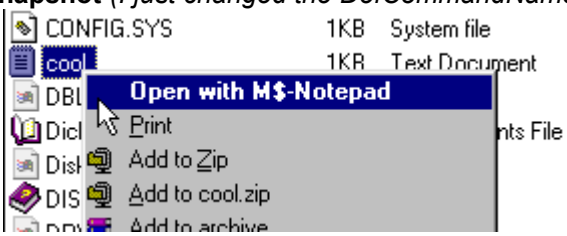
Applies to
acExtAssociation component.

Declaration

```
property DefCommandName: String;
```

Description
The DefCommandName property determines and specifies the text for default command which are used to open the file. The text for default command appears as first (default) item in the context menu for any shell objects.

Snapshot *(I just changed the DefCommandName for .TXT files)*



See also
EXTENSION property.

© 1998-2006, UtilMind Solutions®

9.18.2.3 ExecutableFile

Applies to
acExtAssociation component.

Declaration
```
property ExecutableFile: String;
```

Description
The ExecutableFile property determines the location of executable file, associated with the file extension specified in the EXTENSION property.

For example, when user starts any non-executable file with pre-installed file extension, the executable file, associated with that file extension will be started to process the file. For example, when user clicks the BMP icon, the shell will automatically start the "Paint" program, to allow user to modify the picture.

As a rule, the executable file associated with the file extension must be started with some command line parameters. These parameters informs the executable about the file that need to be processed. See ParamString property for more details.

If you would like to change this property and point another executable, or point an EXE for new file extension, make sure that AccessMode property is `amReadWrite` (write permission).

See also
ParamString property and example of usage.

9.18.2.4 ExtDescription

Applies to
acExtAssociation component.

Declaration
```
property ExtDescription: String;
```

Description
The ExtDescription property determines the unique extension identifier, used internally by Windows shell.

See also
FileDescription property.

9.18.2.5 EXTENSION

Applies to
acExtAssociation component.

Declaration
```
property EXTENSION: String;
```

Description
The EXTENSION property specifies the file extension to retrieve the information about file type.

You can play with EXTENSION property at design-time, specifying various file types (i.e: BMP, DOC, WAV, PAS, CPP, DLL, BPL, DPK and so forth).

© 1998-2006, UtilMind Solutions®

Most file extensions

See also
MIMEType property.

9.18.2.6 FileDescription

Applies to
acExtAssociation component.

Declaration
```
property FileDescription: String;
```

Description
The FileDescription property determines or specifies the description of file type, specified in the EXTENSION property.

You can see description for the file types in the Windows Explorer. Open the Explorer, and select "Detail" view…

To modify the FileDescription value, make sure that AccessMode is `amReadWrite`.

See also
ExtDescription property.

9.18.2.7 IconIndex

Applies to
acExtAssociation and acShortcut components.

Declaration
```
property IconIndex: Integer;
```

Description
The IconIndex property determines or specifies the index of icon within the icon file, specified by IconLocation property.

See also
IconLocation property.

9.18.2.8 IconLocation

Applies to
acExtAssociation and acShortcut components.

Declaration
```
property IconLocation: String;
```

Description
The IconLocation property determines or specifies the location of the file which contains the icon associated with the file type specified in the EXTENSION property. The icon file can be an EXE, DLL or ICO. To determinate or specify the icon index within the icon file, use IconIndex property.

To modify the ParamString value, make sure that AccessMode is `amReadWrite`.

See also

© 1998-2006, UtilMind Solutions®

IconIndex property.

9.18.2.9 LargeIcon

Applies to
acExtAssociation and acShortcut components.

Declaration
```
property LargeIcon: TIcon; // read only !!
```

Description
The LargeIcon property retrieves the 32x32 icon which represents the file type, specified in the EXTENSION, within the Windows shell.

The icon will be retrieved from the file specified in IconLocation property, by index specified in IconIndex property.

See also
SmallIcon property.

9.18.2.10 MIMEType

Applies to
acExtAssociation component.

Declaration
```
property MIMEType: String;
```

Description
The MIMEType property determines or specifies the media type of the file extension. The media type (also known as Multipurpose Internet Mail Extension (MIME)) parameter used by Internet applications to determinate which *content type* it should use on sending the e-mail messages and POSTing the files via HTTP protocol.

For more information about Internet media types, please read RFC 2045, 2046, 2047, 2048, and 2077 (http://www.oac.uci.edu/indiv/ehood/MIME/MIME.html). Check out also the Internet media type registry at ftp://ftp.iana.org/in-notes/iana/assignments/media-types.

See also
EXTENSION property;
acHTTP and acSendMail components.

9.18.2.11 ParamString

Applies to
acExtAssociation component.

Declaration
```
property ParamString: String;
```

Description
The ParamString property determines and specifies the command-line parameters which should be transmitted to an executable file, when user starts any file, associated with this executable.

For example, you have installed the new file extension and would like to transmit to executable the name of data-file which starts your program. You must specify the "%1" parameter to the ParamString property, then, in the executable, catch the first command line parameter using

© 1998-2006, UtilMind Solutions®

ParamStr(1) routine. See FormCreate method in the sample code.

To modify the ParamString value, make sure that AccessMode is `amReadWrite`.

See also
ExecutableFile property and example of usage.

9.18.2.12 SmallIcon

Applies to
acExtAssociation and acShortcut components.

Declaration
```
property SmallIcon: TIcon; // read only !!
```

Description
The SmallIcon property retrieves the 16x16 icon which represents the file type, specified in the EXTENSION, within the Windows shell.

The icon will be retrieved from the file specified in IconLocation property, by index specified in IconIndex property.

See also
LargeIcon property.

9.18.3 Methods

9.18.3.1 InstallExtension

Applies to
acExtAssociation component.

Declaration
```
function TacExtAssociation.InstallExtension(Extension, ExecutableFile,
    ParamString, MIMEType, ExtDescription, FileDescription, IconFile:
    String; IconIndex: Integer): Boolean;
```

Description
The InstallExtension method is the alternative way to install the new file type to the Windows shell.

Parameters	Meaning
Extension	specifies the file extension. For example, '.test';
ExecutableFile	specifies the executable file associated with this file type (see ExecutableFile property);
ParamString	specifies the command-line parameters to transmit to the executable file on startup (see ParamString property);
MIMEType	specifies the media-type of the extension (see MIMEType property). This parameter can be empty;
ExtDescription	unique identifier for file extension, used internally within Windows shell (see ExtDescription property);
FileDescription	description of file type (will be visible in Windows explorer and "Properties" dialog) (see FileDescription property);
IconFile	location of the icon file (can be an EXE, DLL or ICO) (see IconLocation

© 1998-2006, UtilMind Solutions®

property);

IconIndex index of the icon in the *IconFile* (see IconIndex property).

To uninstall the extension from the shell — use UninstallExtension method.

Example

```
procedure TForm1.InstallButtonClick(Sender: TObject);
begin
   acExtAssociation1.InstallExtension('.avi',
   'C:\Program Files\Windows Media Player\MPLAYER2.EXE', '%1', '',
   'avifile', 'AVI Clip', '', 0);
end;
```

See also

UninstallExtension method;
ExecutableFile, ParamString, MIMEType, ExtDescription, FileDescription, IconLocation and IconIndex properties.

9.18.3.2 UninstallExtension

Applies to

acExtAssociation component.

Declaration

```
procedure UninstallExtension(Ext: String);
```

Description

The UninstallExtension method uninstalls the file type, specified in the Ext parameter, from the system registry and clears all registry keys which describes this file extension.

Example

```
procedure TForm1.UninstallButtonClick(Sender: TObject);
begin
   acExtAssociation1.UninstallExtension('myext');
end;
```

See also

Example of usage;
InstallExtension method.

9.18.4 ExtAssociation example

Following example demonstrates how to install and uninstall the file extension in the shell.
Let's say we'd like to install and uninstall the ".TEST" extension and point our executable as primary executable for this file type…

```
procedure TForm1.InstallTestExtensionBtnClick(Sender: TObject);
begin
   acExtAssociation.AccessMode := amReadWrite;
   acExtAssociation.EXTENSION := '.test';
   acExtAssociation.FileDescription := 'My Test File';
   acExtAssociation.ExecutableFile := Application.ExeName;
   acExtAssociation.ParamString := '%1';
   acExtAssociation.IconLocation := Application.ExeName;
   acExtAssociation.IconIndex := 1;
```

© 1998-2006, UtilMind Solutions®

```
{ ALTERNATIVELY, you can use InstallExtension method }
  acExtAssociation.InstallExtension('.test', Application.ExeName, '%1',
    '', 'testfile', 'My Test File', Application.ExeName, 1);
end;

procedure TForm1.UninstallTestExtensionBtnClick(Sender: TObject);
begin
  acExtAssociation.UninstallExtension('.test');
end;

{ how to catch the command line parameter and get the name of data file
(in ParamStr(1)) }
procedure TForm1.FormCreate(Sender: TObject);
begin
  if ParamCount <> 0 then
    ShowMessage('Started by "' + ParamStr(1) + '" file !');
end;
```

9.19 acFileStorage

9.19.1 TacFileStorage

Overview

The FileStorage component capable to upload and hold any data files within your Delphi/BCB forms (within the body of the EXE-program). If your software requires any additional files (.DLL's, WAV's, .TXT's etc), these files could be uploaded straight onto your form and be extracted from executable file at run-time (AutoExtract property). Also you can access to stored files directly from memory without extracting them to disk (see example).

You can use this component if you would like to make high integration of your app with stored files and get access to them at run-time or, if you would like to supply your customers with just one, single executable file!.

To upload files onto the form, or manage already stored files at design-time - invoke the **FileStorage Designer** (double click on component or click on Files property).

See also

acWavPlayer component.

9.19.2 Properties

9.19.2.1 AutoExtract

Applies to

acFileStorage component.

Declaration

```
type
  TAutoExtract = class
  published
    property Enabled: Boolean;
    property ExtractTo: TExtractTo;
    property Overwrite: TacFileStorageOverwrite;
    property RemoveOnTerminate: Boolean;
  end;

property AutoExtract: TAutoExtract;
```

© 1998-2006, UtilMind Solutions®

Description
The AutoExtract is the TPersistent property which contains the list of sub-properties.

You may use these auto extracting features if your would like to extract your files on every start of your application. Make Enabled property True and specify the path where files should be stored. If you don't want to keep extracted files permanenetly on the hard drive and would like to kill all extracted files after closing of your program - set RemoveOnTerminate to True.

If you would NOT extract files automatically on program startup, set Enabled property to False and ignore other auto extracting features. Anyway you may use Extract method at any time.

Note
Since auto extracting features used by acFileStorage on program startup only, you don't need to change anything from AutoExtract subproperties at run-time. You may set these properies at design-time only, all run-time modifications will senseless.

9.19.2.1.1 Enabled

Applies to
acFileStorage component, as subproperty of AutoExtract structure.

Declaration
```
property Enabled: Boolean;
```

Description
Controls whether acFileStorage component should automatically extract all files on program startup.

If AutoExtract.Enabled is TRUE, FileStorage will automatically extract all files on program startup, to location specified in ExtractTo property.

If AutoExtract.Enabled is FALSE then FileStorage will NOT extract files on startup and all other subproperties of the AutoExtract property will ignored.

Note
You don't need to change anything from AutoExtract subproperties at run-time because of these auto extracting features used by acFileStorage on program startup only. You may set this property at design-time only, all run-time modifications will senseless.

9.19.2.1.2 ExtractTo

Applies to
acFileStorage component, as subproperty of AutoExtract structure.

Declaration
```
type
  TExtractTo = class(TPersistent)
  published
    property CreatePath: Boolean;
    property ToplevelDir: TacToplevelDir;
    property Subdirectory: String;
  end;

property ExtractTo: TExtractTo;
```

Examples of usage
1. If you would like to extract stored files to the Desktop - just make TopLevelDir property

© 1998-2006, UtilMind Solutions®

"tdDesktop". Leave SubDirectory string clean.

2. If you would like to extract files to current directory, create subdirectory named, let's say, "plugins" and exctract files there - set ToplevelDir property as "tdCurrentDir" and assign SubDirectory property as "plugins" (`SubDirectory := 'plugins'`). If you would like to create this subdirectory automatically even if it's does not exists - set CreatePath property to True.

3. If you would like to extract your files to non-system folder (for example, to "h:\databases\my own database\") then you don't need to use any toplevel directories, just set ToplevelDir property to "tdNone" and assign to SubDirectory "h:\databases\my own database" path. If you would like to create this subdirectory automatically even if it's does not exists - set CreatePath property to True.

Note
You don't need to change anything from AutoExtract subproperties at run-time because of these auto extracting features used by acFileStorage on program startup only. You may set this property at design-time only, all run-time modifications will senseless.

9.19.2.1.2.1 CreatePath

Applies to
acFileStorage component as subproperty of AutoExtract.ExtractTo structure.

Declaration
```
property CreatePath: Boolean;
```

Description
Controls whether FileStorage should automatically create path for exctractable files.

Examples of usage
1. If you would like to extract stored files to the Desktop - just make TopLevelDir property "tdDesktop". Leave SubDirectory string clean.

2. If you would like to extract files to current directory, create subdirectory named, let's say, "plugins" and exctract files there - set ToplevelDir property as "tdCurrentDir" and assign SubDirectory property as "plugins" (`SubDirectory := 'plugins'`). If you would like to create this subdirectory automatically even if it's does not exists - set CreatePath property to True.

3. If you would like to extract your files to non-system folder (for example, to "h:\databases\my own database\") then you don't need to use any toplevel directories, just set ToplevelDir property to "tdNone" and assign to SubDirectory "h:\databases\my own database" path. If you would like to create this subdirectory automatically even if it's does not exists - set CreatePath property to True.

Note
You don't need to change anything from AutoExtract subproperties at run-time because of these auto extracting features used by acFileStorage on program startup only. You may set this property at design-time only, all run-time modifications will senseless.

9.19.2.1.2.2 SubDirectory

Applies to
acFileStorage component as subproperty of AutoExtract.ExtractTo.

Declaration
```
property SubDirectory: String;
```

© 1998-2006, UtilMind Solutions®

Description

The value of the Directory property points the directory or subdirectory where will be stored your files on program startup.

Examples of usage

1. If you would like to extract stored files to the Desktop - just make TopLevelDir property "`tdDesktop`". Leave SubDirectory string clean.

2. If you would like to extract files to current directory, create subdirectory named, let's say, "plugins" and exctract files there - set ToplevelDir property as "`tdCurrentDir`" and assign SubDirectory property as "plugins" (`SubDirectory := 'plugins'`). If you would like to create this subdirectory automatically even if it's does not exists - set CreatePath property to True.

3. If you would like to extract your files to non-system folder (for example, to "h:\databases\my own database\") then you don't need to use any toplevel directories, just set ToplevelDir property to "`tdNone`" and assign to SubDirectory "h:\databases\my own database" path. If you would like to create this subdirectory automatically even if it's does not exists - set CreatePath property to True.

Note

You don't need to change anything from AutoExtract subproperties at run-time because of these auto extracting features used by acFileStorage on program startup only. You may set this property at design-time only, all run-time modifications will senseless.

See also

Extract method.

9.19.2.1.2.3 ToplevelDir

Applies to

acFileStorage component as subproperty of AutoExtract.ExtractTo structure.

Declaration

```
type
  TacToplevelDir = (tdCurrentDir, tdWindowsDir,
                    tdTempDir, tdSystemDir,
                    tdMediaDir, tdCursorsDir,
                    tdHelpDir, tdSamplesDir,
                    tdDesktop, tdProgramFiles,
                    tdMyDocuments, tdMyPictures,
                    tdAppData, tdNone);

property ToplevelDir: TToplevelDir;
```

Description

The ToplevelDir property specifies the directory where you would like to automatically extract files on program startup.

Examples of usage

1. If you would like to extract stored files to the Desktop - just make TopLevelDir property "`tdDesktop`". Leave SubDirectory string clean.

2. If you would like to extract files to current directory, create subdirectory named, let's say, "plugins" and extract files there - set ToplevelDir property as "`tdCurrentDir`" and assign SubDirectory property as "plugins" (`SubDirectory := 'plugins'`). If you would like to create this subdirectory automatically even if it's does not exists - set CreatePath property to True.

© 1998-2006, UtilMind Solutions®

3. If you would like to extract your files to non-system folder (for example, to "h:\databases\my own database\") then you don't need to use any top level directories, just set ToplevelDir property to "tdNone" and assign to SubDirectory "h:\databases\my own database" path. If you would like to create this subdirectory automatically even if it's does not exists - set CreatePath property to True.

Notes

1. You don't need to change anything from AutoExtract subproperties at run-time because of these auto extracting features used by acFileStorage on program startup only. You may set this property at design-time only, all run-time modifications will senseless.

2. Current directory (tdCurrentDir) is the same as ExtractFilePath(Application.ExeName).

3. If TopLevelDir = tdNone, the file will be extracted to absolute path specified in SubDirectory property.

9.19.2.1.3 Overwrite

Applies to

acFileStorage component, as subproperty of AutoExtract structure.

Declaration

```
type
  TacFileStorageOverwrite = (owIfSizeDifferent,
                             owNever, owAlways);

property Overwrite: TacFileStorageOverwrite;
```

Description

Controls whether FileStorage could overwrite already extracted files on program startup.

Note

You don't need to change anything from AutoExtract subproperties at run-time because of these auto extracting features used by acFileStorage on program startup only. You may set this property at design-time only, all run-time modifications will senseless.

9.19.2.1.4 RemoveOnTerminate

Applies to

acFileStorage component, as subproperty of AutoExtract structure.

Declaration

```
property RemoveOnTerminate: Boolean;
```

Description

If RemoveOnTerminate property is True then all automatically extracted files will be deleted on program termination. If the target Subdirectory is specified, the acFileStorage will also try to delete the subdirectory, if it is empty.

Warning!

If you have more than one instance of your application running and RemoveOnTerminate flag is True, you have a risk that the second closed instance will delete the first instance's files when it closes. Such risk exists even if you are using acOnlyOne to prevent execution of second instances. However there is a way to avoid this. You just need to hook OnAlreadyExists event of acOnlyOne and turn off RemoveOnTerminate flag in acFileStorage:

```
procedure TForm1.acOnlyOne1AlreadyExist(Sender: TObject);
begin
```

© 1998-2006, UtilMind Solutions®

```
    acFileStorage1.AutoExtract.RemoveOnTerminate := False;
    // messagebox
  end;
```

Remarks

It will remove only files extracted per current application session, and will not delete files that was exist before extracting. If you wish to be sure that ALL files are stored on start and definitely deleted on termination — set Overwrite property to `owAlways`.

See also

Enabled and Overwrite properties; SubDirectory property of ExtractTo structure.

9.19.2.2 Count

Applies to

acFileStorage component.

Declaration

```
property Count: Integer; // Read only !
```

Description

The Count is read-only property which holds the number of currently stored files.

See also

Files property.

9.19.2.3 DataSize

Applies to

acFileStorage component.

Declaration

```
property DataSize: Integer; // Read only !
```

Description

The DataSize is read-only property which holds the total size of all currently stored files.

See also

Files property.

9.19.2.4 Files

Applies to

acFileStorage component.

Declaration

```
property Files: TStoredFiles; { Successor of TList. Contains the list of
                                TStoredFile objects }
```

Description

The Files property is the List, which contains all currently stored files. Every item of this list is the file uploaded onto your form at design-time. All files in this list are TStoredFile objects.

If you would like to access stored file at run-time directly from memory you may use following **example**:

© 1998-2006, UtilMind Solutions®

```
var
  StoredFile: TStoredFile;
  DataStream: TMemoryStream;
begin
  StoredFile := FileStorage1.Files[1];
  DataStream := StoredFile.Data; // Data: TMemoryStream
```

Another example (demonstrates how to retrieve the content of text file to Memo control):

```
begin
  TStoredFile(acFileStorage1.Files[0]).Data.Position := 1; // Since Data
is the stream we must reset its position
  Memo1.Lines.LoadFromStream(TStoredFile(acFileStorage1.Files[0]).Data);
end;
```

And yet another example (demonstrates how to retrieve the stored file as string):

```
procedure TForm1.Button1Click(Sender: TObject);
var
  StoredFile: TStoredFile;
  St: String;
begin
  StoredFile := acFileStorage1.Files[0]; // first file from FileStorage

  SetLength(St, StoredFile.Data.Size); // set the length of string
(StoredFile.Data is the TMemoryStream)
  Move(StoredFile.Data.Memory^, St[1], StoredFile.Data.Size);

  // and finally we can put the retreived string to Memo control
  Memo1.Text := St;
end;
```

See also

Count and DataSize properties;
Extract method.

9.19.3 Methods

9.19.3.1 Extract

Applies to

acFileStorage component.

Declaration

```
function Extract(const FileName: String; OutputFile: String = ''):
Boolean; overload;
function Extract(FileIndex: Integer; OutputFile: String = ''): Boolean;
overload;
```

Description

The Extract method is destined to extract stored files from acFileStorage component to disk.

Parameters	Meanings
FileName	specifies the exact (though case sensitive) file name of stored file.
FileIndex	this is alternative to *FileName* parameter in overloaded version of Extract method. Specifies the index of file in Files list.
OutputFile	specifies the location (directory) and the file name of output file. If not specified,

© 1998-2006, UtilMind Solutions®

the FileStorage will extract file to path specified in AutoExtract.ExtractTo.SubDirectory property.

Returns True if file has been successfully extracted.

Example 1

```
procedure TForm1.ExtractBtnClick(Sender: TObject);
begin
  acFileStorage1.Extract('MyFile.exe', 'c:\MyFolder\MyFile.exe');
end;
```

Example 2

```
procedure TOptionsForm.DefaultsBtnClick(Sender: TObject);
begin
  MainForm.acFileStorage1.Extract(0); // extracts first file in Files
list to current folder
end;
```

See also

AutoExtract structure and Files property.

9.19.4 Events

9.19.4.1 OnExtract

Applies to

acFileStorage component.

Declaration

```
procedure OnExtract(Sender: TObject; FileName: String; FileSize:
  Integer; var AllowExtracting: Boolean);
```

Description

The OnExtract event occurs when StoredFile is ready to be extracted.

Write an OnExtract event handler to determinate whether this file could be extracted. Set *AllowExtracting* parameter to False to pervent file from extracting.

Parameters

FileName	file name of the file to extract.
FileSize	size (in bytes) of the file to extract.
AllowExtracting	determines whether the file (specified by *FileName* parameter) could be extracted.

Example

```
procedure TForm1.FileStorageExtract(Sender: TObject;
   FileName: String; FileSize: Integer;
   var AllowExtracting: Boolean);
begin
  if UpperCase(ExtractFileExt(FileName)) = '.TXT' then
    AllowExtracting := False;
end;
```

9.19.4.2 OnExtracted

Applies to

acFileStorage component.

© 1998-2006, UtilMind Solutions®

Declaration

```
procedure OnExtracted(Sender: TObject; FileName: String; FileSize:
  Integer; Successfully: Boolean);
```

Description

The OnExtracted event occurs once StoredFile has extracted.

Write an OnExtracted event handler to determinate whether this file has successfully extracted from FileStorage. If file has been sucessfully extracted (without write errors) then *Successfully* parameter will be True.

Note

The *Successfully* parameter will be False if file already exists and currently blocked by another application (no write permission). For example, this will happend if you trying to extract executable file which currently running.

Parameters

FileName	file name of the extracted file.
FileSize	size (in bytes) of the extracted file.
Successfully	determines whether the file (specified by *FileName* parameter) has successfully extracted.

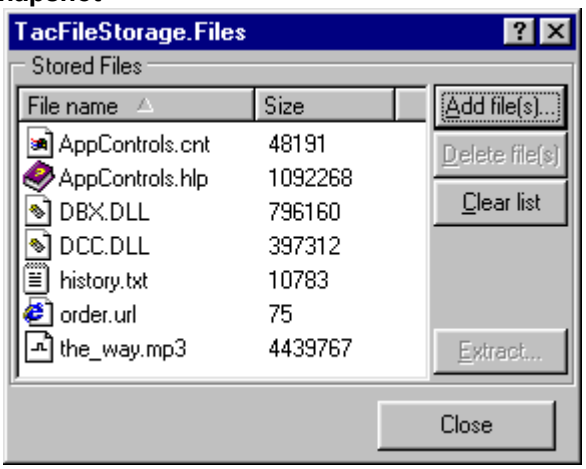
Example

```
procedure TForm1.FileStorageExtracted(Sender: TObject;
  FileName: String; FileSize: Integer; Successfully: Boolean);
begin
  if not Successfully then
   ShowMessage('Error !!');
end;
```

9.19.5 FileStorage designer

The FileStorage designer will help you to upload and organize your files at design-time.

Snapshot



Yeah, all these files (>12 MB) currently uploaded on Delphi form!

9.19.6 Playing .WAVs (example)

Following code demonstrates how to access stored files at run-time without extracting them to hard disk. This sample shows how to play stored .WAV file.

© 1998-2006, UtilMind Solutions®

Delphi:

```
uses MMSystem;

procedure TForm1.Button1Click(Sender: TObject);
var
  StoredFile: TStoredFile;
  Data: TMemoryStream;
begin
  StoredFile := FileStorage1.Files[0];
  Data := StoredFile.Data;

  // the stored data could be retrieved from Data.Memory
  sndPlaySound(Data.Memory, SND_MEMORY or SND_ASYNC);
end;
```

C++ Builder:

```
#include <mmsystem.h>

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TStoredFile *StoredFile = (TStoredFile*)acFileStorage1->Files-
>Items[0];
  TMemoryStream *Data = StoredFile->Data;

  // the stored data could be retrieved from Data->Memory
  sndPlaySound((LPCTSTR)Data->Memory, SND_MEMORY|SND_ASYNC);
}
```

9.20 acFormBackground

9.20.1 TacFormBackground

Overview

The acFormBackground component paints the background (wallpaper) on your form using various gradient effects or bitmap image. Supports 3 smooth gradient effects and 4 styles of drawing the picture: tiled, stretched, tiled with zigzag (like bricks in the wall), and aligning the picture by center.

How to use ?

Drop component on the form and use GradientDirection property to specify direction of gradient effect, or PictureStyle property to specify how to draw the wallpaper.

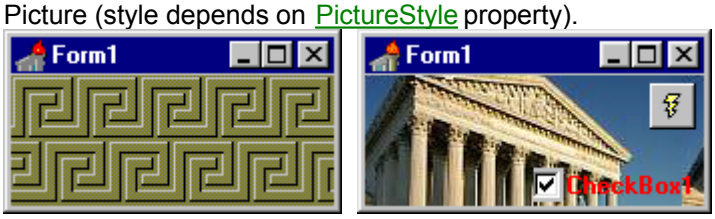
The **Kind** determines what you would like to paint — gradient effect (*bkGradient*) or picture as wallpaper (*bkPicture*).

Snapshots

Gradient effects (depends on GradientDirection property):
gdVertical, *gdHorizontal* and *gdCentral*.

© 1998-2006, UtilMind Solutions®

Picture (style depends on PictureStyle property).



Note
The gradient background / wallpaper does not visible at design-time.

See also
Panels with custom background.

9.20.2 Properties

9.20.2.1 Active

Applies to
acFormBackground, acFormHelp, acFormMagnet, acFormSystemMenu, acAwayTimer, acCursorLocker and acIESniffer components.

Declaration
```
property Active: Boolean;
```

Description
The Enabled property controls whether acFormBackground component should paint the gradient effect or picture on form's background.

Set Active to True to show the custom background (with smooth gradient effect or wallpaper specified in Picture property). Otherwise, set it to False to show standard single-tone background with color specified in Color property of form.

To specify the kind of background (gradient or picture) — use Kind property.

See also
Kind property.

9.20.2.2 GradientColor1

Applies to
acFormBackground component.

Declaration
```
property GradientColor1: TColor;
```

Description
Specified the first (left if GradientDirection is horizontal or top if GradientDirection is vertical) gradient color of your form.

See also
GradientColor2 and GradientDirection properties.

9.20.2.3 GradientColor2

Applies to
acFormBackground component.

© 1998-2006, UtilMind Solutions®

Declaration

```
property GradientColor2: TColor;
```

Description

Specified the second (right if GradientDirection is horizontal or bottom if GradientDirection is vertical) gradient color of your form.

See also

GradientColor1 and GradientDirection properties.

9.20.2.4 GradientDirection

Applies to

acFormBackground component.

Declaration

```
type
  TacGradientDirection = (gdVertical, gdHorizontal, gdCentral);

property GradientDirection: TacGradientDirection;
```

Description

Specifies the direction of the gradient effect. Direction can be vertical, horizontal or central. To specify the colors for gradient effect — use GradientColor1 and GradientColor2 properties.

Snapshots

GradientDirection = gdVertical

GradientDirection = gdHorizontal

GradientDirection = gdCentral

See also

GradientColor1, GradientColor2 and Kind properties.

9.20.2.5 Kind

Applies to

acFormBackground component.

Declaration

```
type
  TacFormBackgroundKind = (bkPicture, bkGradient);

property Kind: TacFormBackgroundKind;
```

© 1998-2006, UtilMind Solutions®

Description

The Kind property specifies how to paint the form background: using the picture (bkPicture), or drawing the gradient effect (bkGradient).

If you would like to paint background using the **picture**:

1. Set Kind property to `bkPicture`;
2. Specify the bitmap image in Picture property;
3. Select the style of drawing in the PictureStyle property.

If you wish to show the **gradient effect**:

1. Set Kind property to `bkGradient`;
2. Specify the direction of the effect in GradientDirection property;
3. Choose two colors to draw the effect in GradientColor1 and GradientColor2 properties.

See also

Active, GradientDirection and PictureStyle properties.

9.20.2.6 Picture

Applies to

acFormBackground property.

Declaration

```
property Picture: TPicture;
```

Description

The Picture property specifies the image that appears as form background. The Picture can contains an icon, betafile, bitmap graphic, or user-defined graphic to be displayed as the background (if Kind = bkPicture).

Setting Picture at design time brings up the Picture Editor, which can be used to specify the file that contains the image.

To specify the style of drawing — use PictureStyle property.

See also

PictureStyle and Kind properties.

9.20.2.7 PictureStyle

Applies to

acFormBackground property.

Declaration

```
type
  TacBackgroundPictureStyle = (bsTile, bsStretch, bsZigzag, bsCenter);

property PictureStyle: TacBackgroundPictureStyle;
```

Description

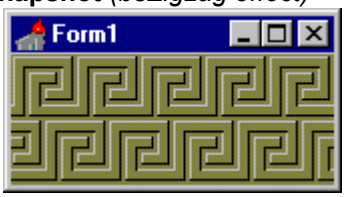
The PictureStyle property specifies the effect used to draw the Picture. There are possible values:

Value	Meaning
bsTile	tiles the bitmap image on form's background;
bsStretch	stretches the image so that it exactly fits the bounds of the form. When the form resized, the image will be resized as well;

© 1998-2006, UtilMind Solutions®

bsZigzag	almost the same as tile effect, but every next row of tile is shifted relatively to previous row, like bricks in the wall;
bsCenter	draws the image in the center of form.

Snapshot *(bsZigzag effect)*



See also

Picture and Kind properties.

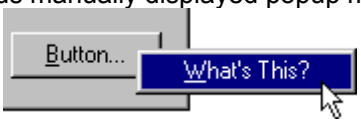
9.21 acFormHelp

9.21.1 TacFormHelp

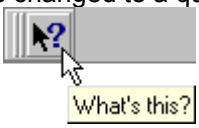
Overview

The acFormHelp component adds the context-sensitive help features to your Delphi/C++ Builder forms without any bulky help files. It traps the context-sensitive help calls and creates its own popup windows from a control's hint. You can choose whether to interpret the hint string as plain text or as kind of rich text allowing you to apply different fonts colors, styles and line breaks. Don't worry about your hints — FormHelp uses the secondary part of a control's hint that is separated by a vertical bar "|". Mouse hints still works as well. With FormHelp, neither help context numbers nor extra help files are required to display context sensitive help. FormHelp's popup windows looks and feels like native context help in standard Microsoft's applications. Even if you're using regular help files, the FormHelp will be great addition and vice versa.

The FormHelp can automatically apply the "What's This?" menu item to every control with context-sensitive help in the secondary part of hint, and invokes the context help either after pressing the "Help" button on the title bar or after selecting "What's This?" menu item. This popup menu is displayed dynamically. TacFormHelp recognizes controls that have their own popup menu and even regards manually displayed popup menus or popup windows.



If the "What's this?" popup menu and button on the title bar of your form still is not enough for you, you can also point the toolbar button and menu item in the main menu or even system menu which can handle the context-help as well. When user clicks this button or menu item, the cursor will be changed to a question mark with a pointer.



If the user then clicks a control in the form, the control receives a WM_HELP message to show the context-sensitive help taken from secondary part of the Hint property.

For easement of context-help authoring, component contains a WYSIWYG help designer (drop acFormHelp onto your form and try to edit secondary part of Hint property of any visible control) and may have an additional button on form's title bar.

© 1998-2006, UtilMind Solutions®

How to use ?

Just drop acFormHelp component onto your form and edit the Hint property of any visual control (like TButton, TCheckBox or TGroupBox). Type any text in the secondary part of Hint, and make additional button on form's title bar visible (if needed). Recompile and execute your application. Now your form will traps all context-sensitive help calls and when user clicks help button on form's title bar and clicks on control, popup window with context-sensitive help will be displayed. To arrange the width of popup window - use PopupWidth property and OnShow event.

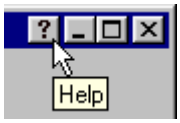
You can also show help-window "manually", using ShowHelp, ShowHelpFromControl and ShowHelpFromPoint methods.

Properties

Active	Activity flag of the component;
AdjustPopupWidth	Whether the width of the popup window with help-message should be adjusted accordingly to the width of visible text in the the popup window;
CaptionButton	Additional button on the form's title bar. Works even if form style don't allow this button (see snapshot below);
Color	Background color of popup window;
Cursor	Cursor image for popup window with help message;
DelayInterval	Specifies the interval before auto-hiding of the popup window;
Font	Font of context-sensitive help text;
PopupWidth	Controls the width of popup window;
SystemMenu	Appies the "What's This?" menu item to the system menu;
TextStyle	Specifies the text style (formatted or plain) and tag brackets ('**[**' and '**]**' by default)
WhatsThis	Whether the controls contains the "What's This?" menu item.

Derived features

acFormHelp is successor of acCustomCaptionButton class. It have a built-in support of the corresponding caption button on the form's title bar and the menu item in the system menu, associated with button.



Notes

Avoiding the Application.Hint problem.

See also

acRichLabel component.

9.21.2 Properties

9.21.2.1 Active

Applies to

acFormHelp, acFormBackground, acFormMagnet, acFormSystemMenu, acAwayTimer, acCursorLocker and acIESniffer components.

Declaration

© 1998-2006, UtilMind Solutions®

```
property Active: Boolean;
```

Description

The Active property controls whether the FormHelp component is Active and can display the context sensitive help by hooking the WM_HELP application message. If Active is True, then when you click ? button on the title bar then clicks any control with specified hint, FormHelp will show popup window which looks and feels like native Windows help.

9.21.2.2 AdjustPopupWidth

Applies to

acFormHelp component.

Declaration

```
property AdjustPopupWidth: Boolean;
```

Description

The AdjustPopupWidth property controls whether the width of the popup window with help-message should be adjusted accordingly to the width of visible text in the the popup window.

When the AdjustPopupWidth is True, the width of right margin (between the right edge of popup window and text) will be equal to the width of left margin.

Snapshot 1 *(AdjustPopupWidth = True)*

Snapshot 2 *(AdjustPopupWidth = False, PopupWidth = 280)*

See also

PopupWidth property.

9.21.2.3 CaptionButton

Applies to

acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost and acFormRoller components.

Declaration

```
property CaptionButton: TacACaptionButton;
```

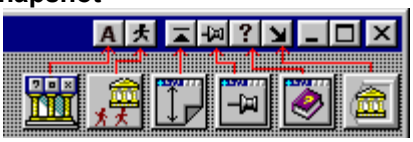
Description

The CaptionButton is the set of properties for managing an additonal buttons on the form's title bar:

BtnOrder	controls the button placement order on the title bar;
Cursor	specifies the cursor image when button released;
CursorDown	specifies the cursor image when button pressed;
Enabled	enables or disables button;
Hint	specifies the button's tooltip;
SeparatorWidth	width between current and previous buttons;
ShowHint	enables or disables the tooltip (hint);
Visible	shows or hides the button from the title bar.

© 1998-2006, UtilMind Solutions®

Snapshot



9.21.2.3.1 BtnOrder

Applies to

acFormHelp, acTrayIcon, acAppAutoRun, acFormTopmost, acFormRoller components as subproperty of CaptionButton.

Declaration

```
property BtnOrder: TacBtnOrder;
```

Description

The BtnOrder property indicates position of caption button, the order in which buttons will be displayed. BtnOrder calculates from right side of title bar to left side (see snapshot below).

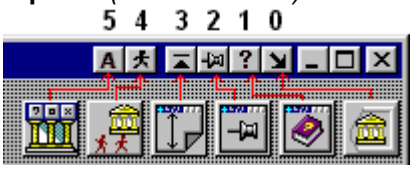
Initially, the button order is always the order in which the components were added to the form, but you can change this by changing the BtnOrder property. The value of the BtnOrder property is unique for each ancestor of the acCaptionButton on the form (like TabOrder property of ancestors of TWinControl). The first caption button added to the form has a BtnOrder value of 0, the second is 1, the third is 2, and so on. These values determine where a control is on the title bar.

If you attempt to give a button a BtnOrder value greater than the number of caption buttons on the form minus one (because numbering starts with 0), AppControls won't accept the new value, but will enter the value that assures the component will be the last in the button order.

The caption button with the BtnOrder value of 0 is the custom button which placed on right side of title bar, after regular system buttons.

To separate the buttons by additional empty space — use SeparatorWidth property. For example snapshot below shows that acAppAutoRun component have BtnOrder = 4 (fifth position from the left of system buttons) and SeparatorWidth = 4 pixels.

Snapshot *(buttons orders):*



See also

SeparatorWidth property and acCaptionButton component.

9.21.2.3.2 Cursor

Applies to

acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller components as subproperty of CaptionButton.

Declaration

```
property Cursor: TCursor;
```

© 1998-2006, UtilMind Solutions®

Description
The Cursor property controls the mouse cursor shape used when the mouse moves over CaptionButton in released state.

See also
CursorDown property and acCaptionButton component.

9.21.2.3.3 CursorDown

Applies to
acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller components as subproperty of CaptionButton.

Declaration
```
property CursorDown: TCursor;
```

Description
The CursorDown property controls the mouse cursor shape used when the mouse moves over CaptionButton in pressed state. Value of CursorDown property for the acFormHelp component are always the same as CursorHelp property

See also
CursorHelp property.

9.21.2.3.4 Enabled

Applies to
acFormHelp, acTrayIcon, acAppAutoRun, acFormTopmost, acFormRoller components as subproperty of CaptionButton.

Declaration
```
property Enabled: Boolean;
```

Description
The Enabled property controls whether the button on the title bar responds to mouse and keyboard messages. If Enabled is True, button responds normally. If Enabled is False, button become disabled and user can not press that button.

See also
Visible property of CaptionButton and acCaptionButton component.

9.21.2.3.5 Hint

Applies to
acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller components as subproperty of CaptionButton.

Declaration
```
property Hint: String;
```

Description
The Hint property is the text string that can appear when mouse pointer moves over the caption button.

If the CaptionButton's ShowHint property is False, the Help Hint won't appear, but the other hints still will.

© 1998-2006, UtilMind Solutions®

Default Hint string for acFormHelp component is 'Help'. You may change this string at run- or design-time if you're writting multilingual program.

See also
acCaptionButton component and ShowHint property.

9.21.2.3.6 SeparatorWidth

Applies to
acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller components as subproperty of CaptionButton.

Declaration
```
property SeparatorWidth: Byte; // in screen pixels
```

Description
The SeparatorWidth property controls the space between current and previous by order caption button in screen pixels.

See also
BtnOrder property and acCaptionButton component.

9.21.2.3.7 ShowHint

Applies to
acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller components as subproperty of CaptionButton.

Declaration
```
property ShowHint: Boolean;
```

Description
The ShowHint property controls whether CaptionButton can show the Hint string when mouse pointer moves over button. If ShowHint is True, Hint will appears.

See also
acCaptionButton component and Hint property.

9.21.2.3.8 Visible

Applies to
acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller components as subproperty of CaptionButton.

Declaration
```
property Visible: Boolean;
```

Description
The Visible property determines whether the caption button currently visible on the title bar. Make Visible property True if you would like to show the button on title bar, or False if you would like to hide the button.

When you changing the Visible property, placements of others button will automacially recalculated and other buttons with higher BtnOrder will be shifted to the right side of the title bar, filling the empty space.

© 1998-2006, UtilMind Solutions®

See also
Enabled property of CaptionButton and acCaptionButton component.

9.21.2.4 Color

Applies to
acFormHelp component *(in general, to all components, successors of TControl)*.

Declaration
```
property Color: TColor;
```

Description
The Color property specifies the background color of popup window with context-sensitive help. Default value is clContextHelp.

See also
ShadowColor property and Additional Colors help topic.

9.21.2.5 Cursor

Applies to
acFormHelp component *(in general, to all components, successors of TControl)*.

Declaration
```
property Cursor: TCursor;
```

Description
The Cursor property identifies the cursor image used when the mouse pointer moves over the popup window which shows the context-sensitive help.

9.21.2.6 CursorHelp

Applies to
acFormHelp component *(in general, to all components, successors of TControl)*.

Declaration
```
property CursorHelp: TCursor;
```

Description
The CursorHelp property specifies the cursor image used to point the control to invoke the help for it. Value of CursorHelp are always the same as value of CursorDown property of CaptionButton structure.

See also
Cursor and CursorDown properties of CaptionButton structure.

9.21.2.7 DelayInterval

Applies to
acFormHelp component.

Declaration
```
property DelayInterval: Word; // in milliseconds
```

Description
The DelayInterval property specifies the delay interval before popup window with context-sensitive help will automatically disappered. After expiration of specified interval, the popup window will be

© 1998-2006, UtilMind Solutions®

automatically closed, even if user did not clicked the mouse buttons.

9.21.2.8 Font

Applies to
acFormHelp component *(in general, to all components, successors of TControl)*.

Declaration
```
property Font: TFont;
```

Description
The Font property is a font object that controls the text attributes for text at context-sensitive help windows.

See also
ParentFont property.

9.21.2.9 Margins

Applies to
acFormHelp component.

Declaration
```
type
  TacFormHelpMargins = class
  published
    property Horizontal: Byte default 10;
    property Vertical: Byte default 4;
  end;
```

Description
The Margins specifies width and height (in pixels) for space between the border and the text in the help window.

See also
MarginHorz and MarginVert properties of acRichLabel component.

9.21.2.9.1 Horizontal

Applies to
acFormHelp component.

Declaration
```
property Horizontal: Byte;
```

Description
The Horizontal property specifies the width (in pixels) for space between the border and the text of help window.

See also
Vertical property.

9.21.2.9.2 Vertical

Applies to
acFormHelp component.

© 1998-2006, UtilMind Solutions®

Declaration

```
property Vertical: Byte;
```

Description

The Vertical property specifies the height (in pixels) for space between the border and the text of help window.

See also

Horizontal property.

9.21.2.10ParentFont

Applies to

acFormHelp component.

Declaration

```
property ParentFont: Boolean;
```

Description

The ParentFont property determines whether the FormHelp should use font of its parent form.

Set ParentFont to true in order to ensure that the text on context-sensitive help window will have the same font as on form and to have a uniform appearance for entire controls on the form. For example, if ParentFont is True, changing the form's Font property to 12-point Courier causes the context-sensitive help windows to to use that font.

When the value of a FormHelp's Font property changes, ParentFont becomes False automatically.

When ParentFont is true for a form, the form uses the default font.

See also

Font property.

9.21.2.11PopupWidth

Applies to

acFormHelp component.

Declaration

```
type
  TacFormHelpPopupWidth = 40..1000;

property PopupWidth: TacFormHelpPopupWidth; // in pixels
```

Description

The PopupWidth property controls the width of popup window with context sensitive help. PopupWidth specifies the width of popup window for all controls on current form. It can assign values in range from 40 up to 1000 pixels. By default this value is 300 pixels.

See also

AdjustPopupWidth property.

9.21.2.12ShadowColor

Applies to

acFormHelp and acRichLabel components.

© 1998-2006, UtilMind Solutions®

Declaration

```
property ShadowColor: TColor;
```

Description

The ShadowColor property specifies the color of shadow (dots that imitate the window shadow) for popup windows with context-sensitive help. ShadowColor used only if ShowShadow property is True.

See also

ShowShadow and Color properties.

9.21.2.13 ShowShadow

Applies to

acFormHelp and acRichLabel components.

Declaration

```
property ShowShadow: Boolean;
```

Description

The ShowShadow property determines whether the FormHelp component should display a shadow (dots that imitate a shadow) behind popup window with context-sensitive help. You may also specify the color for shadow in ShadowColor property.

See also

ShadowColor property.

9.21.2.14 SystemMenu

Applies to

acCaptionButton component and other components that uses TacCustomCaptionButton as ancestor class - acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller.

Declaration

```
property SystemMenu: TacSystemMenu;
```

Description

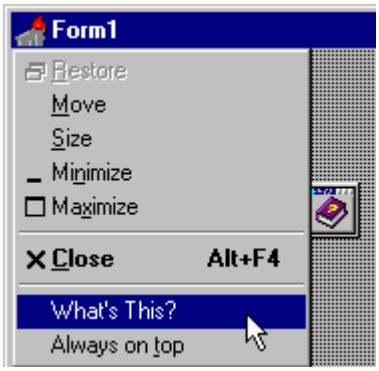
The System Menu is the popup menu that appears when you click on the program icon on the upper-left corner of form's title bar. The SystemMenu property is the list of properties that manages the menu item associated with current button on the form's title bar.

Properties

ApplyToMenu	applies or removes the menu item associated with current button from the system menu;
Caption	text for the title of menu item;
Position	position of current menu item in the system menu;
Separators	specifies the separators for menu item, to separatate it from previous and / or next menu item of the system menu.

Snapshot

© 1998-2006, UtilMind Solutions®



See also
acFormSystemMenu component.

9.21.2.14.1 ApplyToMenu

Applies to
acCaptionButton component and all other components which uses CaptionButton as base ancestor class - acFormHelp, acAppAutoRun, acTrayIcon, acFormTopmost, acFormRoller. ApplyToMenu is subproperty of SystemMenu.

Declaration
```
property ApplyToMenu: Boolean;
```

Description
The ApplyToMenu property controls whether the CaptionButton currently have the menu item assiciated with this button. Set ApplyToMenu property to True to add according menu item (with text specified by Caption property, in position specified by Position property) to the form's system menu and False to remove.

See also
Caption property.

9.21.2.14.2 Caption

Applies to
acCaptionButton component and all other components which uses CaptionButton as base ancestor class - acFormHelp, acAppAutoRun, acTrayIcon, acFormTopmost, acFormRoller. Caption is subproperty of SystemMenu.

Declaration
```
property Caption: String;
```

Description
The Caption property specifies the text for menu item in the form's system menu, associated with current caption button on the title bar. If Caption is not specified, text from Hint property will be taken as item title.

See also
Hint property of CaptionButton.

9.21.2.14.3 Position

Applies to
acCaptionButton component and all other components which uses CaptionButton as base ancestor

© 1998-2006, UtilMind Solutions®

class - acFormHelp, acAppAutoRun, acTrayIcon, acFormTopmost, acFormRoller. Position is subproperty of SystemMenu.

Declaration

```
property Position: Word;
```

Description

The Position property determines the position for menu item in the system menu associated with current caption button on the form's title bar. Change the Position value to move current menu item by system menu.

Maximum value for Position property is the current maximum number of items in the system menu (since orders for menu items starts with 0).

See also

Separators property.

9.21.2.14.4 Separators

Applies to

acCaptionButton component and all other components which uses CaptionButton as base ancestor class - acFormHelp, acAppAutoRun, acTrayIcon, acFormTopmost, acFormRoller. Separators is subproperty of SystemMenu.

Declaration

```
type
  TacMenuSeparators = set of (seBefore, seAfter);

property Separators: TacMenuSeparators;
```

Description

The Separators property specifies the separators for current menu item to separate it from previous and / or next menu items of the system menu. To set separator before menu item — set `seBefore` to True. To set separator after menu item — set `seAfter` to True.

Specify Separators to separate current menu item in the system menu from another menu items.

See also

Position property.

9.21.2.15TextStyle

Applies to

acFormHelp, acInfoBox, acRichLabel, acHeader, acHeaderPanel and acPicturePanel components.

Declaration

```
type
  TacTextStyle = class
  published
    property PlainText: Boolean default False;
    property TagOpen: Char default '[';
    property TagClose: Char default ']';
  end;
```

Description

The TextStyle structure used to specify the style of text displayed by this component.

© 1998-2006, UtilMind Solutions®

The PlainText property specifies whether you would like to use right-formatted text or just flat, plain text. TagOpen and TagClose properties used to specify prefered signs for brackets which identifies begin and end of tags used for rich formatting. By default every tag beginning with '**[**' sign and ending with '**]**' sign.

For example, when PlainText is False, TagOpen is '[' and TagClose is ']', following text:

```
[i]Hello[]  [red][b]World[def]!
```

will looks like:

Hello **World**!

Otherwise, if PlainText is True, the tags will not be used and users will see just flat text, including all tags.

See also
acRichLabel component and Rich-formatting tags overview.

9.21.2.15.1 PlainText

Applies to
acFormHelp, acRichLabel, acInfoBox, acHeader, acHeaderPanel and acPicturePanel components as subproperty of TextStyle structure.

Declaration
```
property PlainText: Boolean;
```

Description
The PlainText property controls whether the text should be displayed as rich text (if PlainText is False) or as plain, non-formatted text (if PlainText is True). When PlainText is True, the text will be displayed as usual flat text and nor tags will be used.

See also
TagOpen and TagClose properties;
Using tags in context-sensitive help topic.

9.21.2.15.2 TagClose

Applies to
acFormHelp, acRichLabel, acInfoBox, acHeader, acHeaderPanel and acPicturePanel components as subproperty of TextStyle structure.

Declaration
```
property TagClose: Char;
```

Description
The TagClose property specifies the sign which marks the ending of formatting area of rich-text. The rich formatting tags of AppControls pack are similar to brackets "<>" of tags in HTML format. Default TagClose value is "**]**" sign.

Examples of standard tags
`[b]` - makes text after this tag **bold**.
`[i]` - makes text after this tag *italic*
`[biu]` - makes text after this tag ***bold, italic and underlined***.
`[]` - clears text formatting.
`[red]` - makes text after this tag red.
`[lime]` - makes text after this tag light green (lime).

© 1998-2006, UtilMind Solutions®

`[def]` or `[default]` - returns the text formatting to default state as specified in Font property.

Note
If PlainText property is True, nor rich text will be displayed. All formating will be shown as usual flat text.

See also
TagOpen and PlainText properties and example of rich-text formatting of context help.

9.21.2.15.3 TagOpen

Applies to
acFormHelp, acRichLabel, acInfoBox, acHeader, acHeaderPanel and acPicturePanel components as subproperty of TextStyle structure.

Declaration
```
property TagOpen: Char;
```

Description
The TagOpen property specifies the sign which marks the beginning of rich-text formatting area. The rich formatting tags of AppControls pack are similar to brackets "<>" of tags in HTML format. Default TagOpen value is "**[**" sign.

Examples of standard tags
`[b]` - makes text after this tag **bold**.
`[i]` - makes text after this tag *italic*
`[biu]` - makes text after this tag ***bold, italic and underlined***.
`[]` - clears text formatting.
`[red]` - makes text after this tag red.
`[lime]` - makes text after this tag light green (lime).
`[def]` or `[default]` - returns the text formatting to default state as specified in Font property.

See also
TagClose and PlainText properties and example of rich-text formatting of context help.

9.21.2.16UseF1

Applies to
acFormHelp component.

Declaration
```
property UseF1: Boolean;
```

Description
The UseF1 property specifies whether the FormHelp should invoke the context-sensitive help from the control under mouse pointer when user press F1 key.

Set UseF1 to True to let the FormHelp to process F1 key presses, or set it to False to disable this feature.

9.21.2.17WhatsThis

Applies to
acFormHelp component.

Declaration

© 1998-2006, UtilMind Solutions®

```
type
  TacWhatsThis = class
  published
    property Enabled: Boolean;
    property Caption: String; // Caption for "What's this?" menu item.
Use this property for localization
    property MenuItem: TMenuItem;
    property ToolbarButton: TControl;
  end;
```

Description

The WhatsThis structure used to specify whether the acFormHelp component supports the "What's This?" feature (whether every control on the form contains the "What's This?" menu item associated with the context-sensitive help).

When the Enabled property is True, the acFormHelp will add the "What's This ?" menu item to every control of the form with context-sensitive help in the secondary part of Hint property. When user selects the "What's This ?" menu item, the context-sensitive help associated with the control will pop up.

You can also specify the Caption for this menu item (for example, to translate it to another language in the multilingual programs), and specify the external menu item and toolbar button which operates with the built-in context-sensitive help. When user clicks the "What's This?" menu item or button, the cursor will be changed to a question mark with a pointer.

Remarks

The "What's This?" menu will not appears on right-click, if the control have its own popup menu (i.e. Edit or Memo controls).

See also

InvokeHelpPointer method.

9.21.2.17.1 Caption

Applies to

acFormHelp component as subproperty of WhatsThis structure.

Declaration

```
property Caption: String;
```

Description

The Caption property is the caption for "What's This?" popup menu item used to invoke the context-sensitive help. Change the Caption property to translate the "What's This?" question to another language in the international programs.

9.21.2.17.2 Enabled

Applies to

acFormHelp component as subproperty of WhatsThis structure.

Declaration

```
property Enabled: Boolean;
```

© 1998-2006, UtilMind Solutions®

Description

The Enabled property controls whether the FormHelp should apply the "What's This ?" menu item to every control with the context-sensitive help in the secondary part of the Hint property. Set Enabled to True to add the "What's This ?" popup menu to controls with context-sensitive help or False otherwise.

9.21.2.17.3 MenuItem

Applies to

acFormHelp component as subproperty of WhatsThis structure.

Declaration

```
property MenuItem: TComponent;
```

Description

The MenuItem property points to the "What's This?" menu item (in any menu, main or popup). When user clicks this menu item, the cursor will be changed to a question mark with a pointer.

If the user then clicks a control in the form, the control receives a WM_HELP message to show the context-sensitive help taken from secondary part of the Hint property.

When you point this property to the menu item, you don't need to handle OnClick event of this element of menu. The FormHelp will handle if for you automatically. However, if you need to specify *many* "What's this?" menu items — call the InvokeHelpPointer method in the OnClick event handler of this menu item, without pointing this property to it.

As you can see, not only standard MenuItems can be assigned to this property. You associate to it ANY component which has "Checked" or "Down" property (anything that can reflect the state of boolean value).

See also

ToolbarButton property and InvokeHelpPointer method.

9.21.2.17.4 ToolbarButton

Applies to

acFormHelp component as subproperty of WhatsThis structure.

Declaration

```
property ToolbarButton: TComponent;
```

Description

The ToolbarButton property points to the "What's This?" button on the toolbar (or any control, in general). When user clicks this button, the cursor will be changed to a question mark with a pointer.

If the user then clicks a control in the form, the control receives a WM_HELP message. The FormHelp component hooks this message to show the context-sensitive help taken from secondary part of the Hint property.

When you point this property to the control, you don't need to handle OnClick event of this control. The FormHelp will handle if for you automatically. However, if you need to specify *many* "What's this?" buttons — just call the InvokeHelpPointer method in the OnClick event handler of this control, without pointing this property to it.

Snapshot

© 1998-2006, UtilMind Solutions®

See also
MenuItem property and InvokeHelpPointer method.

9.21.3 Methods

9.21.3.1 InvokeHelpPointer

Applies to
acFormHelp component.

Declaration
```
procedure InvokeHelpPointer;
```

Description
The InvokeHelpPointer method changes the cursor to a question mark with a pointer. If the user then clicks a control in the form, the control receives a message to show the context-sensitive help taken from secondary part of the Hint property.

Example
```
procedure TForm1.Button2Click(Sender: TObject);
begin
  acFormHelp1.InvokeHelpPointer;
end;
```

The InvokeHelpPointer procedure is equal to SendMessage(FormHandle, WM_SYSCOMMAND, SC_CONTEXTHELP, 0). So example above is equal to following code:

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  SendMessage(Handle, WM_SYSCOMMAND, SC_CONTEXTHELP, 0);
end;
```

See also
MenuItem and ToolbarButton properties of the WhatsThis structure.

9.21.3.2 ShowHelp

Applies to
acFormHelp component.

Declaration
```
procedure ShowHelp(Help: String);
```

Description
The ShowHelp method displays the popup window with text specified by Help string parameter from current mouse position. Text string can contain either plain or rich text.

Example
```
procedure TForm1.Whatsthis1Click(Sender: TObject);
begin
  acFormHelp1.ShowHelp('Hello world!');
end;
```

© 1998-2006, UtilMind Solutions®

See also
ShowHelpFromControl and ShowHelpFromPoint methods;
Rich formatting of context-sensitive help.

9.21.3.3 ShowHelpFromControl

Applies to
acFormHelp component.

Declaration
```
procedure ShowHelpFromControl(Control: TControl);
```

Description
The ShowHelpFromControl method displays the popup window with context-sensitive help, from any visible control specified by Control parameter. Text for popup window will be taken from secondary part of Hint property of specified Control.

Example
```
procedure TForm1.Button2Click(Sender: TObject);
begin
   FormHelp.ShowHelpFromControl(Button2);
end;
```

See also
ShowHelp and ShowHelpFromPoint methods.

9.21.3.4 ShowHelpFromPoint

Applies to
acFormHelp component.

Declaration
```
procedure ShowHelpFromPoint(ShowPoint: TPoint; Help: String);
```

Description
The ShowHelpFromPoint method displays the popup window with context-sensitive help, from point specified by ShowPoint parameter and text specified by Help string.

Example
```
procedure TForm1.Whatsthis1Click(Sender: TObject);
var
  Point: TPoint;
begin
  Point.X := Left + 150;
  Point.Y := Top + 100;
   FormHelp.ShowHelpFromPoint(Point, 'Hello world!');
end;
```

See also
ShowHelp and ShowHelpFromControl methods;
Rich formatting of context-sensitive help.

© 1998-2006, UtilMind Solutions®

9.21.4 Events

9.21.4.1 OnButtonClick

Applies to
acFormHelp component.

Declaration
```
property OnButtonClick: TNotifyEvent;
```

Description
The OnButtonClick event occurs when user click additional button on form's title bar. This event, unfortunately, does not works with regular button.

See also
CaptionButton property, OnShow and OnHide events.

9.21.4.2 OnHide

Applies to
acFormHelp component.

Declaration
```
type
  TacFormHelpOnHideEvent = procedure(Sender: TObject; HelpControl:
TControl) of object;

property OnHide: TacFormHelpOnHideEvent;
```

Description
The OnHide event occurs before hiding of the popup window with context-sensitive help. Write the OnHide event handler to make some special processing before the help window disappears from screen.

Use *HelpControl* parameter to get pointer to control from wich with the context-sensitive help was taken.

See also
OnShow event.

9.21.4.3 OnShow

Applies to
acFormHelp component.

Declaration
```
type
  TacFormHelpOnShowEvent = procedure(Sender: TObject; HelpControl:
TControl; var HelpMessage: string) of object;

property OnShow: TacFormHelpOnShowEvent;
```

Description
The OnShow event occurs before appearing of the popup window with context-sensitive help. Write the OnShow event handler to make some special processing before the help window became visible on screen.

© 1998-2006, UtilMind Solutions®

Use *HelpControl* parameter to get pointer to control from wich with the context-sensitive help was taken.

Use *HelpMessage* parameter to determinate or change the text of context-sensitive help which is about to be displayed.

Example

```
procedure TForm1.FormHelpShow(Sender: TObject; HelpControl: TControl;
   var HelpMessage: string);
begin
  if HelpControl.Name = 'Edit2' then
   begin
    FormHelp.DelayTime := 2000; // Two seconds delay
    FormHelp.Color := $00FFFFEC;
    FormHelp.Font.Name := 'Arial';
    FormHelp.Font.Size := 8;
   end;
  if HelpMessage = '.' then HelpMessage := 'Hello World!';
end;
```

See also

OnHide event.

9.21.4.4 OnWhatsThis

Applies to

acFormHelp component.

Declaration

```
type
  TacWhatsThisEvent = procedure(Sender: TObject; HelpControl: TControl;
MousePos: TPoint; var ShowPopup, ShowHelp: Boolean) of object;

property OnWhatsThis: TacWhatsThisEvent;
```

Description

The OnWhatsThis event occurs when user clicks right mouse button (*right clicks*) on the control with context-sensitive help in the secondary part of Hint property.

Use *HelpControl* parameter to get pointer to control from wich with the context-sensitive help was taken. The coordinates where user clicks the mouse button passed with *MousePos* parameter.

If you don't want to invoke the "What's This ?" popup menu for some certain controls — make *ShowPopup* parameter False in the event handler (see example below). However, even if you don't want to show the popup menu but still would like to show the context-sensitive help, then make *ShowHelp* parameter True.

Example

```
procedure TForm1.acFormHelp1WhatsThis(Sender: TObject;
  HelpControl: TControl; MousePos: TPoint; var ShowPopup,
  ShowHelp: Boolean);
begin
  if HelpControl is TLabel then
   begin
    ShowPopup := False; // don't invoke "What's This?" for all labels
    if HelpControl = Label1 then
```

© 1998-2006, UtilMind Solutions®

```
      ShowHelp := True; // show the context-sensitive help for "Label1"
on right click
   end;
end;
```

See also
WhatsThis structure.

9.21.5 Rich-formatting tags

If you would like to make rich-text formatting of your context sensitive help you can:
1. Use built-in WYSIWYG context-sensitive help designer (drop acFormHelp onto your form and try to edit secondary part of Hint property of any visible control like TButton, TCheckBox or TGroupBox).
2. Make formatting manually in the secondary part of Hint property of any visible control, using special tags:

Style tags *(for changing the style attributes)*:

`[B]` - Bold font.	Example: "[b]**Hello world**[]"
`[I]` - Italic font.	Example: "[i]*Hello world*[] "
`[U]` - Underlined font.	Example: "[u]Hello world[]"
`[S]` - Striked font.	Example: "[s]~~Hello world~~[]"
`[]` - normal, regular font	

You can also combine several styles simultaneously:

`[BI]` - Bold+Italic.	Example: "[bi]***Hello world***[]"
`[US]` - Underline+Striked	Example: "[us]~~Hello world~~[]"
`[BIUS]` - Bold+Italic+Underline+Striked	Example: "[bius]~~***Hello world***~~[]"

Color tags *(for changing the text colors)*:

`[Black]`	Example: the [black]FormHelp[def] component
`[Maroon]`	Example: the [maroon]FormHelp[def] component
`[Green]`	Example: the [green]FormHelp[def] component
`[Olive]`	Example: the [olive]FormHelp[def] component
`[Navy]`	Example: the [navy]FormHelp[def] component
`[Purple]`	Example: the [purple]FormHelp[def] component
`[Teal]`	Example: the [teal]FormHelp[def] component
`[Gray]`	Example: the [gray]FormHelp[def] component
`[Silver]`	Example: the [silver]FormHelp[def] component
`[Red]`	Example: the [red]FormHelp[def] component
`[Lime]`	Example: the [lime]FormHelp[def] component
`[Yellow]`	Example: the [yellow]FormHelp[def] component
`[Blue]`	Example: the [blue]FormHelp[def] component
`[Fuchsia]`	Example: the [fuchsia]FormHelp[def] component
`[Aqua]`	Example: the [aqua]FormHelp[def] component
`[White]`	Example: the [white]FormHelp[def] component

`[Def]` or `[Default]` - these tags will return text color and style to default state, as specified in Font property.

Note
If TextStyle.PlainText property is True, the rich text will not be displayed. All formating will be shown as usual flat text.

© 1998-2006, UtilMind Solutions®

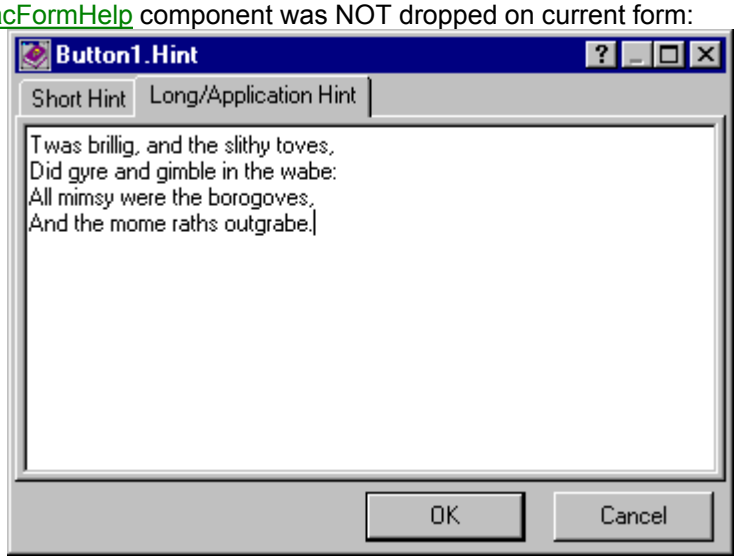
See also
TextStyle structure and TagOpen, TagClose and PlainText properties.

9.21.6 FormHelp designer / Hint property editor

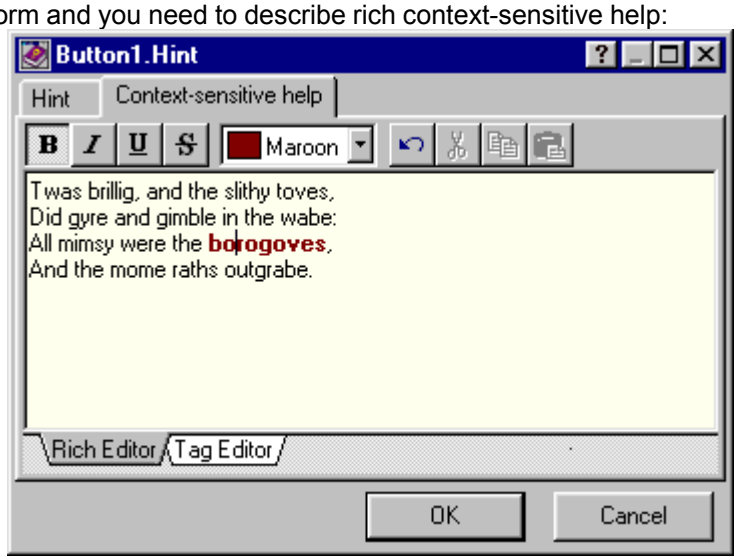
When you installing AppControls / acFormHelp, the FormHelp designer will be installed to the Delphi IDE additionally as new property editor for all "Hint" properties (doubleclick the Hint line in Object Inspector to see the designer).

FormHelp Designer let you to

1. Edit first and secondary part of hint separately, without separating parts by "|" character — if acFormHelp component was NOT dropped on current form:



2. Edit Hint and Context-sensitive help — if acFormHelp component has been dropped on current form and you need to describe rich context-sensitive help:



The "FormHelp designer" will overwrite all previously installed editors for the Hint property.

See also
Description of acFormHelp component.

© 1998-2006, UtilMind Solutions®

9.21.7 Application.Hint problem

All previous FormHelp versions (before v3.0), used first part of hint for displaying the context-help. This makes a big problem, because forms with FormHelp could not show normal hints. To avoid this problem, FormHelp v3.0+ uses only secondary part of hint, also known as Application.Hint. However, unfortunately, sometimes this may lead to another problem.

For example, you are using the *Application.OnHint* event to display the secondary part of hint in the status bar. In this case, the status bar will display the context-sensitive help but with all formatting tags:

To avoid new problem you may use following code:

```
// let's say you would like hook the Application hints... ShowHint
procedure described in the public section of TMainForm class
procedure TMainForm.FormCreate(Sender: TObject);
begin
  Application.OnHint := ShowHint;
end;

procedure TMainForm.ShowHint(Sender: TObject);
begin
  { for example, status bar contains several sections to display  some
useful information but we would like to switch it to the  simple panel
mode to show second part of hint for menu items }
  if (Length(Application.Hint) > 0) and // if Application.Hint is not
empty
     (Copy(Application.Hint, 1, 1) <> '[') then { super kludge. '['
character means that second part of hint is the context-sensitive help.
The '[' sign opens the tag for text formatting. In this case we don't
want to display this text in the status bar. }
   begin
    StatusBar.SimplePanel := True;
    StatusBar.SimpleText := Application.Hint;
   end
  else StatusBar.SimplePanel := False;
end;
```

We really think that second part is more convenient for context-sensitive help than first, because it let us to show commonly used regular hints as well. For example, we have a toolbar with some buttons, and we would like to show both, normal hints and the context-sensitive help. However, if the Application.Hint problem critical for you, we appreciate some feedback with ideas and suggestions! Email us to info@appcontrols.com.

9.22 acFormHints

9.22.1 TacFormHints

Overview

The acFormHints component applies the balloon hints for your controls instead of regular, rectangular hints. Balloon hints are displayed in a cartoon-style "balloon" with rounded corners and a stem pointing to the tool. They can be either single- or multi-line. You can specify the default balloon direction, text alignment and degree of transparency and other features that affect on hint appearance. One component changes appearance for all hints in entire application.

© 1998-2006, UtilMind Solutions®

new! You can select the style of hint dynamically at run-time using OnShowHint event. For example you can select the cartoon-style hints for some certain controls but standard "boxy" style for all other controls.

new! Now you can use MaxWidth property to control the maximum width of balloon and wrap the long strings to next line.

Basic properties

Active	whether the FormHints is currently being in active state and can display balloon-like hints.
BiDiMode	layout in right to left reading order for bi-directional text when the font selected is a Hebrew or Arabic font (bdRightToLeft).
ColorBackground, ColorBorder, ColorShadow	specifies colors for various parts of balloon hint.
Direction	specifies default direction for balloon.
MaxWidth	specifies the maximum width of balloon. Allows to wrap the long strings to next line.
Rounded	appearance of baloon corners. Can be rectangular or rounded.
ShowArrow, ShowShadow	whether the component should display arrow or shadow.
TextAlignment	controls the text alignment within a balloon window
Transparent	makes background of balloon transparent
TransparencyRate	controls the degree of transparency for hint's background.

9.22.2 Properties

9.22.2.1 Active

Applies to
acFormHints component.

Declaration
```
property Active: Boolean;
```

Description
The Active property controls whether the FormHints component is currently being in active state and can display balloon-like hints. If Active is True — FormHints component will hook all hint messages and show hints in balloon window. If Active is False — only regular hint boxes will be shown.

9.22.2.2 BiDiMode

Applies to
acFormHints component.

Declaration
```
type
  TacSimpleBiDiMode = (bdLeftToRight, bdRightToLeft);

property BiDiMode: TacSimpleBiDiMode;
```

Description
The BiDiMode property controls whether the text for balloon hints can be layout in right to left reading order for bi-directional text when the font selected is a Hebrew or Arabic font

© 1998-2006, UtilMind Solutions®

(bdRightToLeft). The default reading order is left to right (bdLeftToRight).

See also
Font property.

9.22.2.3 ColorBackground

Applies to
acFormHints component.

Declaration
```
property ColorBackground: TColor;
```

Description
The ColorBackground property determines the background color for a balloon hints. Default value is clInfoBk.

See also
ColorBorder, ColorShadow and Font properties.

9.22.2.4 ColorBorder

Applies to
acFormHints component.

Declaration
```
property ColorBorder: TColor;
```

Description
The ColorBorder property determines the border color for a balloon hints. Default value is clBlack.

See also
ColorBackground, ColorShadow and Font properties.

9.22.2.5 ColorShadow

Applies to
acFormHints component.

Declaration
```
property ColorShadow: TColor;
```

Description
The ShadowColor property specifies the shadow color (dots that imitate the window shadow) for popup windows with context-sensitive help. ShadowColor used only if ShowShadow property is True.

See also
ColorBackground, ColorShadow and Font properties.

9.22.2.6 Direction

Applies to
acFormHints component.

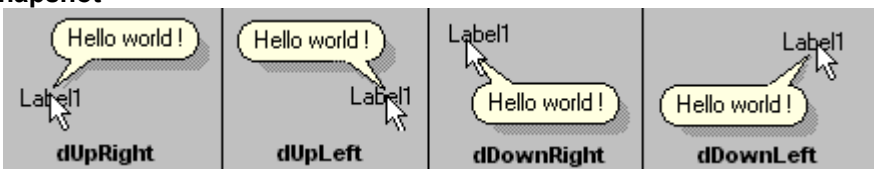
Declaration
```
type
```

© 1998-2006, UtilMind Solutions®

```
  TacHintDirection = (dUpRight, dUpLeft, dDownRight, dDownLeft);

property Direction: TacHintDirection;
```

Description
The Direction property determines default direction for balloon hint.

Snapshot



9.22.2.7 Font

Applies to
acFormHints component.

Declaration
```
property Font: TFont;
```

Description
The Font property is a font object that controls the attributes of hint text. To modify a font, you change the value of the Color, Name, Size, or Style properties of the font object.

See also
ColorBackground, MaxWidth, TextAlignment and BiDiMode properties.

9.22.2.8 MaxWidth

Applies to
acFormHints component.

Declaration
```
property MaxWidth: Word default 0;
```

Description
The MaxWidth property used to control the maximum width of balloon-style hints and wrap long strings to next line, to fit the maximum width of balloon.

If MaxWidth is zero, the hints can have unlimited width, so lines are not wrapping.

This property available since Delphi 3 / BCB 3 or higher. Unfortunately the wrapping of long hints is not allowed in Delphi 2.

See also
Font and TextAlignment properties.

9.22.2.9 Rounded

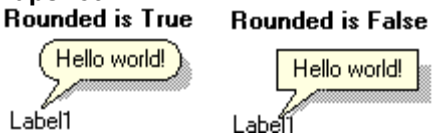
Applies to
acFormHints component.

Declaration
```
property Rounded: Boolean;
```

© 1998-2006, UtilMind Solutions®

Description
The Rounded property controls whether the corners of balloon hint should be rounded (if True), or rectangular as in regular hint window (if False).

Snapshot



See also
RoundRadius property.

9.22.2.10 RoundRadius

Applies to
acFormHints component.

Declaration
```
property RoundRadius: Word;
```

Description
The RoundRadius property determines the radius for rounded corners of ballon hints.

See also
Rounded property.

9.22.2.11 ShowArrow

Applies to
acFormHints component.

Declaration
```
property ShowArrow: Boolean;
```

Description
The ShowArrow property controls whether the FormHints component should display the arrow for balloon hints.

See also
ShowShadow property.

9.22.2.12 ShowShadow

Applies to
acFormHints component.

Declaration
```
property ShowShadow: Boolean;
```

Description
The ShowShadow property determines whether the FormHints should display a shadow (dots that imitate a shadow) behind a balloon window. You may also specify the color for shadow in ColorShadow property.

See also

© 1998-2006, UtilMind Solutions®

ColorShadow and ShowArrow properties.

9.22.2.13TextAlignment

Applies to
acFormHints component.

Declaration
```
type
  TacTextAlignment = (taNone, taLeft, taRight, taCenter);

property TextAlignment: TacTextAlignment;
```

Description
The TextAlignment property controls how the text should be aligned within borders of balloon hint. When TextAlignment is taCenter, text will centered within a balloon. When it taRight, text will pressed to right edge of balloon.

See also
Font and MaxWidth properties.

9.22.2.14TransparencyRate

Applies to
acFormHints component.

Declaration
```
type
  TacTransparencyRate = 0..100;

property TransparencyRate: TacTransparencyRate; // 0%..100%
```

Description
The TransparencyRate property controls the degree of transparency for background of balloon hints. This property works if only Transparency is True. 100% transparet hints don't show the background at all.

See also
Transparent property.

9.22.2.15Transparent

Applies to
acFormHints component.

Declaration
```
property Transparent: Boolean;
```

Description
The Transparent property controls whether the background of balloon hints should be transparent. The TransparencyRate property controls the degree of transparency.

Transparency only take effect on balloon's background. Text, border and shadow of balloon hint will displayed without transparency.

See also
TransparencyRate and ColorBackground properties.

© 1998-2006, UtilMind Solutions®

9.22.3 Events

9.22.3.1 OnShowHint

Applies to

acFormHints component.

Declaration

```
type
  TacOnShowHint = procedure(Sender: TObject; HintControl: TControl;
Hint: String; var AllowShowHint, ShowStandardHint: Boolean) of object;

property OnShowHint: TacOnShowHint;
```

Description

The OnShowHint event occurs when application is about to show the hint. You can write this event handler if you would like to disallow hints for some controls at run-time (see *AllowShowHint* parameter). The control for which we should display the hint passed with *HintControl* parameter.

new! Now you can select the style of hint dynamically using *ShowStandardHint* parameter. Following example demonstrates how to select cartoon-style hints only for TImages and standard style for other controls.

Example

```
procedure TForm1.acFormHints1ShowHint(Sender: TObject;
  HintControl: TControl; Hint: String; var AllowShowHint,
  ShowStandardHint: Boolean);
begin
  ShowStandardHint := not (HintControl is TImage);
end;
```

9.23 acFormHook

9.23.1 TacFormHook

Overview

The acFormHook component hooks all windows messages (WM_*xxx*) that passes to the Delphi/BCB form and allow their processing and handling in OnMessageBefore and OnMessageAfter events.

The acFormHook can also allow your form to accept files and folders by drag'n'drop technology (AcceptFiles property and OnDropFiles event) and notify you on recreation of the window handle (OnFormRecreate event).

Besides message handling, the acFormHook component have a number of useful features such like CloseButton that disables the close button on form's title bar, Topmost that places form on top over other windows, Bottommost which keeps the form under all other windows, NoCaption — removes the title bar from the form, DragByClient — allow users to move your form by client area, and so forth…

new! If you wish to make some specific actions when form just appears on the screen and all form's contents already shown — use OnFormShown event.
new! To be notified when user clicks close [X] button on form's title bar — use OnFormCloseButtonClick event.

© 1998-2006, UtilMind Solutions®

9.23.2 Properties

9.23.2.1 AcceptFiles

Applies to
acFormHook component.

Declaration
```
property AcceptFiles: Boolean;
```

Description
The AcceptFiles property controls whether current form can accept files by drag'n'drop technology. If AcceptFiles is True, any files and folders can be dropped onto this form.

To accept files (get their names) you also must handle OnDropFiles event.

See also
ActivateFormOnAcceptFiles property and OnDropFiles event.

9.23.2.2 ActivateFormOnAcceptFiles

Applies to
acFormHook component.

Declaration
```
property ActivateFormOnAcceptFiles: Boolean;
```

Description
If ActivateFormOnAcceptFiles property is True and user dropped some files or folders to your form, form will automatically activated and focus is shifted from another Windows application to your application.

See also
AcceptFiles property and OnDropFiles event.

9.23.2.3 Bottommost

Applies to
acFormHook component.

Declaration
```
property Bottommost: Boolean;
```

Description
The Bottommost property allows to constantly keep the form under other windows, moving it to the bottom of Z-order.

Set Bottommost to True to constantly keep the form under other windows, or switch it to False to make it normal window where Z-order can be changed by activating or deactivating the window.

You can't set both Topmost and Bottom properties to True. Only one of these properties can be enabled simuntaneously.

See also
Topmost property.

© 1998-2006, UtilMind Solutions®

9.23.2.4 CloseButton

Applies to
acFormHook component.

Declaration
```
type
  TacCloseButton = (cbEnabled, cbDisabled);

property CloseButton: TacCloseButton;
```

Description
The CloseButton property controls whether close button on form taskbar enabled (if CloseButton is `cbEnabled`) or disabled (if CloseButton is `cbDisabled`).

Snapshot *(close button is disabled)*

Note
Even if CloseButton is `cbDisabled`, this will not prevent your form from closing (in example, on Alt+F4, or Ctrl+F4 for MDI childs). If you really would like to prevent closing, you must handle *OnCloseQuery* event in your forms.

9.23.2.5 DragByCaption

Applies to
acFormHook component.

Declaration
```
property DragByCaption: Boolean;
```

Description
The DragByCaption property controls whether your form can be dragged on screen by title bar area. If DragByCaption is False, user will NOT be able to move the form by caption.

See also
DragByClient property.

9.23.2.6 DragByClient

Applies to
acFormHook component.

Declaration
```
property DragByClient: Boolean;
```

Description
The DragByClient property controls whether your form can be moved on screen dby client area. If DragByClient is True, user will be able to move the form dragging it by client area.

See also
DragByCaption property.

© 1998-2006, UtilMind Solutions®

9.23.2.7 NoCaption

Applies to
acFormHook component.

Declaration
```
property NoCaption: Boolean;
```

Description
The NoCaption property controls whether you would like to show the form's title bar (caption, system menu and title buttons). If NoCaption is True, caption will not be displayed. This property works both at design and run-time.

See also
NoPaintBackground property.

9.23.2.8 NoPaintBackground

Applies to
acFormHook component.

Declaration
```
property NoPaintBackground: Boolean;
```

Description
The NoPaintBackground property controls whether you would like to paint the form's background. If NoBackground is True — background will never be painted.

You can specify NoBackground property at design-time, but this will take effect at run-time only.

See also
NoCaption property.

9.23.2.9 Topmost

Applies to
acFormTopmost and acFormHook components.

Declaration
```
property Topmost: Boolean;
```

Description
The Topmost property determines whether your form can always stay on top over other windows. If Topmost is True, system will place your form at the top of the Z order and your will always on top over other windows. If Topmost is False, another windows can be placed over your form.

Note

The Topmost property of the acFormHook component is highly integrated with others components of the AppControls pack.

So, even if you use acFormHook, acFormTopmost, acFormMagnet and acAppBar components *together on one form* and changing the Topmost property of the acFormHook component, the Topmost property of the acFormTopmost and AlwaysOnTop property of the acAppBar component will be changed at run-time accordingly.

If you wish to keep the window UNDER other windows — use Bottommost property.

© 1998-2006, UtilMind Solutions®

See also
Bottommost property;
acFormTopmost, acFormMagnet and acAppBar components.

9.23.3 Events

9.23.3.1 OnCaptionChanged

Applies to
acFormHook component.

Declaration
```
property OnCaptionChanged: TNotifyEvent;
```

Description
The OnCaptionChanged event occurs when the text on the tile bar of form is changed.

Example
```
procedure TForm1.acFormHook1CaptionChanged(Sender: TObject);
begin
  Label1.Caption := Caption; // Form1.Caption
end;
```

9.23.3.2 OnDropFiles

Applies to
acFormHook component.

Declaration
```
type
  TacDropFilesEvent = procedure(Sender: TObject; Files: TStrings; const
Pos: TPoint) of object;

property OnDropFiles: TacDropFilesEvent;
```

Description
The OnDropFiles event occurs when user drop files or folders onto your form using drag'n'drop technology. The list of dropped files passed in the Files procedure parameter and point where files has been dropped — in Pos parameter.

To allow your form to accept files and folders — make sure that AcceptFiles property is True. To forbid accepting the files — make AcceptFiles property False.

See also
AcceptFiles and ActivateFormOnAcceptFiles properties and small example how to handle this event.

9.23.3.3 OnFormCloseButtonClick

Applies to
acFormHook component.

Declaration
```
type
  TacFormCloseButtonClickEvent = procedure(Sender: TObject; var
AllowClose: Boolean) of object;
```

© 1998-2006, UtilMind Solutions®

```
property OnFormCloseButtonClick: TacFormCloseButtonClickEvent;
```

Description

The OnFormCloseButtonClick event occurs when user clicks the close [X] button on the form's title bar.

Use *AllowClose* parameter to allow or prevent closing of the form on clicking the X button. Set it to True to allow close form or to False to disallow.

Example *(demonstrates how to minimize the form to tray icon instead of closing it)*

```
procedure TMainForm.acFormHook1FormCloseButtonClick(Sender: TObject;
  var AllowClose: Boolean);
begin
  AllowClose := False;
  acTrayIcon1.MinimizeToTray;
end;
```

9.23.3.4 OnFormRecreate

Applies to

acFormHook component.

Declaration

```
property OnFormRecreate: TNotifyEvent;
```

Description

The OnFormRecreate event occurs on recreation of form's window handle (for example, after changing of the BorderStyle or FormStyle property values).

If your form style or extended style was forcedly modified by your program using SetWindowLong API (see Win32 Programmer's Reference for more information about SetWindowLong routine), or you have hooked the form's WndProc, then after form recreation all style changes will be overwritten by default form settings.

Write the OnFormRecreate event handler to restore necessary form styles and hooks after recreation of your form.

9.23.3.5 OnFormShown

Applies to

acFormHook component.

Declaration

```
property OnFormShown: TNotifyEvent;
```

Description

The OnFormShown event occurs at once the form are displayed on the screen and all contents already painted.

Write this event handler if you wish to make some specific actions when the form just appears on the screen with all its contents. Unlike standard OnShow event of the form which occurs before the form actually shown, and unlike OnActivate, which occurs every time when the form is activated-reactivated, this event occurs just once, when everything is painted.

© 1998-2006, UtilMind Solutions®

9.23.3.6 OnMessageAfter

Applies to
acFormHook component.

Declaration
```
type
  TacWndProcMessageAfterEvent = procedure(Sender: TObject; var Message:
TMessage) of object;

property OnMessageAfter: TacWndProcMessageAfterEvent;
```

Description
The OnMessageAfter event occurs after processing of window message by default procedure handler.

Write the OnMessageAfter event handler to hook all windows messages (WM_*xxx*) sent to this form, after their processing by default message handler (CallWindowProc API routine).

Caution
Thousands of messages per second flow though this event. Be careful when coding the handler, because it can affect the performance of the entire application.

See also
OnMessageBefore event and small example how to hook windows messages.

9.23.3.7 OnMessageBefore

Applies to
acFormHook component.

Declaration
```
type
  TacWndProcMessageBeforeEvent = procedure(Sender: TObject; var Message:
TMessage; var Handled: Boolean) of object;

property OnMessageBefore: TacWndProcMessageBeforeEvent;
```

Description
The OnMessageBefore event occurs when acFormHook component hooks the message (WM_*xxx*) for this window, before message passes to the default procedure handler.

Write the OnMessageBefore event handler to hook all windows messages (WM_*xxx*) sent to this form, before their processing by default message handler (CallWindowProc API routine).

If you have processed this message and do not want to pass it to default message handler, set Handled parameter True.

Caution
Thousands of messages per second flow though this event. Be careful when coding the handler, because it can affect the performance of the entire application.

See also
OnMessageAfter event and small example how to hook windows messages.

© 1998-2006, UtilMind Solutions®

9.24 acFormMagnet

9.24.1 TacFormMagnet

Overview

The acFormMagnet allows to magnetize your forms to an edges of screen / work area, borders of parent window and to other forms. Forms can be attached each to others, so when user moves the form, others, "glued" forms will be moved too. With acFormMagnet, your forms can looks and feels like windows of famous WinAmp player.

With acFormMagnet forms obtains following abilities:

1. to be attracted to edges of screen or work area (considering placement of task / appbars)
2. to be attracted to borders of parent form (for child windows).
3. to attract other forms (with acFormMagnet on them) to itself.
4. to glue other forms to itself (when user move form, others, "glued" forms will be moved too)
5. to move forms, dragging by client area.

Key properties

Active	activity flag. Means that this window can be magnetized to other forms and to screen edges;
FormDragable	allows to move forms, dragging by client area;
FormTopmost	controls Z-order of form and allow the form to stay on top over other windows;
Glue	enables or disables gluing feature. When Glue is True and user moves the form, others, attached forms will be moved too;
Options	specifies the behavior of the magnet (type of magnetization and gluing);
Power	maximum range (in screen pixels) to other object that attract the form;
ScreenMagnet	list of properties that allow the form to be magnetized to the screen edges.

events

OnDragMove	occurs when form changes the screen position;
OnGlue	occurs when form is about to be glued to other magnet;
OnMagnet	occurs when form is about to be magnetized to other magnet.

How to use ?

Just drop component onto your form, compile the program and try to move form near the screen edges. See description of properties for customization.

9.24.2 Properties

9.24.2.1 Active

Applies to

acFormMagnet, acFormBackground, acFormHelp, acFormSystemMenu, acAwayTimer, acCursorLocker and acIESniffer components.

Declaration

```
property Active: Boolean;
```

Description

The Active property controls whether the acFormMagnet component and magneting and gluing features is currently active. If Active is True, the FormMagnet will be able to magnetize your form to other windows (with acFormMagnet component on them) and to the edges of screen / workarea.

See also

© 1998-2006, UtilMind Solutions®

Glue, Power and ScreenMagnet properties.

9.24.2.2 FormDragable

Applies to
acFormMagnet component.

Declaration
```
property FormDragable: Boolean;
```

Description
The FormDragable property controls whether the form can be moved over screen by client area. If FormDragable is True, user will be able to move the form dragging it by client area.

See also
DragByClient and DragByCaption properties of acFormHook component.

9.24.2.3 FormTopmost

Applies to
acFormMagnet component.

Declaration
```
property FormTopmost: Boolean;
```

Description
The FormTopmost property determines whether the form can always stay on top over other windows. If FormTopmost is True, system will place the form at the top of the Z-order and form will always on top over other windows. If FormTopmost is False, another windows can be placed over the form.

See also
Topmost property of the acFormHook component.

9.24.2.4 Glue

Applies to
acFormMagnet component.

Declaration
```
property Glue: Boolean;
```

Description
The Glue property controls whether the current form is able to glue other forms to itself. If Glue is True, other forms will be glued to current form and when user moves the current form, others, attached forms will be moved too.

See also
Active property.

9.24.2.5 Options

Applies to
acFormMagnet component.

Declaration
```
type
```

© 1998-2006, UtilMind Solutions®

```
  TacMagnetOption = (moDockOnMoving, moDockToInnerBorders, moDockChains,
moGlueChains);
  TacMagnetOptions = set of TacMagnetOption;

property Options: TacMagnetOptions;
```

Description

Most options not yet implemented and coming in next version of the AppControls pack (v**2.3**)

Please check out MagnetType property.

See also

9.24.2.6 MagnetType

Applies to

acFormMagnet component.

Declaration

```
type
  TacMagnetType = (mkOnMoving, mkAfterMoving);

property MagnetType: TacMagnetType;
```

Description

The MagnetType property controls how the acFormMagnet component will magnetize current form to other forms and screen sides.

If MagnetType is `mkOnMoving`, the form will be attracted to others, every time when user moving the form on the screen (during the moving process).

If Magnet is `mkAfterMoving`, the form can be moved freely by whole screen and will be pulled to other forms once user releases the mouse button (after the moving process).

9.24.2.7 Power

Applies to

acFormMagnet component.

Declaration

```
property Power: Byte; // in screen pixels
```

Description

The Power property controls the maximum range (in screen pixels), where the form can be attracted to other object.

For example, if Power is 32, and move the form near other form with FormMagnet, approaching the form on distance of 32 screen pixels, forms will be magnetized each others.

See also

DesktopPower property.

9.24.2.8 ScreenMagnet

Applies to

acFormMagnet component.

© 1998-2006, UtilMind Solutions®

Declaration

```
type
  TacScreenMagnet = class(TPersistent)
  published
    property Area: TacScreenArea;
    property Edges: TacScreenEdges;
    property ScreenPower: Byte;
  end;
```

Description

The ScreenMagnet is the list of properties which allow the forms to be magnetized to the edges of screen or work area.

9.24.2.8.1 Area

Applies to

acFormMagnet component as subproperty of ScreenMagnet.

Declaration

```
type
  TacScreenArea = (saFullScreen, saWorkArea);

property Area: TacScreenArea;
```

Description

The Area property specifies the desktop area which can magnetize the form. If Area is `saFullScreen`, form can be attracted to the edge of whole screen. If Area is `saWorkArea`, form can be attracted only to edges of working area, considering the placements of taskbar and other application desktop toolbars (appbars).

See also

Edges and ScreenPower properties.

9.24.2.8.2 Edges

Applies to

acFormMagnet component as subproperty of the ScreenMagnet structure.

Declaration

```
type
  TacScreenEdges = set of (seLeft, seTop, seRight, seBottom);

property Edges: TacScreenEdges;
```

Description

The Edges property is the set of possible screen (or desktop) edges. You can specify the screen to allow or disallow your form to be attached to certain edge of screen (or desktop, depending on value of Area property).

Values	Meaning
seLeft	allows your form to be magnetized (attached) by the *left* edge of screen;
seTop	allows attaching to the *top* edge of screen, when True;
seRight	allows attaching to the *right* edge of screen, when True;
seBottom	allows attaching to the *bottom* edge of screen, when True;

See also

© 1998-2006, UtilMind Solutions®

Area and ScreenPower properties.

9.24.2.8.3 ScreenPower

Applies to
acFormMagnet component as subproperty of ScreenMagnet.

Declaration
```
property ScreenPower: Byte;
```

Description
The ScreenPower property controls the maximum range (in screen pixels), where the form can be attracted to the edge of screen / work area or border of parent form (for MDI childs).

For example, if ScreenPower is 32, and user moves the form near the edge of screen / work area, approaching the form on distance of 32 screen pixels, the form will be magnetized to certain edge of screen (or to the edge of work area, depending on value of Area property).

See also
Power and Area properties.

9.24.3 Methods

9.24.3.1 DoMagnet

Applies to
acFormMagnet component.

Declaration
```
procedure DoMagnet;
```

Description
Call DoMagnet method to pull the current form to other forms (with acFormMagnet component on them), or to the edges of screen / work area, or borders of parent form (for MDI childs).

9.24.4 Events

9.24.4.1 OnDragMove

Applies to
acFormMagnet component.

Declaration
```
property OnDragMove: TNotifyEvent;
```

Description
The OnDragMove event occurs when the form changes the screen position.

9.24.4.2 OnGlue

Applies to
acFormMagnet component.

Declaration
```
type
  TOnGlue = procedure(Sender: TObject; AnotherForm: TForm;
              var AllowGlue: Boolean) of object;
```

© 1998-2006, UtilMind Solutions®

```
property OnGlue: TOnGlue;
```

Description
The OnGlue event occurs when the form is about to be glued to other form. You may allow or disallow the gluing, using `AllowGlue` variable.

See also
OnMagnet event.

9.24.4.3 OnMagnet

Applies to
acFormMagnet component.

Declaration

```
type
  TOnMagnet = procedure(Sender: TObject; AnotherForm: TForm;
                var AllowMagnet: Boolean) of object;

property OnMagnet: TOnMagnet;
```

Description
The OnMagnet event occurs when the form is about to be attracted to other form. You may allow or disallow magnetization, using `AllowMagnet` variable.

See also
OnGlue event.

9.25 acFormPlacementSaver

9.25.1 TacFormPlacementSaver

Overview
The acFormPlacementSaver component saves and restores the form placement and Z-order from system registry.

How to use ?
Just drop this component onto your form and component will remember form's placement on each new program restart

To enable or disable the placement saver's features - use Enabled property. The location in the system registry can be specified by RegKey and RegLocation properties. If you don't want to save form size (for example, you are using non-resizeable window) — make RestoreFormSize property False. To save form placement "manually" — use Save method.

If you would like to load or store some additional settings when the acFormPlacementSaver stores and loads the information — write the OnLoadSettings and OnSaveSettings event handlers.

You can use the acFormPlacementSaver together with previously configured **Position** property of your form.

When the acFormPlacementSaver can not find the placement information in the registry (in example, on first program startup), it will use pre-configured placement, specified in the Form.Position property. If the placement information has been found on startup, acFormPlacementSaver will use position saved to the registry.

© 1998-2006, UtilMind Solutions®

9.25.2 Properties

9.25.2.1 Enabled

Applies to
acFormPlacementSaver, acFormSizeRestrictions, acCaptionButton, acFormBackground components.

Declaration
```
property Enabled: Boolean;
```

Description
The Enabled property controls whether the acFormPlacementSaver component can save and restore the form's placement and Z-order. Make Enabled True if you would like to remember and restore the form's placement on program restart.

See also
Load and Save methods.

9.25.2.2 FlickerFreeMaximize

Applies to
acFormPlacementSaver components.

Declaration
```
property FlickerFreeMaximize: Boolean; // False by default
```

Description
The FlickerFreeMaximize property determines whether you want to maximize your form (when it restores from registry on program startup) without flickering on the screen, when the application realigns all controls.

Set the FlickerFreeMaximize to True, if you wish to re-align all controls on the form *before* the application displays the forms on the screen. However, it takes little bit more time to display the form, since it recreate the form and re-aligns the controls in background (but you may also use this time to display the splash screen on program startup).

9.25.2.3 MaximizeIfNoInfoFound

Applies to
acFormPlacementSaver components.

Declaration
```
property MaximizeIfNoInfoFound: Boolean; // False by default
```

Description
The MaximizeIfNoInfoFound property can be used to maximize the application window on its first start (when the placement information is not yet exists in the registry).

Set MaximizeIfNoInfoFound to True, if you want your form to be maximized by default.

9.25.2.4 Options

Applies to
acFormPlacementSaver component and acAppBar component as subproperty of RegistrySaver.

Declaration

© 1998-2006, UtilMind Solutions®

```
type
  TacFormPlacementOptions = set of (saveOnClose,
                                    saveOnMoveResize);

property Options: TacFormPlacementOptions;
```

Description

The Options property determinates when the acFormPlacementSaver component should save current form placement and Z-order to the System Registry.

The possible values of Options are

Value	Meaning
saveOnClose	saves the form's placement when user close the form.
saveOnMoveResize	saves the form's placement when user move or resize the form.

If options above is specified, acFormPlacementSaver will save form placement automatically. If you would like to save placement "manually" — call Save method.

See also

Load adn Save methods.

9.25.2.5 RegKey

Applies to

acAppStartCounter, acFormPlacementSaver components and acAppBar component as subproperty of RegistrySaver property.

Declaration

```
property RegKey: String;
```

Description

The RegKey property defines the path to the key in the system registry where you would like to store the information about current form placement. All stored values will be updated every time when you move or close the form (dependently of Options).

Default property value is `'\Software\Default'`. You should point this property to location where stored the configuration settings of your own application. In example - `'\Software\UtilMind Solutions\The Cool Program\Settings'`.

See also

RegLocation property.

9.25.2.6 RegLocation

Applies to

acAppStartCounter, acFormPlacementSaver components and acAppBar component as subproperty of RegistrySaver property.

Declaration

```
property RegLocation: TacRegLocation;
```

Description

The RegLocation specifies the root key of the hierarchy of subkeys, where you would like to store the information about current form placement. RegLocation property can point to HKEY_CURRENT_USER (rlCurrentUser) or HKEY_LOCAL_MACHINE (rlLocalMachine) root keys.

© 1998-2006, UtilMind Solutions®

When RegLocation is `rlCurrentUser`, acFormPlacementSaver stores settings in the registry section for current logged user. When RegLocation is `rlLocalMachine` settings will be saved for ALL users of current machine.

See also
RegKey property.

9.25.2.7 RestoreFormSize

Applies to
acFormPlacementSaver components.

Declaration
```
property RestoreFormSize: Boolean;
```

Description
The RestoreFormSize property controls whether you would like to save / restore size of your form. If you're using non-resizeable form with bsDialog or bsSingle form style, you may would not to save form's size — make this property False.

For example — you are designing dialog box and save / restore its placement using acFormPlacementSaver component. Form's size can be changed at design-time, however, if RestoreFormSize is True, previous size will be taken from system registry anyway. Set this property to False to don't allow component restore size.

9.25.3 Methods

9.25.3.1 Load

Applies to
acFormPlacementSaver component.

Declaration
```
procedure Load;
```

Description
Restores previous form placement (position, size, state) from the system registry. Registry location should be specified by RegKey and RegLocation properties.

This method executes automatically after creation of the form, if Enabled property is True.

See also
Enabled property and Save method.

9.25.3.2 Save

Applies to
acFormPlacementSaver component.

Declaration
```
procedure Save;
```

Description
Saves current form placement (position, size, state) to the system registry. Registry location should be specified by RegKey and RegLocation properties. Saved settings will used to restore form placement on next creation of this form.

© 1998-2006, UtilMind Solutions®

This method executes automatically when the form is about to be destroyed, or after every movement (as specified in Options property), if Enabled property is True.

Because of high integration with others AppControls components, the Save method saves even Z order of the form (see Topmost property of the acFormHook component, Topmost property of acFormTopmost and AlwaysOnTop property of the acAppBar for more information).

See also
Enabled and Options properties;
Load method.

9.25.4 Events

9.25.4.1 OnLoadSettings

Applies to
acFormPlacementSaver component.

Declaration
```
type
  TacFormPlacementSaverEvent = procedure(Sender: TObject; Reg:
TRegistry) of object;

property OnLoadSettings: TacFormPlacementSaverEvent;
```

Description
The OnLoadSettings occurs when the component loads the information about form placement to the system registry. Write OnLoadSettings event handler to load some additional settings together with form placement.

Example
```
procedure TMainForm.PlacementSaver1SaveSettings(Sender: TObject;
  Reg: TRegistry);
var
  Test: Integer;
begin
  Test := Reg.ReadInteger('Test');
end;
```

See also
OnRegKeyNotFound and OnSaveSettings events.

9.25.4.2 OnRegKeyNotFound

Applies to
acFormPlacementSaver component.

Declaration
```
property OnRegKeyNotFound: TNotifyEvent;
```

Description
The OnRegKeyNotFound event occurs if the form placement information was not found in the registry and the component is unable to restore previous form's placement.

Use this event to specify default values for some settings of your application, if it unable to restore them from registry.

© 1998-2006, UtilMind Solutions®

Example

```
// This example taken from one of ours end-user application. It
specifies default values for some settings,
// makes program executable on Windows startup and puts the form to
right-bottom corner of desktop
uses acUtils; // this is to use GetWorkArea routine

procedure TMainForm.SaverRegKeyNotFound(Sender: TObject);
var
  WorkArea: TRect;
begin
  // set default options
  PlaySound := True;
  PopupWindow := True;
  FlashIcon := True;
  DontResolveHostnames := False;
  LoadActive := True;
  acAppAutoRun1.AutoRun := True;

  // move form to right-bottom corner of desktop
  WorkArea := GetWorkArea;
  SetBounds(WorkArea.Left + WorkArea.Right - Width,
            WorkArea.Top + WorkArea.Bottom - Height,
            Width, Height);
end;
```

See also

OnLoadSettings and OnSaveSettings events.

9.25.4.3 OnSaveSettings

Applies to

acFormPlacementSaver component.

Declaration

```
type
  TacFormPlacementSaverEvent = procedure(Sender: TObject; Reg:
TRegistry) of object;

property OnSaveSettings: TacFormPlacementSaverEvent;
```

Description

The OnSaveSettings occurs when the component stores the information about form placement to the system registry. Write OnSaveSettings event handler to store some additional settings together with form placement.

Example

```
procedure TMainForm.PlacementSaver1SaveSettings(Sender: TObject;
  Reg: TRegistry);
begin
  Reg.WriteInteger('Test', 1);
end;
```

See also

OnLoadSettings and OnRegKeyNotFound events.

© 1998-2006, UtilMind Solutions®

9.26 acFormResizeGrip

9.26.1 TacFormResizeGrip

Overview

The acFormResizeGrip attach the SizeGrip rectangle to the bottom-right corner of the form at run-time. This allows the user to resize the form dragging this grip.

How to use ?

This component should be just placed on the form. If there is nothing aligned so as to cover the bottom-right corner of the form then you will see the standard SizeGrip rectangle on the form at runtime.

Snapshot

Notes

1. The size grip can be visible only when the form's scrollbars is hidden.
2. The size of grip is always same as width and height of the scrollbar and based on the system metric settings.
3. Remember that you can not resize forms with fixed border (i.e: BorderStyle is bsSingle or bsDialog).

9.26.2 Properties

9.26.2.1 Visible

Applies to

acFormResizeGrip, acCaptionButton and acTrayIcon components.

Declaration

```
property Visible: Boolean;
```

Description

The Visible property controls whether a size grip is visible at the bottom-right corner of the form.

9.27 acFormRoller

9.27.1 TacFormRoller

Overview

The TacFormRoller component allows to reduce or expand (roll up / down) your Delphi/BCB forms without minimizing them to taskbar or system tray, giving the impression that the form is shrinking or expanding. Rolling process can be displayed with smooth animation. Since the acFormRoller is successor of acCustomCaptionButton class, component is able to add the custom button on form's title bar and associated menu item to system menu.

Key properties

Animate	whether the rolling should be animated;
AnimateSpeed	determines how fast the form rolls up or down;
CaptionButton	manages the "Roll up / down" button on the form's title bar.
HintRollDown, HintRollUp	the tooltip text for caption button when the form in reduced and normal state;

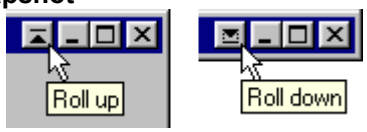
© 1998-2006, UtilMind Solutions®

RemainHeight form height at reduced state;
RollUp determines the current form state (True if reduced and False when form in normal state);
SystemMenu manages the menu item in the system menu that associated with the caption button on the form's title bar.

events

OnRollerChanged occurs when user clicks the caption button or selects the menu item in the system menu associated with this button.

Snapshot



How to change the glyphs for rollup button ?

To change the glyphs for roll up/down buttons you have to modify the ACCB_ROLLER_UP and ACCB_ROLLER_DOWN bitmap resources in the "CaptionButtons.res" file of the AppControls pack. You can change them using standard Image Editor program which supplies with every Delphi/BCB version.

After modifying of these glyphs in the "CaptionButtons.res", they will always with you in any programs, so you will do not need to specify the glyphs every time when you drop component onto form.

See also

Other components which contains a built-in caption button: acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller.

9.27.2 Properties

9.27.2.1 Animate

Applies to

acFormRoller component

Declaration

```
property Animate: Boolean;
```

Description

The Animate property Animate determines whether the form is rolled up and down by steps or all at once. If Animate is True, the form is rolled up and down by steps to give the impression that the form is shrinking or expanding. If Animate is False, the size of the form is simply changed from full size to the height of the caption or the RemainHeight value.

See also

AnimateSpeed property.

9.27.2.2 AnimateSpeed

Applies to

acFormRoller component.

Declaration

```
type
```

© 1998-2006, UtilMind Solutions®

```
  TacFormRollerAnimateSpeed = 0..10;

property AnimateSpeed: TacFormRollerAnimateSpeed; // Default 8
```

Description

AnimateSpeed determines how fast the form rolls up or down. Valid values for AnimateSpeed property are 0 (the slowest) to 10 (the fastest).

Animation speeds are processor speed independent, i.e., an AnimateSpeed setting of 5 will cause the form to change size at the same rate on a 25 MHz 486 and a 500 MHz Pentium III.

See also

Animate property.

9.27.2.3 CaptionButton

Applies to

acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller components.

Declaration

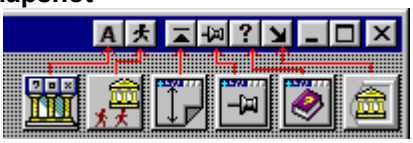
```
property CaptionButton: TacACaptionButton;
```

Description

The CaptionButton is the set of properties for managing an additonal buttons on the form's title bar:

BtnOrder	controls the button placement order on the title bar;
Cursor	specifies the cursor image when button released;
CursorDown	specifies the cursor image when button pressed;
Enabled	enables or disables button;
Hint	specifies the button's tooltip;
SeparatorWidth	width between current and previous buttons;
ShowHint	enables or disables the tooltip (hint);
Visible	shows or hides the button from the title bar.

Snapshot



See also

SystemMenu structure.

9.27.2.3.1 BtnOrder

Applies to

acFormRoller, acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost components as subproperty of CaptionButton.

Declaration

```
property BtnOrder: TacBtnOrder;
```

Description

The BtnOrder property indicates position of caption button, the order in which buttons will be displayed. BtnOrder calculates from right side of title bar to left side (see snapshot below).

Initially, the button order is always the order in which the components were added to the form, but

© 1998-2006, UtilMind Solutions®

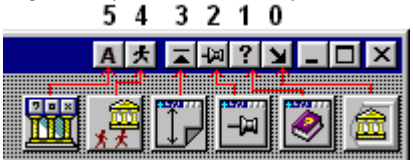
you can change this by changing the BtnOrder property. The value of the BtnOrder property is unique for each ancestor of the acCaptionButton on the form (like TabOrder property of ancestors of TWinControl). The first caption button added to the form has a BtnOrder value of 0, the second is 1, the third is 2, and so on. These values determine where a control is on the title bar.

If you attempt to give a button a BtnOrder value greater than the number of caption buttons on the form minus one (because numbering starts with 0), AppControls won't accept the new value, but will enter the value that assures the component will be the last in the button order.

The caption button with the BtnOrder value of 0 is the custom button which placed on right side of title bar, after regular system buttons.

To separate the buttons by additional empty space — use SeparatorWidth property. For example snapshot below shows that acAppAutoRun component have BtnOrder = 4 (fifth position from the left of system buttons) and SeparatorWidth = 4 pixels.

Snapshot *(buttons orders):*



See also

SeparatorWidth property and acCaptionButton component

9.27.2.3.2 Cursor

Applies to

acFormRoller, acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost components as subproperty of CaptionButton.

Declaration

```
property Cursor: TCursor;
```

Description

The Cursor property controls the mouse cursor shape used when the mouse moves over CaptionButton in released state.

See also

CursorDown property and acCaptionButton component.

9.27.2.3.3 CursorDown

Applies to

acFormRoller, acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost components as subproperty of CaptionButton.

Declaration

```
property CursorDown: TCursor;
```

Description

The CursorDown property controls the mouse cursor shape used when the mouse moves over CaptionButton in pressed state.

See also

© 1998-2006, UtilMind Solutions®

Cursor property and acCaptionButton component.

9.27.2.3.4 Enabled

Applies to
acFormRoller, acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost components as subproperty of CaptionButton.

Declaration
```
property Enabled: Boolean;
```

Description
The Enabled property controls whether the button on the title bar responds to mouse and keyboard messages. If Enabled is True, button responds normally. If Enabled is False, button become disabled and user can not press that button.

See also
Visible property of CaptionButton and acCaptionButton component.

9.27.2.3.5 Hint

Applies to
acFormRoller, acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost components as subproperty of CaptionButton.

Declaration
```
property Hint: String;
```

Description
The Hint property is the text string that can appear when mouse pointer moves over the caption button.

If the CaptionButton's ShowHint property is False, the Help Hint won't appear, but the other hints still will.

Note
Default Hint string for acFormRoller component is 'Roll up' at normal state and 'Roll down' at reduced window state. Do not use this property to control the tooltip text. For this purpose you can use the HintRollUp and HintRollDown properties!

See also
acCaptionButton component and ShowHint property.

9.27.2.3.6 SeparatorWidth

Applies to
acFormRoller, acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost components as subproperty of CaptionButton.

Declaration
```
property SeparatorWidth: Byte; // in screen pixels
```

Description
The SeparatorWidth property controls the space between current and previous by order caption button in screen pixels.

© 1998-2006, UtilMind Solutions®

See also
BtnOrder property and acCaptionButton component.

9.27.2.3.7 ShowHint

Applies to
acFormRoller, acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost components as subproperty of CaptionButton.

Declaration
```
property ShowHint: Boolean;
```

Description
The ShowHint property controls whether CaptionButton can show the Hint string when mouse pointer moves over button. If ShowHint is True, Hint will appears.

See also
acCaptionButton component and Hint property.

9.27.2.3.8 Visible

Applies to
acFormRoller, acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost components as subproperty of CaptionButton.

Declaration
```
property Visible: Boolean;
```

Description
The Visible property determines whether the caption button currently visible on the title bar. Make Visible property True if you would like to show the button on title bar, or False if you would like to hide the button.

When you changing the Visible property, placements of others button will automacially recalculated and other buttons with higher BtnOrder will be shifted to the right side of the title bar, filling the empty space.

See also
Enabled property of CaptionButton and acCaptionButton component.

9.27.2.4 HintRollDown

Applies to
acFormRoller component.

Declaration
```
property HintRollDown: String; // Default is 'Roll down'
```

Description
The HintRollDown property controls the hint (tooltip text) for the button on the form's title bar and the caption of the menu item in the system menu, when the form is currently in reduced state (RollUp is True).

See also
HintRollUp property.

© 1998-2006, UtilMind Solutions®

9.27.2.5 HintRollUp

Applies to
acFormRoller component.

Declaration
```
property HintRollUp: String; // Default is 'Roll up'
```

Description
The HintRollDown property controls the hint (tooltip text) for the button on the form's title bar and the caption of the menu item in the system menu, when the form is currently in normal state (RollUp is False).

See also
HintRollDown property.

9.27.2.6 RemainHeight

Applies to
acFormRoller component.

Declaration
```
property RemainHeight: Word;
```

Description
The RemainHeight property specifies the height of form's client area that should be visible even if form is reduced (rolled up). Default value is 0, that means the client area of form is completely hidden except for the caption bar. Using larger values for RemainHeight allows part of the form to remain visible.

See also
RollUp property, OnRollerChanged event.

9.27.2.7 RollUp

Applies to
acFormRoller component.

Declaration
```
property RollUp: Boolean;
```

Description
The RollUp property determines whether the form is rolled up or down and the form's height is reduced (RollUp is True) or expanded to it's normal state (RollUp is False). Change this property at run-time to reduce or expand the window.

When you change this property at run-time, button and menu item in the system menu will be changed accordingly — when RollUp is False and when RollDown is True.

User can not resize the form when it's height is reduced (rolled up).

See also
RemainHeight property, OnRollerChanged event.

© 1998-2006, UtilMind Solutions®

9.27.2.8 SystemMenu

Applies to

acCaptionButton component and other components that uses TacCustomCaptionButton as ancestor class - acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller.

Declaration

```
property SystemMenu: TacSystemMenu;
```

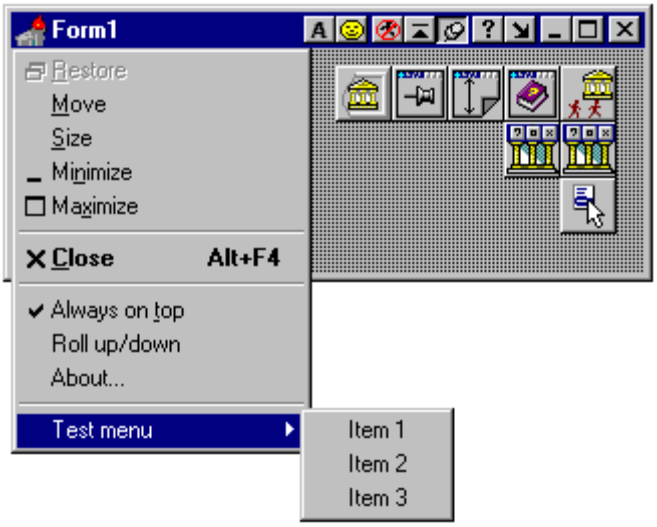
Description

The System Menu is the popup menu that appears when you click on the program icon on the upper-left corner of form's title bar. The SystemMenu property is the list of properties that manages the menu item associated with current button on the form's title bar.

Properties

ApplyToMenu	applies or removes the menu item associated with current button from the system menu;
Caption	text for the title of menu item;
Position	position of current menu item in the system menu;
Separators	specifies the separators for menu item, to separatate it from previous and / or next menu item of the system menu.

Snapshot



See also

CaptionButton structure;
acFormSystemMenu component.

9.27.2.8.1 ApplyToMenu

Applies to

acCaptionButton component and all other components which uses CaptionButton as base ancestor class - acAppAutoRun, acTrayIcon, acFormTopmost, acFormRoller. ApplyToMenu is subproperty of SystemMenu.

Declaration

```
property ApplyToMenu: Boolean;
```

© 1998-2006, UtilMind Solutions®

Description

The ApplyToMenu property controls whether the CaptionButton currently have the menu item assiciated with this button. Set ApplyToMenu property to True to add according menu item (with text specified by Caption property, in position specified by Position property) to the form's system menu and False to remove.

See also

Caption property.

9.27.2.8.2 Caption

Applies to

acCaptionButton component and all other components which uses CaptionButton as base ancestor class - acAppAutoRun, acTrayIcon, acFormTopmost, acFormRoller. Caption is subproperty of SystemMenu.

Declaration

```
property Caption: String;
```

Description

The Caption property specifies the text for menu item in the form's system menu, associated with current caption button on the title bar. If Caption is not specified, text from Hint property will be taken as item title.

Note

Default caption for menu item in the system menu for acFormRoller component is 'Roll up' at normal state and 'Roll down' at reduced window state. Do not use this property to control the tooltip text. For this purpose you can use the HintRollUp and HintRollDown properties!

See also

Hint property of CaptionButton

9.27.2.8.3 Position

Applies to

acCaptionButton component and all other components which uses CaptionButton as base ancestor class - acAppAutoRun, acTrayIcon, acFormTopmost, acFormRoller. Position is subproperty of SystemMenu.

Declaration

```
property Position: Word;
```

Description

The Position property determines the position for menu item in the system menu associated with current caption button on the form's title bar. Change the Position value to move current menu item by system menu.

Maximum value for Position property is the current maximum number of items in the system menu (since orders for menu items starts with 0).

See also

Separators property.

© 1998-2006, UtilMind Solutions®

9.27.2.8.4 Separators

Applies to
acCaptionButton component and all other components which uses CaptionButton as base ancestor class - acAppAutoRun, acTrayIcon, acFormTopmost, acFormRoller. Separators is subproperty of SystemMenu.

Declaration
```
type
  TacMenuSeparators = set of (seBefore, seAfter);

property Separators: TacMenuSeparators;
```

Description
The Separators property specifies the separators for current menu item to separate it from previous and / or next menu items of the system menu. To set separator before menu item — set `seBefore` to True. To set separator after menu item — set `seAfter` to True.

Specify Separators to separate current menu item in the system menu from another menu items.

See also
Position property.

9.27.3 Events

9.27.3.1 OnRollerChanged

Applies to
acFormRoller component.

Declaration
```
type
  TacFormRollerChangedEvent = procedure(Sender: TObject; RollUp:
Boolean) of object;

property OnRollerChanged: TacFormRollerChangedEvent;
```

Description
The OnRollerChanged event occurs when user click the button on the form's title bar or selects the menu item in the system menu, associated with the caption button.

Parameters
RollUp determines current form / button state. True if form's height is reduced (rolled up) and False when form in normal state.

See also
RollUp, RemainHeight properties.

9.28 acFormSizeRestrictions

9.28.1 TacFormSizeRestrictions

Overview
The acFormSizeRestrictions component brings the restrictions to the size of form and controls minimum and maximum tracking size of your windows.

© 1998-2006, UtilMind Solutions®

To enable or disable size restrictions — use Enable property. To specify the maximum and minumum form size — specify MaxWidth, MaxHeight, MinWidth and MinHeight properties.

new! To specify minimum/maximum size of CLIENT area of your form — use MaxClientWidth, MaxClientHeight, MinClientWidth and MinClientHeight properties. The component automatically calculate the difference between real form size and the size of client area and allows to restrictist the form size considering the visible area of form (client area).

9.28.2 Properties

9.28.2.1 Enabled

Applies to
acFormSizeRestrictions, acFormPlacementSaver, acCaptionButton, acFormBackground components.

Declaration
```
property Enabled: Boolean;
```

Description
The Enabled property controls whether do you want to set restrictions on form size. If Enabled is True, the minimum window size will be values specified in MinWidth and MinHeight property and maximum size - values in MaxWidth and MaxHeight properties.

9.28.2.2 MaxClientHeight

Applies to
acFormSizeRestrictions component.

Declaration
```
property MaxClientHeight: Word;
```

Description
The MaxClientHeight property controls the maximum height of client area your form (visible working area without borders and caption bar).

If you would not to specify restriction for height, set this value to 0.

See also
MaxWidth, MaxHeight, MinWidth, MinHeight + MaxClientWidth, MaxClientHeight, MinClientWidth, MinClientHeight properties.

9.28.2.3 MaxClientWidth

Applies to
acFormSizeRestrictions component.

Declaration
```
property MaxClientWidth: Word;
```

Description
The MaxClientWidth property controls the maximum width of client area your form (visible working area without borders and caption bar).

If you would not to specify restriction for width, set this value to 0.

© 1998-2006, UtilMind Solutions®

See also
MaxWidth, MaxHeight, MinWidth, MinHeight +
MaxClientWidth, MaxClientHeight, MinClientWidth, MinClientHeight properties.

9.28.2.4 MinClientHeight

Applies to
acFormSizeRestrictions component.

Declaration
```
property MaxClientHeight: Word;
```

Description
The MaxClientHeight property specifies the minimum height of client area your form (visible working area without borders and caption bar).

If you would not to specify restriction for height, set this value to 0.

See also
MaxWidth, MaxHeight, MinWidth, MinHeight +
MaxClientWidth, MaxClientHeight, MinClientWidth, MinClientHeight properties.

9.28.2.5 MinClientWidth

Applies to
acFormSizeRestrictions component.

Declaration
```
property MaxClientHeight: Word;
```

Description
The MaxClientHeight property specifies the minimum width of client area your form (visible working area without borders and caption bar).

If you would not to specify restriction for width, set this value to 0.

See also
MaxWidth, MaxHeight, MinWidth, MinHeight +
MaxClientWidth, MaxClientHeight, MinClientWidth, MinClientHeight properties.

9.28.2.6 MaxHeight

Applies to
acFormSizeRestrictions component and FloatRestrictions structure of acAppBar component.

Declaration
```
property MaxHeight: Word;
```

Description
The MaxHeight property controls the maximum height of your form. If size restrictions is Enabled and MaxHeight value is not 0, user will not able to make vertical resize of your window with height value greater than specified in MaxHeight property.

If you would not to specify restriction for maximum form height, leave this value 0.

See also

© 1998-2006, UtilMind Solutions®

MaxWidth, MaxHeight, MinWidth, MinHeight +
MaxClientWidth, MaxClientHeight, MinClientWidth, MinClientHeight properties.

9.28.2.7 MaxWidth

Applies to
acFormSizeRestrictions component and FloatRestrictions structure of acAppBar component.

Declaration
```
property MaxWidth: Word;
```

Description
The MaxWidth property controls the maximum width of your form. If size restrictions is Enabled and MaxWidth value is not 0, user will not able to resize the form horizontally to make the form width greater than specified in MaxWidth property.

If you would not to specify restriction for maximum form width, leave this value 0.

See also
MaxWidth, MaxHeight, MinWidth, MinHeight +
MaxClientWidth, MaxClientHeight, MinClientWidth, MinClientHeight properties.

9.28.2.8 MinHeight

Applies to
acFormSizeRestrictions component and FloatRestrictions structure of acAppBar component.

Declaration
```
property MinHeight: Word;
```

Description
The MinHeight property controls the minimum height of your form. If size restrictions is Enabled and MinHeight value is not 0, user will not able to resize the form horizontally to make the form height less than specified in MinHeight property.

If you would not to specify restriction for minimum form height, set this value to 0.

See also
MaxWidth, MaxHeight, MinWidth, MinHeight +
MaxClientWidth, MaxClientHeight, MinClientWidth, MinClientHeight properties.

9.28.2.9 MinWidth

Applies to
acFormSizeRestrictions component and FloatRestrictions structure of acAppBar component.

Declaration
```
property MinWidth: Word;
```

Description
The MinWidth property controls the minimum width of your form. If size restrictions is Enabled and MinWidth value is not 0, user will not able to resize the form horizontally to make the form width less than specified in MinWidth property.

If you would not to specify restriction for minimum form width, set this value to 0.

© 1998-2006, UtilMind Solutions®

See also
MaxWidth, MaxHeight, MinWidth, MinHeight +
MaxClientWidth, MaxClientHeight, MinClientWidth, MinClientHeight properties.

9.29 acFormSystemMenu

9.29.1 TacFormSystemMenu

Overview
The acFormSystemMenu component encapsulates the properties and methods of the window menu (also known as System menu or Control menu) and allows to work with system menu of your form as with usual TPopupMenu component. You can disable and delete regular menu items, or insert custom menu items to the existing system menu.

Disabling of regular system menu items will change appearance of standard caption buttons on the title bar. You can disable X close button, maximize / minimize and menu items

properties

Active	activity flag. Controls whether system menu should contain custom menu items (if True) or only default items (if False);
InsertPosition	specifies the position where to insert custom menu items, specified by Items property;
Items	contains custom menu items which will be inserted to the system menu in position specified by InsertPosition property. In design-time you can edit custom items like standard PopupMenu;
SysItems	contains list of default (system) items of the system menu. You can enable, disable or hide them from the menu;
SystemMenuHandle	returns handle of the system menu.

9.29.2 Properties

9.29.2.1 Active

Applies to
acFormSystemMenu, acFormBackground, acFormHelp, acFormMagnet, acAwayTimer, acCursorLocker and acIESniffer components.

Declaration
```
property Active: Boolean;
```

Description
The Active property is the activity flag of the acFormSystemMenu component. It controls whether the system menu of current form should be changed accordingly current settings and whether the system menu should contain custom menu items, specified in Items property.

Set Active to False to restore default system menu.

See also
Items and InsertPosition properties.

9.29.2.2 InsertPosition

Applies to
acFormSystemMenu component.

Declaration
```
property InsertPosition: Word;
```

© 1998-2006, UtilMind Solutions®

Description

The InsertPosition property specifies the zero-based relative position, where to insert additional elements of system menu, described in the Items property, between the regular menu elements.

Set the InsertPosition to 0 to make the insertion of the Items *before* regular elements of the system menu, and increase this value to shift insertion down relatively to the regular elements.

Snapshots

InsertPosition = 0

InsertPosition = 5

See also

Items property.

9.29.2.3 Items

Applies to

acFormSystemMenu component.

Declaration

```
property Items: TMenuItems;
```

Description

Specifies the additional elements for the system menu. Use InsertPosition property to specify the position where to insert described items, between regular elements of the system menu.

Items is a single TMenuItem object that describes the elements of the menu in its own Items property. Because Items is the default property of TMenuItem, the Items property of TacFormSystemMenu can be treated as an indexed array of menu items that describe the individual items in the menu. That is, instead of writing

```
FirstItem := acFormSystemMenu1.Items.Items[0];
```

use

```
FirstItem := acFormSystemMenu1.Items[0];
```

Set the Items property at design time by clicking on the Items property in the Object Inspector. The Menu Designer automatically inserts a menu item. As each menu item is entered, outlined regions appear at the possible locations of additional menu items. Click in those regions to specify additional menu items. At runtime, create a TMenuItem that describes the elements of the menu programmatically, and set the Items property of the men

See also

InsertPosition and SysItems properties.

9.29.2.4 SysItems

Applies to

acFormSystemMenu component.

© 1998-2006, UtilMind Solutions®

Declaration

```
type
  TacSystemMenuItemState = (isActive, isHidden, isDisabled);
  TacFormSystemMenuItems = class
  published
    property Restore: TacSystemMenuItemState;
    property Move: TacSystemMenuItemState;
    property Size: TacSystemMenuItemState;
    property Minimize: TacSystemMenuItemState;
    property Maximize: TacSystemMenuItemState;
    property Close: TacSystemMenuItemState;
    property MDINext: TacSystemMenuItemState;
  end;

property SysItems: TacFormSystemMenuItems;
```

Description

The SysItems structure allows to modify appearance of the regular elements of the system menu and standard caption buttons on the form title bar, associated with these elements.

Properties	Meaning
Restore	;
Move	;
Size	;
Minimize	;
Maximize	;
Close	;
MDINext	.

See also

Items and InsertPosition properties.

9.29.2.5 SystemMenuHandle

Applies to

acFormSystemMenu component + all successors of TacFormHookComponent class.

Declaration

```
property SystemMenuHandle: hMenu; // Read-only !!
```

Description

The SystemMenuHandle determines the handle of the system menu for current form for copying and modifying.

This is read-only property and used internally by acFormSystemMenu component. See also GetSystemMenu function in the Win32 API help.

9.30 acFormTopmost

9.30.1 TacFormTopmost

Overview

The acFormTopmost component controls Z-order (Topmost property) of your forms and allows to place your forms over other windows. Since acFormTopmost is successor of acCustomCaptionButton class, it able to add the custom button on form's title bar and associated menu item to system menu. Add "Always on top" feature to your forms without single line

© 1998-2006, UtilMind Solutions®

of code!

Key properties

CaptionButton manages the "Roll up / down" button on the form's title bar.
SystemMenu manages the menu item in the system menu that associated with the caption button on the form's title bar.
Topmost determines the form state. True when form is "Always on top" and False when form in normal state.

events

OnTopmostChanged occurs when user clicks the caption button or selects the menu item in the system menu associated with this button.

Snapshot

How to change the glyphs for "Always On Top" button ?

To change the glyphs for "Always On Top" on buttons you have to modify the ACCB_TOPMOST_UP and ACCB_TOPMOST_DOWN bitmap resources in the "CaptionButtons.res" file of the AppControls pack. You can change them using standard Image Editor program which supplies with every Delphi/BCB version.

After modifying of these glyphs in the "CaptionButtons.res", they will always with you in any programs, so you will do not need to specify the glyphs every time when you drop component onto form.

See also

Other components which contains a built-in caption button: acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller.

9.30.2 Properties

9.30.2.1 CaptionButton

Applies to

acFormTopmost, acAppAutoRun, acTrayIcon, acFormHelp and acFormRoller components.

Declaration

```
property CaptionButton: TacACaptionButton;
```

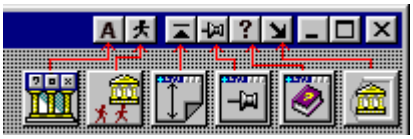
Description

The CaptionButton is the set of properties for managing an additonal buttons on the form's title bar:

BtnOrder controls the button placement order on the title bar;
Cursor specifies the cursor image when button released;
CursorDown specifies the cursor image when button pressed;
Enabled enables or disables button;
Hint specifies the button's tooltip;
SeparatorWidth width between current and previous buttons;
ShowHint enables or disables the tooltip (hint);
Visible shows or hides the button from the title bar.

Snapshot

© 1998-2006, UtilMind Solutions®



See also
SystemMenu structure.

9.30.2.1.1 BtnOrder

Applies to
acFormTopmost, acAppAutoRun, acTrayIcon, acFormHelp, acFormRoller components as subproperty of CaptionButton.

Declaration
```
property BtnOrder: TacBtnOrder;
```

Description
The BtnOrder property indicates position of caption button, the order in which buttons will be displayed. BtnOrder calculates from right side of title bar to left side (see snapshot below).

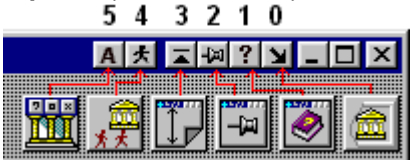
Initially, the button order is always the order in which the components were added to the form, but you can change this by changing the BtnOrder property. The value of the BtnOrder property is unique for each ancestor of the acCaptionButton on the form (like TabOrder property of ancestors of TWinControl). The first caption button added to the form has a BtnOrder value of 0, the second is 1, the third is 2, and so on. These values determine where a control is on the title bar.

If you attempt to give a button a BtnOrder value greater than the number of caption buttons on the form minus one (because numbering starts with 0), AppControls won't accept the new value, but will enter the value that assures the component will be the last in the button order.

The caption button with the BtnOrder value of 0 is the custom button which placed on right side of title bar, after regular system buttons.

To separate the buttons by additional empty space — use SeparatorWidth property. For example snapshot below shows that acAppAutoRun component have BtnOrder = 4 (fifth position from the left of system buttons) and SeparatorWidth = 4 pixels.

Snapshot *(buttons orders):*



See also
SeparatorWidth property and acCaptionButton component.

9.30.2.1.2 Cursor

Applies to
acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller components as subproperty of CaptionButton.

Declaration

```
property Cursor: TCursor;
```

Description
The Cursor property controls the mouse cursor shape used when the mouse moves over CaptionButton in released state.

See also
CursorDown property and acCaptionButton component.

9.30.2.1.3 CursorDown

Applies to
acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller components as subproperty of CaptionButton.

Declaration
```
property CursorDown: TCursor;
```

Description
The CursorDown property controls the mouse cursor shape used when the mouse moves over CaptionButton in pressed state.

See also
Cursor property and acCaptionButton component.

9.30.2.1.4 Enabled

Applies to
acFormTopmost, acFormHelp, acTrayIcon, acAppAutoRun, acFormRoller components as subproperty of CaptionButton.

Declaration
```
property Enabled: Boolean;
```

Description
The Enabled property controls whether the button on the title bar responds to mouse and keyboard messages. If Enabled is True, button responds normally. If Enabled is False, button become disabled and user can not press that button.

See also
Visible property of CaptionButton and acCaptionButton component.

9.30.2.1.5 Hint

Applies to
acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller components as subproperty of CaptionButton.

Declaration
```
property Hint: String;
```

Description
The Hint property is the text string that can appear when mouse pointer moves over the caption button.

If the CaptionButton's ShowHint property is False, the Help Hint won't appear, but the other hints still will.

© 1998-2006, UtilMind Solutions®

Default Hint string for acFormTopmost component is 'Always on top'. You may change this string at run- or design-time if you're writting multilingual program.

See also
acCaptionButton component and ShowHint property.

9.30.2.1.6 SeparatorWidth

Applies to
acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller components as subproperty of CaptionButton.

Declaration
```
property SeparatorWidth: Byte; // in screen pixels
```

Description
The SeparatorWidth property controls the space between current and previous by order caption button in screen pixels.

See also
BtnOrder property and acCaptionButton component.

9.30.2.1.7 ShowHint

Applies to
acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller components as subproperty of CaptionButton.

Declaration
```
property ShowHint: Boolean;
```

Description
The ShowHint property controls whether CaptionButton can show the Hint string when mouse pointer moves over button. If ShowHint is True, Hint will appears.

See also
acCaptionButton component and Hint property.

9.30.2.1.8 Visible

Applies to
acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller components as subproperty of CaptionButton.

Declaration
```
property Visible: Boolean;
```

Description
The Visible property determines whether the caption button currently visible on the title bar. Make Visible property True if you would like to show the button on title bar, or False if you would like to hide the button.

When you changing the Visible property, placements of others button will automacially recalculated and other buttons with higher BtnOrder will be shifted to the right side of the title bar, filling the empty space.

© 1998-2006, UtilMind Solutions®

See also
Enabled property of CaptionButton and acCaptionButton component.

9.30.2.2 MenuItem

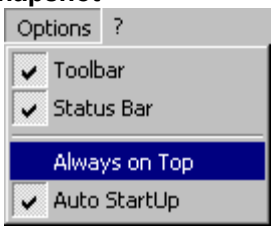
Applies to
acFormTopmost component.

Declaration
```
property MenuItem: TMenuItem;
```

Description
The MenuItem property can be used to associate the menu item of your application, which should reflect current state of Topmost property. When user clicks that menu item, the state of Topmost property will be changed accordingly.

As you can see, not only standard MenuItems can be assigned to this property. You associate to it ANY component which has "Checked" or "Down" property (anything that can reflect the state of boolean value, i.e: non-standard menu item or "speed button").

Snapshot



See also
Topmost property.

9.30.2.3 SystemMenu

Applies to
acCaptionButton component and other components that uses TacCustomCaptionButton as ancestor class - acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller.

Declaration
```
property SystemMenu: TacSystemMenu;
```

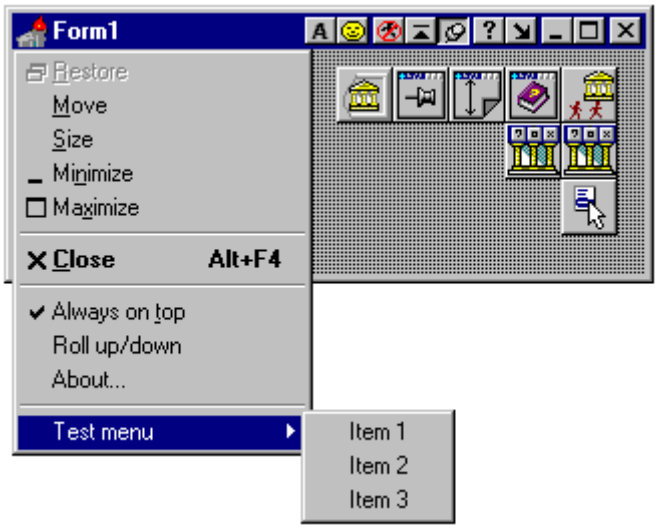
Description
The System Menu is the popup menu that appears when you click on the program icon on the upper-left corner of form's title bar. The SystemMenu property is the list of properties that manages the menu item associated with current button on the form's title bar.

Properties

ApplyToMenu	applies or removes the menu item associated with current button from the system menu;
Caption	text for the title of menu item;
Position	position of current menu item in the system menu;
Separators	specifies the separators for menu item, to separatate it from previous and / or next menu item of the system menu.

Snapshot

© 1998-2006, UtilMind Solutions®



See also
CaptionButton structure;
acFormSystemMenu component.

9.30.2.3.1 ApplyToMenu

Applies to
acCaptionButton component and all other components which uses CaptionButton as base ancestor class - acAppAutoRun, acTrayIcon, acFormTopmost, acFormRoller. ApplyToMenu is subproperty of SystemMenu.

Declaration
```
property ApplyToMenu: Boolean;
```

Description
The ApplyToMenu property controls whether the CaptionButton currently have the menu item assiciated with this button. Set ApplyToMenu property to True to add according menu item (with text specified by Caption property, in position specified by Position property) to the form's system menu and False to remove.

See also
Caption property.

9.30.2.3.2 Caption

Applies to
acCaptionButton component and all other components which uses CaptionButton as base ancestor class - acAppAutoRun, acTrayIcon, acFormTopmost, acFormRoller. Caption is subproperty of SystemMenu.

Declaration
```
property Caption: String;
```

Description
The Caption property specifies the text for menu item in the form's system menu, associated with current caption button on the title bar. If Caption is not specified, text from Hint property will be taken as item title.

© 1998-2006, UtilMind Solutions®

See also
Hint property of CaptionButton.

9.30.2.3.3 Position

Applies to
acCaptionButton component and all other components which uses CaptionButton as base ancestor class - acAppAutoRun, acTrayIcon, acFormTopmost, acFormRoller. Position is subproperty of SystemMenu.

Declaration
```
property Position: Word;
```

Description
The Position property determines the position for menu item in the system menu associated with current caption button on the form's title bar. Change the Position value to move current menu item by system menu.

Maximum value for Position property is the current maximum number of items in the system menu (since orders for menu items starts with 0).

See also
Separators property.

9.30.2.3.4 Separators

Applies to
acCaptionButton component and all other components which uses CaptionButton as base ancestor class - acAppAutoRun, acTrayIcon, acFormTopmost, acFormRoller. Separators is subproperty of SystemMenu.

Declaration
```
type
  TacMenuSeparators = set of (seBefore, seAfter);

property Separators: TacMenuSeparators;
```

Description
The Separators property specifies the separators for current menu item to separate it from previous and / or next menu items of the system menu. To set separator before menu item — set `seBefore` to True. To set separator after menu item — set `seAfter` to True.

Specify Separators to separate current menu item in the system menu from another menu items.

See also
Position property.

9.30.2.4 Topmost

Applies to
acFormTopmost and acFormHook components.

Declaration
```
property Topmost: Boolean;
```

Description
The Topmost property determines whether the form can always stay on top over other windows. If

© 1998-2006, UtilMind Solutions®

Topmost is True, system will place the form at the top of the Z-order and form will always on top over other windows. If Topmost is False, another windows can be placed over the form.

The Topmost property of the acFormTopmost associated with the CaptionButton on the title bar (even if button is invisible) and according menu item in the SystemMenu. When you change the Topmost property, the state of caption button will changed too. When you click the caption button - Topmost property will changed accordingly to button state.

The Topmost property of acFormTopmost component is highly integrated with Topmost property of the acFormHook component, and AlwaysOnTop property of the acAppBar component.

So, even if you use acFormTopmost, acFormHook and acAppBar components *together on one form* and changing the Topmost property of the acFormTopmost component, the Topmost property of the acFormHook and AlwaysOnTop property of the acAppBar component will be changed at run-time accordingly.

See also
OnTopmostChanged event and MenuItem property;
acFormHook and acAppBar components.

9.30.3 Events

9.30.3.1 OnTopmostChanged

Applies to
acFormTopmost component.

Declaration

```
type
  TacFormTopmostChangedEvent = procedure(Sender: TObject; Topmost:
Boolean) of object;

property OnTopmostChanged: TacFormTopmostChangedEvent;
```

Description
The OnTopmostChanged event occurs when Z order of the form changed to topmost or non-topmost. Write the OnTopmostChanged event handler to perform special processing when the form's topmost status is changed. Current topmost status of the form passes to Topmost parameter of the event handler.

See also
Topmost property.

9.31 acHTTP

9.31.1 TacHTTP

Overview
The acHTTP component is extremely easy to use WinInet-based HTTP client component which allows to post and get any data from the Web via HTTP protocol. With acHTTP you can grab Web pages, download files and documents (or only their headers without the content), get results of the CGI programs (for example, results of web-based search engines / databases), or even upload files

© 1998-2006, UtilMind Solutions®

to the Web-based programs.

The acHTTP can grab web contents both in binary and text formats, supports cache of Internet Explorer, can pause and resume broken downloads, read data from password protected directories and automatically supports several proxy authentication schemes (basic, digest, NTLM etc).

Key features

- works with both, HTTP and HTTPS protocols + can read files from local area network using "file://" prefix for URLs;
- can either POST or GET data to remote CGI programs;
- can upload files by HTTP protocol, using multipart/form-data POST method, introduced in RFC 1867;
- can download or upload data to password protected Web directories;
- flexible cache control, provided by WinINet library. The acHTTP just use standard cache of Internet Explorer to retreive or write downloaded files to cache and save bandwidth from re-downloading. The cache control can be is easily customized by various options;
- can pause and resume broken downloads;
- automatically accepts SSL certificates, can allow or disallow access to sites with invalid or expired certificates;
- can automatically retreive proxy information from Internet Options of Control Panel, make HTTP requests with custom proxy settins or directly without any proxy server;
- automatically supports several secure proxy authentication schemes: basic, digest, NTLM (NT Lan Manager), MSN (Microsoft Network), DPA (Distributed Password Authentication) and RPA (Remote Passphrase Authentication by CompuServe)
- using super stable threading mechanism, which allows to execute HTTP requests from both, standard, and ActiveX forms.

How to use ?

Just specify the location of Internet document which you would like to receive to URL property, and call Read method to start downloading. Use OnDone event to handle received data, or use OnHTTPError, OnConnLost and OnHostUnreachable events to handle errors. To show the download progress, write OnProgress event handler.

If you need to send some data to the CGI program via POST method, specify the request in the POSTData property before Reading. Usually acHTTP component can automatically recognize which request method you would like to use, POST or GET, however if you wish you can specify it in the RequestMethod property.

If you want to just check the HTTP headers of remote document *without* downloading it (or *before* downloading) — write the OnHeaderInfo event handler to receive all the headers (document size, content type, language, encoding, last modification and expiration date and so forth). You can also use this event to check the file information and decide whether you really want to download it…

To read files from password protected Web directories you should specify login information in Username and Password properties. Also use OnPasswordRequest event to specify login information dynamically.

acHTTP can *upload* files via HTTP protocol, using *multipart/form-data* POST method, introduced in RFC 1867. Uploading still easy to use as well as downloading. For more details see Upload method and OnUploadFieldRequest, OnUploadProgress events.

acHTTP automatically recognize and supports several Proxy authentication schemes (basic, digest, NTLM (NT Lan Manager), MSN (Microsoft Network), DPA (Distributed Password Authentication) and RPA (Remote Passphrase Authentication by CompuServe)). If user works

© 1998-2006, UtilMind Solutions®

through secure proxy server which requires user authentication — specify ProxyUsername and ProxyPassword properties to the Proxy structure, or write OnProxyAuthenticationRequest event handler to prompt users for the username and password required to access the Web via proxy server.

If you need to terminate downloading process imediately — call Abort method. In case if you downloading some binary data to file (specified in OutputFileName property), you can resume the downloading at any time using Resume method.

The acHTTP automatically uses simple but smart scheme of checking whether the file which you're trying to Resume has been updated or modified. Before downloading of the data which should be appended, it downloads small data chunk (with size specified in TransferBufferSize property), *before the break*, and compares with the same data chunk at the end of file.

In case if compared data are equal — it continue downloading and append downloaded data to the end of local file. Otherwise it assume that file which beging downloaded has been changed, and starts download from beginning.

By default TransferBufferSize = 4Kb, so every time when you call Resume method, the component download 4Kb of extra "rollback" data to check file consistancy.

Also, the acHTTP unit provides another, extremally simpe way to receive some text information from the Web, using HTTPReadString function, without any acHTTP component on the form and without specifying its properties and handling the events. The HTTPReadString can be used to download some text string or HTML document from specified location.

Usage example

In Delphi: http://www.appcontrols.com/demos/httpdemo-delphi.zip
In C++ Builder: http://www.appcontrols.com/demos/httpdemo-bcb.zip
Compiled executable: http://www.appcontrols.com/demos/exe/HTTPDemo.exe

Important note!

C++ Builder programmers: Don't forget to add "INET.LIB" file (or "WININET.LIB" in C++ Builder 6 and higher, this file can be found in "..\CBuilder*X*\Lib" directory) to your project which uses the acHTTP. The INET.LIB contains the prototypes for required Internet routines from WinInet.DLL.

See also

acHTTPDownloadQueue, acThread, acAutoUpgrader, acSendMail and acWebImage components.

9.31.2 Properties

9.31.2.1 AcceptTypes

Applies to

acHTTP component.

Declaration

```
property AcceptTypes: String; // default is "*/*"
```

Description

The AcceptTypes property specifies the array of media types (also known as Multipurpose Internet Mail Extension (MIME) type) which you would like to receive from the Web using the acHTTP component (HTTP client). These strings indicates content types accepted by the client. If AcceptTypes is empty, no types are accepted by the client.

For example, if you would like to get HTML files only, set AcceptTypes property to **"text/html"**. To

© 1998-2006, UtilMind Solutions®

receive flat-text files only, set AccessTypes to "**text/plain**".

Servers interpret a lack of accept types to indicate that the client accepts only documents of type "**text/***" (that is, only text documents, and not pictures or other binary files.

To specify multiple MIME types, sepearate them by comma sign "**,**" (ie: "image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-excel, application/msword, */*").

Some servers always checking the media types accepted by client to determinate the data format prefered by client. For example, lately some servers can returns human readable data both in HTML and WML formats. HTML used for output to standard Web-browsers and WML for output to celluar phones, handheld computers and other WAP devices. To determinate the client type, server uses the HTTP_ACCEPT environment of client:

```
if ($ENV{'HTTP_ACCEPT'} =~ /wml/) {
  print "Location: http://website.com/wap/index.wml\n\n";
}else {
  print "Location: http://website.com/index.html\n\n";
```

When server founds "WML" word between the accepted types, it will redirect the client to certain WAP page. As you can see, sometimes the HTTP output depend on accepted media types.

For more details on Accept header please see the reference at
http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.1

To determinate the MIME type for some certain file type (file extension) — use acExtAssociation component.

Remark

Some servers does not check accepting types at all and can return files with ANY media type.

See also

AddHeaders, Agent, URL and Referer properties;
MIMEType property of acExtAssociation component.

9.31.2.2 AddHeaders

Applies to

acHTTP and acSendMail components.

Declaration

```
property AddHeaders: TStringList;
```

Description

The AddHeaders property specifies any additional HTTP headers that should pass to the server. You can specify ANY optional headers that may be required by server to process request.

For example, if server can return preferable content taking in account the language that user can read:

```
AddHeadeers.Clear;
{ Assume that user user can read Russian, English, German and French
content (sorted by priority). }
AddHeaders.Add('ACCEPT_LANGUAGE: ru,en,de,fr');
```

List of widely used HTTP headers (* sample values marked red):
ACCEPT_CHARSET: iso-8859-1,*,utf-8
ACCEPT_ENCODING: gzip, deflate
ACCEPT_LANGUAGE: en-us,es
CONNECTION: Keep-Alive

© 1998-2006, UtilMind Solutions®

FROM: someone@somewhere.com
IF_MODIFIED_SINCE: Tue, 06 Feb 2001 18:30:50 GMT
RANGE: bytes=0-255
…see also quick reference to HTTP headers at http://www.cs.tut.fi/~jkorpela/http.html, or full reference at http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html

If you would like to test the HTTP headers specified in *your* HTTP client on real server, you can read content from following URL: http://www.appcontrols.com/cgi/test/http_headers.cgi

See also
AcceptTypes, Agent properties.

9.31.2.3 Agent

Applies to
acHTTP and acSendMail components.

Declaration
```
property Agent: String;
```

Description
The Agent property specifies the name of HTTP client.

The "user agent" can be name of your program or program version. For example, user agent of MS Internet Explorer 5.01 which installed to Windows 98 is `Mozilla/4.0 (compatible; MSIE 5.01; Windows 98)`.

See also
AcceptTypes and AddHeaders properties.

9.31.2.4 Busy

Applies to
acHTTP and acSendMail components.

Declaration
```
property Busy: Boolean; // Read-only !!
```

Description
The Busy property determines whether the HTTP component (its thread) are busy when its execute some operations. When Busy property is True, the acHTTP currently requesting/downloading data from the Web, or processing OnDone event handler.

Note
You can NOT download any data when the HTTP is busy, and must wait until component done all operations. The Read method will return False when the component is busy.

See also
Read method; OnDone and OnAnyError events;
Running property of acThread component.

9.31.2.5 CacheOptions

Applies to
acHTTP, acAutoUpgrader and acWebImage components.

© 1998-2006, UtilMind Solutions®

Declaration

```
type
  TacCacheOption    = (coAlwaysReload, coReloadIfNoExpireInformation,
                       coReloadUpdatedObjects, coPragmaNoCache,
                       coNoCacheWrite, coCreateTempFilesIfCantCache,
                       coUseCacheIfNetFail);
  TacCacheOptions = set of TacCacheOption;

property CacheOptions: TacCacheOptions;
```

Description

The CacheOptions property controls the cache options for the acHTTP component and determines how the component should use standard Internet Explorer's cache.

The cache control has following options:

Value	Meaning
coAlwaysReload	Forces a download of the requested file, object, or directory listing from the origin server, not from the cache.;
coReloadIfNoExpireInformation	Forces a reload if there was no Expires time and no LastModified time returned from the server when determining whether to reload the item from the network;
coReloadUpdatedObjects	Reloads HTTP resources if the resource has been modified since the last time it was downloaded;
coPragmaNoCache	Forces the request to be resolved by the origin server, even if a cached copy exists on the proxy;
coNoCacheWrite	Does not add the downloaded entity to the cache. Note: since secure pages won't be cached, this option is always False when downloading document by HTTPS protocol;
coCreateTempFilesIfCantCache	Causes a temporary file to be created if the file cannot be cached;
coUseCacheIfNetFail	Returns the resource from the cache if the network request for the resource fails due if connection with the server has been reset, or the attempt to connect to the server failed.

See also

InternetOptions property;
Read method; OnHeaderInfo event.

9.31.2.6 FileName

Applies to

acHTTP component.

Declaration

```
property FileName: Boolean; // Read-only !!
```

Description

The FileName is optional and read-only property used to exctract the file name from the HTTP address specified in the URL property.

If you would like to specify the target file for downloaded data — use OutputFileName property.

Example

```
acHTTP1.URL := 'http://www.abc.com/download/filename.zip';
```

© 1998-2006, UtilMind Solutions®

```
Result := acHTTP1.FileName;
// Result will be 'filename.zip';
```

See also
URL, HostName and OutputFileName properties.

9.31.2.7 HostName

Applies to
acHTTP component.

Declaration
```
property HostName: Boolean; // Read-only !!
```

Description
The HostName is optional and read-only property used to exctract the host name from the HTTP address specified in the URL property.

Example
```
acHTTP1.URL := 'http://www.abc.com/download/filename.zip';
Result := acHTTP1.HostName;
// Result will be 'www.abc.com';
```

See also
URL and FileName properties.

9.31.2.8 InternetOptions

Applies to
acHTTP, acAutoUpgrader and acWebImage components.

Declaration
```
type
  TacInternetOption = (ioIgnoreCertificateInvalid,
ioIgnoreCertificateDateInvalid,
                       ioIgnoreUnknownCertificateAuthority,
                       ioIgnoreRedirectToHTTP, ioIgnoreRedirectToHTTPS,
                       ioKeepConnection, ioNoAuthentication,
                       ioNoAutoRedirect, ioNoCookies);
  TacInternetOptions := set of TacInternetOption;

property InternetOptions: TacInternetOptions;
```

Description
The InternetOptions property is the set of options used to specify some behaviors of acHTTP component.

The property is set of following options:

Value	Meaning
ioIgnoreCertificateInvalid	Disables checking of SSL/PCT-based certificates that are returned from the server against the host name given in the request. WinINet functions use a simple check against certificates by comparing for matching host names and simple wildcarding rules;
ioIgnoreCertificateDateInvalid	Disables checking of SSL/PCT-based certificates for proper validity dates;

© 1998-2006, UtilMind Solutions®

ioIgnoreUnknownCertificateAuthority	Specifies whether the component should ignore unknown certificate authority problems, if the server's SSL certificate has been "signed", but by unknown or untrusted authority;
ioIgnoreRedirectToHTTP	Disables detection of this special type of redirect. When this flag is used, WinINet functions transparently allow redirects from HTTPS to HTTP URLs;
ioIgnoreRedirectToHTTPS	Disables detection of this special type of redirect. When this flag is used, WinINet functions transparently allow redirects from HTTP to HTTPS URLs;
ioKeepConnection	Uses keep-alive semantics, if available, for the connection. This flag is required for Microsoft Network (MSN), NT LAN Manager (NTLM), and other types of authentication;
ioNoAuthentication	Does not attempt authentication automatically;
ioNoAutoRedirect	Does not automatically handle redirection;
ioNoCookies	Does not automatically add cookie headers to requests, and does not automatically add returned cookies to the cookie database.

See also

CacheOptions property;
Read method;
OnHeaderInfo, OnRedirected and OnProxyAuthenticationRequest events.

9.31.2.9 OutputFileAttributes

Applies to

acHTTP component.

Declaration

```
type
  TacFileAttribute = (atrArchive, atrHidden, atrReadOnly, atrSystem,
atrTemporary, atrOffline);
  TacFileAttributes = set of TacFileAttribute;

  TacOutputFileAttributes = class
  published
    property Complete: TacFileAttributes default [atrArchive];
    property Incomplete: TacFileAttributes default [atrArchive,
atrTemporary];
  end;

property OutputFileAttributes: TacOutputFileAttributes;
```

Description

The OutputFileAttributes property allows to specify the file attributes for file which being downloaded to location specified in OutputFileName property.

There is 2 "sub-properties": *Complete* and *Incomplete*.

Incomplete property specifies flags for incomplete file which still being downloaded (or paused, but still not downloaded to local drive completely).

Complete specifies attributes which being set to the OutputFileName once the download finished and file completely downloaded.

© 1998-2006, UtilMind Solutions®

You can use this property to make difference between complete and incomplete files for other application, or to quickly determinate whether local file has been completely downloaded.

Also, anyway you will be able to rename the downloaded file and change its file attributes (using SetFileAttributes WinAPI), in the OnDone event handler.

Note

`atrTemporary` and `atrOffline` options has no effect in Win95/98/ME. These options are only for NT-family systems. You can set it but they will work only in NT/XP etc.

Example

```
acHTTP1.URL := 'http://www.domain.com/filename.zip';
acHTTP1.OutputFileName := 'c:\filename.zip';
acHTTP1.OutputFileAttributes.Complete := [atrArchive]; // Normal
acHTTP1.OutputFileAttributes.Incomplete := [atrArchive, atrHidden]; //
"hide" incomplete file
acHTTP1.Read;

// and when file will be successfully downloaded you can
// change the file attributes to Normal in the OnDone event handler
procedure TForm1.acHTTP1Done(Sender: TObject; const ContentType: string;
  FileSize: Integer; Stream: TStream);
begin
  // set file attributes to normal
  Windows.SetFileAttributes(PChar(FileSize), FILE_ATTRIBUTE_NORMAL);
end;
```

See also

OutputFileName property;
Read, ReadRange, Resume and Pause methods;
OnOutputFileError and OnDone events.

9.31.2.10 OutputFileName

Applies to

acHTTP component.

Declaration

```
property OutputFileName: String;
```

Description

The OutputFileName property specifies the target filename for downloaded resource. If the OutputFileName specified, the acHTTP uses the TFileStream instead of TMemoryStream as the storage for downloaded data.

Use the OutputFileName to specify the target filename if you need save data to the file AND if you don't want to use memory-stream of OnDone event.

If you would like to hide the temporary (incomplete) file while it downloading — set OutputFileAttributes.Incomplete property to `[atrHidden]`.

In case if the OutputFileName can not be created (e.g. path not exists, or file locked by system etc) — OnOutputFileError event occur.

Note

The downloaded document will not be stored to the "OutputFileName", in case if component

© 1998-2006, UtilMind Solutions®

received erroneous status response code (not 200 - OK and not 206 - Partial Content (if request was initiated by Resume or ReadRange methods)).

In case of HTTP error, the downloaded content will be represented as TMemoryStream in various events of acHTTP.

Example

```
acHTTP1.URL := 'http://www.domain.com/filename.zip';
acHTTP1.OutputFileName := 'c:\filename.zip';
acHTTP.Read;
```

See also

OutputFileAttributes property;
Read, ReadRange, Resume and Pause methods;
OnDone, OnHTTPError and OnOutputFileError events;
HTTP Status Codes.

9.31.2.11Password

Applies to

acHTTP component.

Declaration

```
property Password: String;
```

Description

The Password property specifies the password to access the data in password protected Web directories. You don't need to specify the password if you reading non-protected data.

You can also specify the login information dynamically, when it necessary, in the OnPasswordRequest event handler.

See also

Username property and OnPasswordRequest event.

9.31.2.12POSTData

Applies to

acHTTP component.

Declaration

```
property POSTData: String;
```

Description

The POSTData property specifies any optional data to send with the HTTP request. The optional data can be the resource or information being posted to the server.

The POSTData property is generally used to POST some data to the CGI programs. For example, you would like to emulate submission of some online form. Set the POSTData accordingly to the fields of HTML form. The format is following:

```
FieldName1=Value1&FieldName2=Value2&VieldNameN=ValueN
```

(Separate entries with "&" character.) Then just point the URL property to the CGI script (specified as <form action=URL> tag in HTML format) and call Read method to submit the form.

Example: *(requesting data from the CGI script (at Torry.net) via POST method)*

```
procedure TForm1.ReadBtnClick(Sender: TObject);
```

© 1998-2006, UtilMind Solutions®

```
begin
  acHTTP1.URL := 'http://www.torry.net/quicksearch.php';
  acHTTP1.POSTData := 'String=HTTP&Exact=Yes&Title=No';
  acHTTP1.Read;
end;

procedure TForm1.acHTTP1Done(Sender: TObject; ContentType: string;
FileSize: Integer; Stream: TStream);
begin
  // Receiving output from Stream
end;
```

Remarks

1. When you trying to POST data to the CGI program specifying the POSTData property, make sure that RequestMethod has set to `rmAutoDetect` or `rmPOST`;
2. The acHTTP component unable to post data to the CGI program when user *working offline* (even if user connected to the Internet). In this case, when you call the Read method, the OnHostUnreachable event occurs. Posting to the CGI programs requires active Internet connection.

See also

RequestMethod and URL properties and Read method;
OnDone event.

9.31.2.13Proxy

Applies to

acHTTP, acAutoUpgrader and acWebImage components.

Declaration

```
type
  TacHTTPAccessType = (atPreconfig, atDirect, atUseProxy);
  TacHTTPProxy = class
  published
    property AccessType: TacHTTPAccessType; // atPreconfig by default
    property ProxyServer: String;
    property ProxyPort: Integer;     // 8080 by default
    property ProxyBypass: TStrings; // 127.0.0.1 by default
    property ProxyUsername: String;
    property ProxyPassword: String;
  end;

property Proxy: TacHTTPProxy;
```

Description

The Proxy structure controls the connection type for the acHTTP component and settings for estabilishing connection via proxy. Connection type (AccessType) can be "pre-configured" (acHTTP will use settings from Control Panel), direct, or via specified proxy server.

If proxy requires authentication — write OnProxyAutenticationRequest event handler to prompt and specify user's login information.

See also

OnProxyAuthenticationRequest event.

© 1998-2006, UtilMind Solutions®

9.31.2.13.1 AccessType

Applies to
acHTTP, acAutoUpgrader and acWebImage components as subproperty of Proxy structure.

Declaration
```
type
  TacHTTPAccessType = (atPreconfig, atDirect, atProxy);

property AccessType: TacHTTPAccessType;
```

Description
The AccessType property controls how the acHTTP component should access the remote server to download data. The AccessType can be *direct* or via the *proxy* server. If you'd like to use the access type previously configured in the Control Panel, leave the AccessType = atPreconfig.

Values	Meaning
atPreconfig	retrieves the proxy or direct configuration from the registry (user can configure the access type in the Control Panel);
atDirect	uses the direct connection and resolves all host names locally;
atProxy	access the remote data via the proxy server. Passes all requests to the proxy, unless a proxy bypass list is not empty and the name to be resolved bypasses the proxy. To specify the proxy server and port — use ProxyServer and ProxyPort properties. To configure bypass list — use ProxyBypass property.

See also
ProxyServer, ProxyPort and ProxyBypass properties.

9.31.2.13.2 ProxyBypass

Applies to
acHTTP, acAutoUpgrader and acWebImage components as subproperty of Proxy structure.

Declaration
```
property ProxyBypass: String;
```

Description
The ProxyBypass contains the list of host names or IP addresses, or both, that should not be routed through the proxy. The list can contain wildcards. If the ProxyBypass is empty, the acHTTP reads the bypass list from the registry.

You can use wild cards to match domain and host names or addresses — for example, www.*.com;128.*;240.*;*.mygroup.*;*x* and so on. Use semicolon (;) to separate entries.

See also
ProxyServer and ProxyPort properties.

9.31.2.13.3 ProxyPassword

Applies to
acHTTP, acAutoUpgrader and acWebImage components as subproperty of Proxy structure.

Declaration
```
property ProxyPassword: String;
```

© 1998-2006, UtilMind Solutions®

Description
The ProxyPassword property specifies the password required for connection via proxy server, if it's require secure authentication.

Alternativaly you can write OnProxyAuthenticationRequest event handler and specify proxy username and password dynamically.

Note
If your proxy server uses some "challenge-response" authentication scheme — please make sure that `ioKeepConnection` option of the InternetOptions is set to True. Otherwise, connection with proxy could be dropped.

See also
ProxyUsername and InternetOptions properties;
OnProxyAuthenticationRequest event.

9.31.2.13.4 ProxyPort

Applies to
acHTTP, acAutoUpgrader and acWebImage components as subproperty of Proxy structure.

Declaration
```
property ProxyPort: Integer;
```

Description
The ProxyPort property specifies the port number for the proxy server. The default value for proxy ports is **8080**.

See also
ProxyServer and ProxyBypass properties.

9.31.2.13.5 ProxyServer

Applies to
acHTTP, acAutoUpgrader and acWebImage components as subproperty of Proxy structure.

Declaration
```
property ProxyServer: String;
```

Description
The ProxyServer property specifies the host name of the proxy server to use if the proxy access was specified in the AccessType property.

See also
ProxyPort and ProxyBypass properties.

9.31.2.13.6 ProxyUsername

Applies to
acHTTP, acAutoUpgrader and acWebImage components as subproperty of Proxy structure.

Declaration
```
property ProxyUsername: String;
```

Description
The ProxyUsername property specifies the username required for connection via proxy server, if it's

© 1998-2006, UtilMind Solutions®

require secure authentication.

Alternativaly you can write OnProxyAuthenticationRequest event handler and specify proxy username and password dynamically.

Note

If your proxy server uses some "challenge-response" authentication scheme — please make sure that `ioKeepConnection` option of the InternetOptions is set to True. Otherwise, connection with proxy could be dropped.

See also

ProxyPassword and InternetOptions properties;
OnProxyAuthenticationRequest event.

9.31.2.14Range

Applies to

acHTTP component.

Declaration

```
type
  TacHTTPRange = class(TPersistent)
  published
    property StartRange: Cardinal;
    property EndRange: Cardinal;
  end;

property Range: TacHTTPRange;
```

Description

The Range structure specifies a range of binary data for partial download. Set content ranges if you would like to download you would like to download just some defined part of the file.

For example, you would like to download the part of file, 40 bytes beginning from 50th byte. Then just set StartRange to 50 and EndRange to 90.

Or, for another instance, you would like to resume broken download and read the file beginning from 1234 bytes till the end of file. Set StartRange to 1234 and EndRange to 0 (unlimited range).

Remark

The Range works for binary files only. For dynamic ASCII content (i.e: HTML, output of CGI scripts etc) it will download entire file anyway.

See also

AddHeaders property.

9.31.2.14.1 EndRange

Applies to

acHTTP component as subproperty of Range structure.

Declaration

```
property EndRange: Integer;
```

Description

The EndRange property allows to specify the ending position (in bytes) of file for partial download.

© 1998-2006, UtilMind Solutions®

Set the EndRange to 0 if you would like to download data till the end of file.

See also
StartRange property.

9.31.2.14.2 StartRange

Applies to
acHTTP component as subproperty of Range structure.

Declaration
```
property StartRange: Integer; //
```

Description
The StartRange property specifies the starting position (in bytes) of the block of data to download.

For example, if StartRange = 50 and EndRange = 0, the acHTTP will download the file beginning from 50th byte till the end of file. If StartRange = 50 and EndRange = 89, it downloads 40 bytes from 50th till 89th byte.

If StartRange = 0, the acHTTP will read data from beginning of file.

See also
EndRange property.

9.31.2.15Referer

Applies to
acHTTP component.

Declaration
```
property Referer: String;
```

Description
The Referer property specifies the location of the document from which the URL in the request was obtained. If this parameter is empty, no "referrer" is specified.

See also
URL property.

9.31.2.16RequestMethod

Applies to
acHTTP component.

Declaration
```
type
  TacHTTPRequestMethod = (rmAutoDetect, rmGET, rmPOST);

property RequestMethod: TacHTTPRequestMethod;
```

Description
The RequestMethod *(HTTP method)* property is instruction sent in a request message that notifies an HTTP server of the action to perform on the specified resource.

For example, **rmGET** specifies that a resource is being retrieved from the server. **rmPOST** specifies that client should upload (post) some specific information which should processed by server before

© 1998-2006, UtilMind Solutions®

downloading the data.

When RequestMethod is **rmAutoDetect**, the HTTP component will autimatically detect how to request data from server. When POSTData string is empty, component will use GET method. If POSTData specified, POST method will be used.

When you reading data from CGI script using GET method, you should specify reqired information behind question mark, i.e.: http://www.website.com/cgi-bin/script.cgi?datafield1=datavalue2&datafield2=datavalue2

If CGI program accepts data via POST method, you should specify required data in the POSTData property together with the URL string.

Example 1: *(requesting data from the CGI script (at DelphiPages.com) via GET method)*

```
procedure TForm1.ReadBtnClick(Sender: TObject);
begin
  acHTTP1.URL :=
'http://www.delphipages.com/result.cfm?SR=HTTP&AO=and&RequestTimeout=500
';
  acHTTP1.Read;
end;

procedure TForm1.acHTTP1Done(Sender: TObject; ContentType: string;
FileSize: Integer; Stream: TStream);
begin
  // Receiving content from Stream
end;

{ Click link below to see demo:
http://www.delphipages.com/result.cfm?SR=HTTP&AO=and&RequestTimeout=500 }
```

Example 2: *(requesting data from the CGI script (at Torry.net) via POST method)*

```
procedure TForm1.ReadBtnClick(Sender: TObject);
begin
  acHTTP1.URL := 'http://www.torry.net/quicksearch.php';
  acHTTP1.POSTData := 'String=HTTP&Exact=Yes&Title=No';
  acHTTP1.Read;
end;

procedure TForm1.acHTTP1Done(Sender: TObject; ContentType: string;
FileSize: Integer; Stream: TStream);
begin
  // Receiving content from Stream
end;
```

See also

URL property.

9.31.2.17ShowGoOnlineMessage

Applies to

acHTTP component.

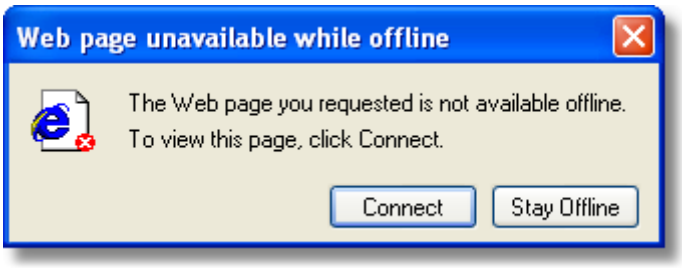
Declaration

© 1998-2006, UtilMind Solutions®

```
property ShowGoOnlineMessage: Boolean;
```

Description

The ShowGoOnlineMessage property controls whether the component could display standard Internet Explorer's dialog, requesting to switch to the online mode to download fresh content from specified URL.



Set ShowGoOnlineMessage property to True, if you want to display this dialog before HTTP request, when the global online status is "Work Offline" (can be specified by user selecting "File | Work Offline" menu item in Internet Explorer window).

See also

URL, WorkOffline properties;
IsGlobalOffline method.

9.31.2.18Suspended

Applies to

acHTTP, acSendMail and acThread components.

Declaration

```
property Suspended: Boolean;
```

Description

The Suspended property indicates whether a thread (used for downloading) is currenly suspended.

Set Suspended to True to suspend download process temporary; set it False to resume it.

See also

Thread, ThreadPriority and WaitThread properties.
Suspended property of acThread component

9.31.2.19Timeouts

Applies to

acHTTP component.

Declaration

```
type
  TacHTTPTimeouts = class(TPersistent)
  published
    property ConnectTimeout: Cardinal default 0;
    property ReceiveTimeout: Cardinal default 0;
    property SendTimeout: Cardinal default 0;
  end;

property Timeouts: TacHTTPTimeouts;
```

© 1998-2006, UtilMind Solutions®

Description
The Timeouts structure used to specify the time-out values for HTTP requests. All units are in milliseconds. If values are set to 0, the component will use default sustem values.

There are three time-out values:

ConnectTimeout	Sets or retrieves the time-out value to use for Internet connection requests. If a connection request takes longer than this time-out value, the request is canceled. When attempting to connect to multiple IP addresses for a single host (a multihome host), the timeout limit is cumulative for all of the IP addresses.
ReceiveTimeout	The time-out value, in milliseconds, to receive a response to a request. If the receiving of data takes longer than this time-out value, the receiving is canceled.
SendTimeout	The time-out value to send a request. If the send takes longer than this time-out value, the send is canceled.

9.31.2.19.1 ConnectTimeout

Applies to
acHTTP component.

Declaration
```
property ConnectTimeout: DWord;
```

Description
The ConnectTimeout property sets or retrieves the time-out value to use for Internet connection requests. If a connection request takes longer than this time-out value, the request is canceled. When attempting to connect to multiple IP addresses for a single host (a multihome host), the timeout limit is cumulative for all of the IP addresses.

See also
ReceiveTimeout and SendTimeout properties.

9.31.2.19.2 ReceiveTimeout

Applies to
acHTTP component.

Declaration
```
property ReceiveTimeout: DWord;
```

Description
The ReceiveTimeout property sets or retrieves the time-out value to use for Internet connection requests. If the receiving of data takes longer than this time-out value, the receiving is canceled.

See also
ConnectTimeout and SendTimeout properties.

9.31.2.19.3 SendTimeout

Applies to
acHTTP component.

Declaration
```
property SendTimeout: DWord;
```

© 1998-2006, UtilMind Solutions®

Description
The SendTimeout property sets or retrieves the time-out value to use for Internet connection requests. If the send takes longer than this time-out value, the send is canceled.

See also
ConnectTimeout and ReceiveTimeout properties.

9.31.2.20Thread

Applies to
acHTTP and acSendMail components.

Declaration
```
property Thread: TacCustomThread; // Read-only !!
```

Description
The Thread property is the pointer to the process thread, which used for downloading the data from the Web, avoiding suspending of main application thread. This is read-only public property.

See also
Suspended, ThreadPriority and WaitThread properties;
acThread component.

9.31.2.21ThreadPriority

Applies to
acHTTP, acSendMail, acHostnameResolver and acAutoUpgrader components.

Declaration
```
property ThreadPriority: TThreadPriority;
```

Description
ThreadPriority indicates the priority used when scheduling the thread. Adjust the priority higher or lower as needed.

TThreadPriority type defines the possible values for the Priority property of the acThread component, as defined in the following table. The system schedules CPU cycles to each thread based on a priority scale; the Priority property adjusts a thread's priority higher or lower on the scale.

Values	Meaning
tpIdle	The thread executes only when the system is idle. The system will not interrupt other threads to execute a thread with tpIdle priority.
tpLowest	The thread's priority is two points below normal.
tpLower	The thread's priority is one point below normal.
tpNormal	The thread has normal priority.
tpHigher	The thread's priority is one point above normal.
tpHighest	The thread's priority is two points above normal.
tpTimeCritical	The thread gets highest priority.

Warning
Boosting the thread priority of a CPU intensive operation may "starve" the other threads in the application. Only apply priority boosts to threads that spend most of their time waiting for external events.

See also

© 1998-2006, UtilMind Solutions®

Suspended, Thread and WaitThread properties.
acThread component.

9.31.2.22TransferBufferSize

Applies to
acHTTP and acAutoUpgrader components.

Declaration
```
type
  TacBufferSize = 256..MaxInt; // 2147483647 bytes maximum

property TransferBufferSize: TacBufferSize; // 4096 bytes by default
```

Description
The TransferBufferSize property specifies the size of buffer (in bytes) for writing or reading data from the Web. For example, if TransferBufferSize is 4096 and you call Read method, the acHTTP will read data by 4Kb blocks and trigger OnProgress event every time after downloading each 4Kb of data.

Also the value of TransferBufferSize serves as the size of rollback chunk, which automatically read the component when resuming the downloading to local file on call of Resume method.

See also
Read, Pause, Resume and Upload methods and OnProgress event.

9.31.2.23URL

Applies to
acHTTP and acWebImage components.

Declaration
```
property URL: String;
```

Description
The URL property specifies the location of the Web resource in the Internet (address of document, file, CGI program etc) to download data from remote HTTP server.

The URL address should be specified in following form:
```
[http[s]://]hostname[[:port]/objectname]
```
Examples:
```
http://www.appcontrols.com/download/diskcontrols_trial.exe
utilmind.com:80
https://secure.element5.com/register.html?productid=140005
```

The acHTTP supports secure transaction semantics. This translates to using Secure Sockets Layer/Private Communications Technology (SSL/PCT) and is only meaningful in HTTP requests. You can specify URLs either with **http://** or **https://** prefixes.

See also
Referer property.

9.31.2.24Username

Applies to
acHTTP components.

© 1998-2006, UtilMind Solutions®

Declaration

```
property Username: String;
```

Description

The Username property specifies the username to access the data in password protected Web directories. You don't need to specify the username if you reading non-protected data.

You can also specify the login information dynamically, when it necessary, in the OnPasswordRequest event handler.

See also

Password property and OnPasswordRequest event.

9.31.2.25WaitThread

Applies to

acHTTP, acSendMail, acThread and acSQLQueryThread components.

Declaration

```
property WaitThread: Boolean;
```

Description

The WaitThread property controls whether the procedure that calls the Read method (which downloads the data from the Web) should be suspended and wait until the scanning process will be done.

Set the WaitThread to True, if you would like to read the data from the Web so that the application does not continue with next lines of code after calling the Read method. Your application will done download (or inform about error) before continuing to next step.

If your application can wait, until the the HTTP request will be completed, only for some limited time interval — specify WaitTimeout property.

See also

WaitTimeout, Thread, ThreadPriority and Suspended properties;
Read method.

9.31.2.26WaitTimeout

Applies to

acHTTP, acSendMail, acThread and acSQLQueryThread components.

Declaration

```
property WaitTimeout: Integer;
```

Description

The WaitTimeout property specifies the time interval (limit), in milliseconds unit, which application able to wait until the HTTP request will be completed.

For example, if the maximum time which you can allow to complete HTTP request is 5 seconds, set this value to 5000 (milliseconds). If application can wait infinitely, set WaitTimeout to 0.

When the timeout is expired, the component automatically terminates the HTTP request. To be notified when the WaitTimeout is expired — write OnWaitTimeoutExpired event handler.

Notes

© 1998-2006, UtilMind Solutions®

The WaitTimeout only works together with WaitThread property, only when it set to True.

See also
WaitThread, Thread, ThreadPriority and Suspended properties;
Read and Abort methods;
OnWaitTimeoutExpired event.

9.31.2.27 WorkOffline

Applies to
acHTTP component.

Declaration
```
property WorkOffline: Boolean;
```

Description
The WorkOffline property controls whether you would like to browse the Web data offline, and read cached pages even if user are disconnected from Internet.

If requested file is not available for offline reading, the acHTTP component will try to connect the remote host to download data from the Web. If user is disconnected, OnHostUnreachable event will occurs.

This feature is the same as "Work Offline" option of the MS Internet Explorer. All requested data will be read from cache instead of downloading files from the Web.

The WorkOffline property works even if caching features is disabled (even if `coAlwaysReload` value of CacheOptions is True).

To determinate whether the global status is Offline, use IsGlobalOffline method-function of the component.

See also
CacheOptions property;
OnDone and OnHostUnreachable events.

9.31.3 Methods

9.31.3.1 Abort

Applies to
acHTTP, acSendMail, acAutoUpgrader, acWebImage and acSQLQueryThread components.

Declaration
```
procedure Abort(DeleteOutputFile: Boolean = False; HardTerminate:
Boolean = False);
```

Description
The Abort method terminates execution of thread which reads the data from the Web by HTTP(s) protocol. After calling the Abort method, the OnAborted event occurs.

The Abort method contains 2 optional parameters:

DeleteOutputFile — deletes the file, with downloaded information, specified in OutputFileName property, if True;

HardTerminate — terminate the thread which runs the download process imediately, without

© 1998-2006, UtilMind Solutions®

releasing Internet handles. ⚠ Do not set this parameter to True, unless it really necessary, since hard termination can lead to memory leaks!

ℹ The download process can be resumed, in case if you downloading data to file specified in OutputFileName property. You just need specify the incompletely downloaded file to OutputFileName property again and call Resume method.

However, even if you're downloading the data just to memory instead of file, the component can easily resume the downloading (on call of Read method), because of smart behaviours of Internet Explorer's cache. In case if you have specified to use cache in the CacheOptions property — everything can be retreived from cache, even broken, interrupted downloads.

See also
OutputFileName and CacheOptions properties;
Read, Pause and Resume methods;
OnAborted event.

9.31.3.2 IsGlobalOffline

Applies to
acHTTP component.

Declaration
```
function IsGlobalOffline: Boolean;
```

Description
The IsGlobalOffline method-function returns whether the global online status of Internet Explorer is *offline*. Users can change this status selecting "File | Work Offline" menu item in Internet Explorer's window.

See also
WorkOffline and ShowGoOnlineMessage properties.

9.31.3.3 Pause

Applies to
acHTTP component.

Declaration
```
procedure Pause;
```

Description
The Pause method terminates execution of running download process. Actually, this method is the same as "Abort" method with both its optional parameters set to False: `Abort(False, False)`.

ℹ The download process can be resumed, in case if you downloading data to file specified in OutputFileName property. You just need specify the incompletely downloaded file to OutputFileName property again and call Resume method.

However, even if you're downloading the data just to memory instead of file, the component can easily resume the downloading (on call of Read method), because of smart behaviours of Internet Explorer's cache. In case if you have specified to use cache in the CacheOptions property — everything can be retreived from cache, even broken, interrupted downloads.

See also

© 1998-2006, UtilMind Solutions®

OutputFileName and CacheOptions properties;
Read, Pause and Resume methods;
OnAborted event.

9.31.3.4 Read

Applies to
acHTTP component.

Declaration
```
function Read(ForceWaitThread: Boolean = False): Boolean; // returns
False if busy OR WaitTimeout expired
```

Description
The Read method initiate the HTTP request to download the data from location specified in the URL property. Function returns False if component currently busy (already processing request), OR WaitTimeout is expired (if you waiting for completion of request in the function that calls this medod, using WaitThread property).

The *ForceWaitThread* is optional (not necessary to specify) parameter, which can temporary set WaitThread to True for only current call of Read method.

Example *(code demonstrates how to search for 'HTTP' keyword in Torry.net)*
Delphi:
```
procedure TForm1.ReadBtnClick(Sender: TObject);
begin
  acHTTP1.URL := 'http://www.torry.net/quicksearch.php';
  acHTTP1.POSTData := 'String=HTTP&Exact=Yes&Title=No';
  acHTTP1.Read;
end;
```

C++ Builder:
```
void __fastcall TForm1::ReadBtnClick(TObject *Sender)
{
  acHTTP1->URL = "http://www.torry.net/quicksearch.php";
  acHTTP1->POSTData = "String=HTTP&Exact=Yes&Title=No";
  acHTTP1->Read();
}
```

Remarks
The Read method fails and OnHostUnreachable event occurs if you're trying to POST some data to the CGI but user currently *working offline* (even if connection to the Internet present). Posting to the CGI programs requires active Internet connection.

See also
URL, RequestMethod, POSTData and Busy properties;
WaitThread and WaitTimeout properties;
Upload, Abort, Pause and Resume methods;
HTTPReadString function.

9.31.3.5 ReadRange

Applies to
acHTTP component.

© 1998-2006, UtilMind Solutions®

Declaration

```
function ReadRange(StartRange: Cardinal; EndRange: Cardinal = 0;
ForceWaitThread: Boolean = False);
```

Description

The ReadRange property downloads part of some binary data file from the Web. The part of that file can be specified by *StartRange* and *EndRange* parameters.

StartRange parameter specifies starting position of data block to download, and the *EndRange* parameter specified the end of the block. In case if EndRange is 0, the component will download part from *StartRange* till the end of file.

Alternatively, for partial download, you can use StartRange and EndRange properties in the Range structure and use usual Read method.

See also

URL, RequestMethod and Busy properties;
WaitThread and WaitTimeout properties;
Read, Abort, Pause, Resume methods.

9.31.3.6 Resume

Applies to

acHTTP component.

Declaration

```
procedure Resume;
```

Description

The Resume method used to resume downloading of data to some file specified in the OutputFileName property.

After calling of this method, the component determinates the size of file specified in the OutputFileName, and initiates downloading of the rest of broken or paused download, to append the rest of data to the end of local file.

The acHTTP automatically uses simple and smart scheme of checking whether the file which you're trying to Resume has been updated or modified. Before downloading of the data which should be appended, it downloads small data chunk (with size specified in TransferBufferSize property), *before the break*, and compares with the same data chunk at the end of file.

In case if compared data are equal — it continue downloading and append downloaded data to the end of local file. Otherwise it assume that file which beging downloaded has been changed, and starts download from beginning.

By default TransferBufferSize = 4Kb, so every time when you call Resume method, the component download 4Kb of extra "rollback" data to check file consistancy.

Note

Use Resume method only if you downloading the data to file, specified in the OutputFileName property. Otherwise, in case if file name not specified, it will download some object from the Web from beginning. Effect will be the same as you calling Read method.

You can use Resume method instead of Read. In case if file does not exists — Resume method will create it and download some file from beginning.

© 1998-2006, UtilMind Solutions®

See also
OutputFileName, CacheOptions and TransferBufferSize properties;
Read, Pause and Abort methods.

9.31.3.7 Upload

Applies to
acHTTP component.

Declaration
```
function Upload(NumberOfFields: Word): Boolean; // returns False if busy
```

Description
The Upload method starts HTTP request to upload data via HTTP protocol, using *multipart/form-data* POST method, introduced in RFC 1867.

Before starting the uploading, it requests fields and their names which should be uploaded using OnUploadFieldRequest event. To specify number of fields which should be uploaded — pass it in *NumberOfFields* parameter (this specifies how many times the OnUploadFieldRequest should be triggered to request another piece of data).

After requesting the data required to build HTTP request, it starts it with *multipart/form-data* Content Type in the HTTP header and constantly trigger OnUploadProgress event after each data block with size specified in TransferBufferSize property.

Example
```
procedure TForm1.UploadBtnClick(Sender: TObject);
begin
  acHTTP1.Upload(2); // upload 2 files
{ To specify the data which should uploaded — use OnUploadFieldRequest
event }
end;
```

Remarks
Unfortunately the web server itself can NOT receive files by HTTP protocol. For this purpose you should use some intermediate CGI program, in example, written in C, Perl or PHP (or even in Delphi, if you're running Windows server). If you would like to get examples on how to create scripts which can receive files by HTTP protocol, please check out PHP.net (PHP manuals), or www.cgi-resources.com (CGI Resource Index).

Check out also **OnUploadFieldRequest** topic for more detailed description on how to upload data.

Some versions of Apache HTTP server has a bug which does not allow to upload files to password protected directories. In case if you always receive timeout error when trying to upload file to password protected URL and even modifications of timeout values in PHP.INI won't help, don't despair and try to upload it to normal directory.

See also
UploadByFieldNames method;
OnUploadFieldRequest, OnUploadProgress and OnUploadCGITimeoutFailed events;
Read, Abort methods and TransferBufferSize property.

9.31.3.8 UploadByFieldNames

Applies to
acHTTP component.

© 1998-2006, UtilMind Solutions®

Declaration

```
function UploadByFieldNames(const FieldNames: Array of String): Boolean;
// returns False if busy
```

Description

The UploadByFieldNames, like the Upload method starts HTTP request to upload data via HTTP protocol, using *multipart/form-data* POST method, introduced in RFC 1867.

However, unlike, the Upload method, the UploadByFiles allows to specify the field names as parameters, so in the OnUploadFieldRequest event you just need to write the data to *UploadStream* (data proper to each *FieldName*), without requiring to specify the *FieldName*'s.

Before starting the uploading, it requests fields which should be uploaded using OnUploadFieldRequest event. To specify number of fields which should be uploaded — pass it in *NumberOfFields* parameter (this specifies how many times the OnUploadFieldRequest should be triggered to request another piece of data).

After requesting the data required to build HTTP request, it starts it with *multipart/form-data* Content Type in the HTTP header and constantly trigger OnUploadProgress event after each data block with size specified in TransferBufferSize property.

Example

```
procedure TForm1.UploadBtnClick(Sender: TObject);
begin
  acHTTP1.Upload(['field1', 'field2']); // upload 2 files
{ To specify the data which should uploaded – use OnUploadFieldRequest
event }
end;
```

Remark

Unfortunately the web server itself can NOT receive files by HTTP protocol. For this purpose you should use some intermediate CGI program, in example, written in C, Perl or PHP (or even in Delphi, if you're running Windows server). If you would like to get examples on how to create scripts which can receive files by HTTP protocol, please check out PHP.net (PHP manuals), or www.cgi-resources.com (CGI Resource Index).

Check out also **OnUploadFieldRequest** topic for more detailed description on how to upload data.

See also

UploadByFieldNames method;
OnUploadFieldRequest and OnUploadProgress events;
Read, Abort methods and TransferBufferSize property.

9.31.4 Events

9.31.4.1 OnAborted

Applies to

acHTTP, acSendMail, acAutoUpgrader and acWebImage components.

Declaration

```
property OnAborted: TNotifyEvent;
```

Description

The OnAborted event occurs when user interrupts the process of reading the data from Internet, after calling the Abort method.

© 1998-2006, UtilMind Solutions®

See also
Abort method;
OnWaitTimeoutExpired event.

9.31.4.2 OnAnyError

Applies to
acHTTP and acSendMail components.

Declaration
```
property OnAnyError: TNotifyEvent;
```

Description
The OnAnyError event occurs when ANY error has occured: connection lost (OnConnLost), host unreachable (OnHostUnreachable) or server returned the HTTP error in response header (OnHTTPError).

See also
OnConnLost, OnHostUnreachable and OnHTTPError events.

9.31.4.3 OnBeforeSendRequest

Applies to
acHTTP component.

Declaration
```
type
  TacHTTPBeforeSendRequest = procedure(Sender: TObject; hRequest:
hInternet) of object;

property OnBeforeSendRequest: TacHTTPBeforeSendRequest;
```

Description
The OnBeforeSendRequest event occurs just before the component sends HTTP query to the server, at once after opening the internet request (by HTTPOpenRequest function of WinInet). The hRequest parameter is the Internet handle, returned by HTTPOpenRequest and can be used to specify additional Internet options to the request.

Write OnBeforeSendRequest event handler if you want to specify custom Internet options using "low level" InternetSetOptions function (from WinInet unit), to the hRequest handle.

Example
```
procedure TForm1.acHTTP1BeforeSendRequest(Sender: TObject;
  hRequest: Pointer);
begin
   InternetSetOption(hRequest, INTERNET_OPTION_IGNORE_OFFLINE, nil, 0);
end;
```

9.31.4.4 OnConnLost

Applies to
acHTTP and acSendMail components *(but with different parameters)*.

Declaration
```
type
  TacHTTPConnLostEvent = procedure(Sender: TObject;
    const ContentType: String; FileSize, BytesRead: Integer;
```

© 1998-2006, UtilMind Solutions®

```
    Stream: TStream) of object;

property OnConnLost: TacHTTPConnLostEvent;
```

Description

The OnConnLost event occurs when the connection with remote server lost for some reason, at the moment of downloading the data. However you still can use some data which was already downloaded (*Stream* parameter).

There are following parameters which passes to the event handler:

Parameters	Meaning
ContentType	the media type of received data. For example if you downloaded usual text-file, the ContentType will be "text/plan". For HTML page ContentType will "text/html", for executable file — "application/binary" and "image/jpeg" for JPEG, JPG and JPE files. For more information about Internet media types, please read RFC 2045, 2046, 2047, 2048, and 2077 (http://www.oac.uci.edu/indiv/ehood/MIME/MIME.html). Check out also the Internet media type registry at ftp://ftp.iana.org/in-notes/iana/assignments/media-types;
FileSize	total size of data which we've tryed to download, in bytes (if was possible to determinate). Note: some servers can send files without information about content length;
BytesRead	size of data which already received, in bytes;
Stream	stream which contains already downloaded data. (Check out description of TStream, TMemoryStream and TFIleStream classes for more info).

See also

OnAnyError, OnProgress and OnDone events.

9.31.4.5 OnDone

Applies to

acHTTP, acWebImage and acSQLQueryThread components.

Declaration

```
type
  TacHTTPDoneEvent = procedure(Sender: TObject;
    const ContentType: String;
    FileSize: Integer; Stream: TStream) of object;

property OnDone: TacHTTPDoneEvent;
```

Description

The OnDone event occurs when the acHTTP component has successfully downloaded requested Web resource.

The component pass to the OnDone event handler 3 following parameters:

Parameters	Meaning
ContentType	the media type (also known as Multipurpose Internet Mail Extension (MIME) type) of requested document. For example if you downloaded usual text-file, the ContentType will be "text/plan". For HTML page ContentType will "text/html", for executable file — "application/binary" and "image/jpeg" for JPEG, JPG and JPE files. For more information about Internet media types, please read RFC 2045, 2046, 2047, 2048, and 2077 (http://www.oac.uci.edu/indiv/ehood/MIME/MIME.html). Check out also the Internet media type registry at ftp://ftp.iana.org/in-

© 1998-2006, UtilMind Solutions®

	notes/iana/assignments/media-types;
FileSize	size of downloaded data in bytes;
Stream	the stream which contains downloaded data (this is actually TMemoryStream. Check out description of TMemoryStream class for more details on how to retreive data from it). However, it can be nil (null) if the OutputFileName property was specified before request (before calling the Read method).

Example

Delphi:

```
procedure TForm1.acHTTP1Done(Sender: TObject;
  ContentType: string; FileSize: Integer; Stream: TStream);
var
  Str: String;
begin
  if Stream = nil then
    Exit; // can be already stored to file specified by OutputFileName

  with TMemoryStream(Stream) do // yes, this is actually TMemoryStream
   if OutToMemoBox1.Checked then // output to Memo1
    begin
     Memo1.Lines.LoadFromStream(Stream);

{    // alternatively:
     SetLength(Str, Size);
     Stream.Read(Str[1], Size); // or Move(Memory^, Str[1], Size);
     Memo1.Text := Str; }
    end
   else
    begin // save to file
     Memo1.Text := 'Saved to c:\httptest.dat';
     SaveToFile('c:\httptest.dat');
    end;

  StatusBar1.Panels[0].Text := 'Successfully downloaded ' +
IntToStr(FileSize) + ' bytes';
end;
```

C++ Builder:

```
void __fastcall TForm1::acHTTP1Done(TObject *Sender,
      AnsiString ContentType, int FileSize, TStream *Stream)
{
  AnsiString Str;

  if (Out1->Checked) {
    Str.SetLength(Stream->Size);
    Move(((TMemoryStream*)Stream)->Memory, &Str[1], Stream->Size);
    Memo1->Text = Str;
  }else {
    Memo1->Text = "Saved to c:\httptest.dat";
    ((TMemoryStream*)Stream)->SaveToFile("c:\httptest.dat");
  }

  StatusBar1->Panels->Items[0]->Text = "Successfully downloaded " +
IntToStr(FileSize) + " bytes";
```

© 1998-2006, UtilMind Solutions®

```
}
```

See also
OnProgress, OnHTTPError, OnHeaderInfo and OnDoneInterrupted events;
Read and Abort methods;
OutputFileName, OutputFileAttributes and FileName properties.

9.31.4.6 OnDoneInterrupted

Applies to
acHTTP component.

Declaration
```
property OnDoneInterrupted: TNotifyEvent;
```

Description
The OnDoneInterrupted event occurs if the download process was interrupted in OnHeaderInfo event handler.

This is optional event to be notified when the component has terminated the HTTP query after receving the headers of the document before that the download process begins. This event only executed if you set *ContinueDownload* parameter to False in the OnHeaderInfo event handler.

Note, that no errors events will be receive if you set *ContinueDownload* of OnHeaderInfo to True, but this event. So you must handle all errors inside OnHeaderInfo event.

See also
OnHeaderInfo event.

9.31.4.7 OnHeaderInfo

Applies to
acHTTP component.

Declaration
```
type
  TacHTTPHeaderInfoEvent = procedure(Sender: TObject; ErrorCode:
Integer;
    const RawHeadersCRLF, ContentType, ContentLanguage, ContentEncoding:
String;
    ContentLength: Integer; const Location: String;
    const Date, LastModified, Expires: TDateTime; const ETag: String;
    var ContinueDownload: Boolean) of object;

property OnHeaderInfo: TacHTTPHeaderInfo;
```

Description
The OnHeaderInfo event returns the headers the response from the HTTP server, *before downloading* the document content.

You can write this event handler to receive all response headers and to decide whether you want to download the document. If you decided to NOT download it, (for example, if ErrorCode is not 200-OK, and not 206-Partial content) simply set *ContinueDownload* parameter to False in the event handler.

The acHTTP passes to the OnHeaderInfo event handler following parameters:

© 1998-2006, UtilMind Solutions®

Parameters	Meaning
ErrorCode	the status code of HTTP request (see the list of possible status codes);
RawHeadersCRLF	contains ALL the headers reseived from the HTTP server in response to request, as plain text string, separated by CRLF characters (0D0A);
ContentType	contains the identifier of MIME-type of requested document. For more information about Internet media types, please read RFC 2045, 2046, 2047, 2048, and 2077. Check out also the Internet media type registry at ftp://ftp.iana.org/in-notes/iana/assignments/media-types;
ContentLanguage	identifies a language of document content (if provided), or contains empty string if the language is not provided or not applicable for the type requested document;
ContentEncoding	identifies the encoding method of requested document;
ContentLength	determinates the size of document, if the document is binary file. Unfortunately most servers does not provide the content length for ASCII documents with "text/*" MIME-type, since their content can be generated dynamically by CGI programs (in Perl or PHP);
Location	determines the location from where the content is about to be downloaded (use this parameter to get the actual location of document in case if connection has been redirected by server to another location);
Date	shows the date and time at which the HTTP response was originated;
LastModified	the date and time at which the server belives the resource was last modified. *Note: Servers without a clock assign ETag parameter instead of last modified and expiration time*;
Expires	the date and time after which the resource should be considered outdated. *Note: Servers without a clock assign ETag parameter instead of last modified and expiration time*;
ETag	ETag, also known as "*Expires Tag*", or some another additional information from server. This information generated automatically by server without a clock (in this case *LastModified* and *Expires* values are not set), or generated dynamically by CGI program (in Perl or PHP) and used to transfer some important information, i.e: whether the document expired etc;
ContinueDownload	used to interrupt the download process. If you don't want to continue download the requested file — set it to False.

Notes

All date/time values that passed to the OnHeaderInfo even handler, is already converted from the server's time to local time, taking in account the time zone and daylight bias. (The acHTTP uses InternetTimeToDateTime function to convert the string representation of date/time to TDateTime value.) To get unmodified string parameters — parse the raw headers provided by *RawHeadersCRLF* parameter.

This event does not occur if you downloading data from local file (use "file://" prefix in the URL).

If you set ContinueDownload parameter to True (imediately terminate the process without downloading of the content of document), neither OnAnyError and OnHTTPError events will not called. The only event which you will received after it — is OnDoneInterrupted. This means that if you plan to handle HTTP errors in OnHTTPError event handler, you must move this code to OnHeaderInfo event handler.

See also

OnDone, OnDoneInterrupted, OnHTTPError and OnAnyError events;
Abort method;

© 1998-2006, UtilMind Solutions®

HTTP status codes.

9.31.4.8 OnHostUnreachable

Applies to
acHTTP, acSendMail and acAutoUpgrader components.

Declaration
```
property OnHostUnrachable: TNotifyEvent;
```

Description
The OnHostUnreachable event occurs if the acHTTP can not connect to the remote host specified in the URL property. Possible reasons of this problem is:
1. User currently not connected to the Internet;
2. Hostname is unknown (check spelling of domain name);
3. Remote server is down (disconnected from Internet).

Remarks
1. The OnHostUnreachable event occurs also when user currently *working offline* (even if connection to the Internet present) and would like to post some data, specified in the POSTData property, to the CGI program. Posting to the CGI programs requires active Internet connection.

Example
Delphi:
```
procedure TForm1.acHTTP1HostUnreachable(Sender: TObject);
begin
   Application.MessageBox(PChar('Host http://' + acHTTP1.HostName + ' is
unreachable.'#13#10#10'Please check your Internet connection
and'#13'retry your HTTP request again later.'),
                          PChar(Application.Title),
                          MB_OK or MB_ICONSTOP);
end;
```

C++ Builder:
```
void __fastcall TForm1::acHTTP1HostUnreachable(TObject *Sender)
{
  AnsiString Msg =
      "Host http://" + acHTTP1->HostName + " is unreachable.\n\n"
      "Please check your Internet connection and\n"
      "try to upgrade this software again later.";
   Application->MessageBox(Msg.c_str(),
                           Application->Title.c_str(),
                           MB_OK | MB_ICONSTOP);
}
```

See also
URL property; OnAnyError event.

9.31.4.9 OnHTTPError

Applies to
acHTTP component.

Declaration
```
type
  TacHTTPErrorEvent = procedure(Sender: TObject;
    ErrorCode: Integer; Stream: TStream) of object;
```

© 1998-2006, UtilMind Solutions®

```
property OnHTTPError: TacHTTPErrorEvent;
```

Description

The OnHTTPError event occurs if some error code has received in the header of response from HTTP server.

ErrorCode parameter contains the number which identifies the HTTP error (see the list of HTTP Status Codes to recognize an error).

Stream contains the error page generated by server.

Most often errors is 404 (requested document not found), 403 (view forbidden) and 500 (CGI script failed).

To handle error #401 (Access denied / Password required required to access) — write OnPasswordRequest event handler. *(If OnPasswordRequest event handler exists, OnHTTPError will not occurs on error 401.)*

Alternatively you can get HTTP error code in OnHeaderInfo event handler and decide whether to continue download (error page), or not, so this will save some client's bandwidth from downloading error page from server.

Remarks

If it returns 0 in *ErrorCode* paramter, this means that component for some reason is unable to determinate the status code of HTTP query. However this is not always means fatal error like OnHostUnreachable. It's possible that server simply did not sent the status code in the header of response.

Example

Delphi:

```
procedure TForm1.acHTTP1HTTPError(Sender: TObject;
  ErrorCode: Integer; Stream: TStream);
var
  Str: String;
begin
  with Stream as TMemoryStream do
   if OutToMemo1.Checked then
    begin // Output to Memo1
     SetLength(Str, Size);
     Move(Memory^, Str[1], Size);
     Memo1.Text := Str;
    end
   else // Save to file
    begin
     Memo1.Text := 'Saved to c:\httptest.dat';
     SaveToFile('c:\httptest.dat');
    end;

  case ErrorCode of
    404: Str := '404: Document not found';
    500: Str := '500: CGI script failed';
    else // Mysterious reason
      Str := IntToStr(ErrorCode);
   end;
```

© 1998-2006, UtilMind Solutions®

```
  if (ErrorCode = HTTP_STATUS_OK) or (ErrorCode =
HTTP_STATUS_PARTIAL_CONTENT) then // consts from WinInet.pas
   begin
    ContinueDownload := False;
    Exit;
   end;

  StatusBar1.Panels[0].Text := 'HTTP Error #' + Str;
end;
```

C++ Builder:

```
void __fastcall TForm1::acHTTP1HTTPError(TObject *Sender,
   int ErrorCode, TStream *Stream)
{
  AnsiString Str;

  if (OutToMemo1->Checked) { // Output to Memo1
    Str.SetLength(Stream->Size);
    Move(((TMemoryStream*)Stream)->Memory, &Str[1], Stream->Size);
    Memo1->Text = Str;
  }else { // Save to file
    Memo1->Text = "Saved to c:\httptest.dat";
    ((TMemoryStream*)Stream)->SaveToFile("c:\httptest.dat");
  }

  switch (ErrorCode) {
    case 404: Str = "404: Document not found";
    case 500: Str = "500: CGI script failed";
    default:  Str = IntToStr(ErrorCode); // Mysterious reason
  };

  StatusBar1->Panels->Items[0]->Text = "HTTP Error #" + Str;
}
```

See also

HTTP Status Codes
OnHeaderInfo, OnPasswordRequest, OnAnyError, OnDone and OnHostUnreachable events.

9.31.4.10 OnOutputFileError

Applies to

acHTTP component.

Declaration

```
property OnOutputFileError: TNotifyEvent;
```

Description

The OnOutputFileError occurs if the HTTP component tries to download data to file, specified in OutputFileName property, but the file can not be created (e.g. path not exists, or file locked by system etc).

See also

OutputFileName and OutputFileAttributes properties;
OnAnyError event.

© 1998-2006, UtilMind Solutions®

9.31.4.11 OnPasswordRequest

Applies to

acHTTP, acAutoUpgrader and acWebImage components.

Declaration

```
type
  TacHTTPPasswordRequestEvent = procedure(Sender: TObject;
    const Realm: String; var TryAgain: Boolean) of object;

property OnPasswordRequest: TacHTTPPasswordRequestEvent;
```

Description

The OnPasswordRequest event can be used to implement the dialog which asks user for his username and password to access the password protected Web area.

Set *TryAgain* parameter to True to retry the HTTP query with correct username and password. (Don't forget to specify correct login information in this event handler to Username and Password properties.)

Note

If you leave this event unhandled (set TryAgain parameter to False), the component will generate error code #401 (Access Denied), which will be passed to OnHTTPError event.

Example

Delphi:

```
procedure TForm1.HTTP1PasswordRequest(Sender: TObject;
  const Realm: String; var TryAgain: Boolean);
begin
  { UserPassForm is any form with two edit boxes used for entering the
login information (username and password) }
  UserPassForm.RealmLabel := Realm;
  if UserPassForm.ShowModal = ID_OK then
  begin
    acHTTP1.Username := UserPassForm.UsernameEdit.Text;
    acHTTP1.Password := UserPassForm.PasswordEdit.Text;
    TryAgain := True; // Retry HTTP query (download attempt)
  end;
end;
```

C++ Builder:

```
void __fastcall TForm1::acHTTP1PasswordRequest(TObject *Sender,
      AnsiString Realm, bool &TryAgain)
{
  UserPassForm->RealmLabel = Realm;
  if (UserPassForm->ShowModal() == ID_OK)
  {
    acHTTP1->Username = UserPassForm->UsernameEdit->Text;
    acHTTP1->Password = UserPassForm->PasswordEdit->Text;
    TryAgain = True; // Retry HTTP query (download attempt)
  };
}
```

See also

© 1998-2006, UtilMind Solutions®

Username and Password properties; OnHTTPError event;
HTTP Status Codes.

9.31.4.12 OnProgress

Applies to

acHTTP, acAutoUpgrader and acSendMail components.

Declaration

```
type
  TauHTTPProgressEvent = procedure(Sender: TObject;
    const ContentType: String; FileSize, BytesRead, ElapsedTime,
    EstimatedTimeLeft: Integer; PercentsDone: Byte;
    TransferRate: Single; Stream: TStream) of object;

property OnProgress: TacHTTPProgressEvent;
```

Description

The OnProgress event occurs every time when component has successfully downloaded part of data received in response to HTTP query.

Write OnProgress event handler to show the download progress (*PercentsDone* parameter indicate the progress in percents) or handle part of data which already received (*Stream* parameter). Also, the acHTTP component automatically calculates elapsed and estimated time before finishing download and speed of data transfer.

The acHTTP passes to the OnProgress event handler following parameters:

Parameter	Meaning
ContentType	the MIME-type of received data. For example if you downloaded usual text-file, the ContentType will be "text/plan". For HTML page ContentType will "text/html", for executable file — "application/binary" and "image/jpeg" for JPEG, JPG and JPE files. For more information about Internet media types, please read RFC 2045, 2046, 2047, 2048, and 2077. Check out also the Internet media type registry at ftp://ftp.iana.org/in-notes/iana/assignments/media-types;
DataSize	total size of data which we currently downloading, in bytes (if possible to determinate). **Note:** usually server does not provide information about content-length for non-binary data (i.e: for "text/html" or "text/plain" content types);
BytesRead	size of already received data, in bytes;
ElapsedTime	time elapsed from beginning of download (in seconds);
EstimatedTimeLeft	estimated time left before finishing of download (Formula: `X := FileSize / BytesRead * ElapsedTime - ElapsedTime`);
PercentsDone	progress in percents (0%..100%);
TransferRate	speed of data transfer (in Kb/s);
Stream	stream which contains already downloaded data. (Check out description of TStream, TMemoryStream and TFileStream classes for more info).

The OnProgress event triggered after downloading each block of data with size specified in TransferBufferSize property

Example

Delphi:

```
procedure TForm1.HTTP1Progress(Sender: TObject; ContentType: String;
```

© 1998-2006, UtilMind Solutions®

```
    FileSize, BytesRead, ElapsedTime, EstimatedTimeLeft: Integer;
    PercentsDone: Byte; TransferRate: Single; Stream: TStream);
begin
  // progress bar position
  CurrentFileProgressBar.Position := PercentsDone;

  // file size
  FileSizeLabel.Caption :=
       Format('File size: %.1f Kb', [FileSize / 1024]);

  // downloaded (in Kb)
  DownloadedLabel.Caption :=
      Format('Downloaded: %.1f Kb:', [BytesRead / 1024]);

  // transfer rate
  TransRateLabel.Caption :=
       Format('Transfer rate: %.1f Kb/s', [TransferRate]);

  // estimated time left
  EstTimeLeftLabel.Caption :=
       Format('Estimated time left: %d:%.2d:%.2d',
          [EstimatedTimeLeft div 60 div 60,   // hours
           EstimatedTimeLeft div 60 mod 60,   // minutes
           EstimatedTimeLeft mod 60 mod 60]); // seconds
end;
```

C++ Builder:

```
void __fastcall TForm1::acHTTP1Progress(TObject *Sender,
      AnsiString ContentType, int FileSize, int BytesRead,
      int ElapsedTime, int EstimatedTimeLeft,
      BYTE PercentsDone, float TransferRate, TStream *Stream)
{
  // progress bar position
  ProgressCurrentFile->Position = PercentsDone;

  // file size
  FileSizeLabel->Caption =
      Format("File size: %.1f Kb:",
      ARRAYOFCONST(((float)FileSize / 1024)));

  // downloaded (in Kb)
  DownloadedLabel->Caption =
      Format("Downloaded: %.1f Kb:",
      ARRAYOFCONST(((float)BytesRead / 1024)));

  // transfer rate
  TransRateLabel->Caption =
      Format("Transfer rate: %.1f Kb/s",
      ARRAYOFCONST(((float)TransferRate)));

  // estimated time left
  EstTimeLeftLabel->Caption =
      Format("Estimated time left: %d:%.2d:%.2d",
      ARRAYOFCONST((EstimatedTimeLeft / 60 / 60,    // hours
                    EstimatedTimeLeft / 60 % 60,    // minutes
```

© 1998-2006, UtilMind Solutions®

```
                        EstimatedTimeLeft % 60 % 60))); // seconds
}
```

See also
OnDone event.

9.31.4.13OnProxyAuthenticationRequest

Applies to
acHTTP, acAutoUpgrader and acWebImage components.

Declaration

```
type
  TacHTTPProxyAuthenticationRequestEvent = procedure(Sender: TObject;
    var ProxyUsername, ProxyPassword: String;
    var TryAgain: Boolean) of object;

property OnProxyAuthenticationRequest:
TacHTTPProxyAuthenticationRequestEvent;
```

Description
The OnProxyAuthenticationRequest event should be used to prompt users for their username/password to access the Web via secure proxy server which requires authentication.

Write this event to prompt and specify the *ProxyUsername* and *ProxyPassword* parameters, required for the proxy authentication, and set *TryAgain* parameter to True, to retry the HTTP query with provided login information.

Alternatively you can specify ProxyUsername and ProxyPassword properties in the Proxy structure, before the request. In case if specified username and password is okay, the OnProxyAuthenticationRequest event will not occur.

When you specify *ProxyUsername* and *ProxyPassword* parameters in this event — they also will be put to the Proxy structure, to be used on next HTTP request.

Notes
If your proxy server uses some "challenge-response" authentication scheme — please make sure that `ioKeepConnection` option of the InternetOptions is set to True. Otherwise, connection with proxy could be dropped.

If you leave this event unhandled (if you set TryAgain parameter to False), the component will generate error code #407 (Proxy Authentication Required), which will be passed to OnHTTPError event.

See also
Proxy and InternetOptions properties;
OnHTTPError event;
HTTP Status Codes.

9.31.4.14OnRedirected

Applies to
acHTTP component.

Declaration

```
type
```

© 1998-2006, UtilMind Solutions®

```
  TacHTTPRedirected = procedure(Sender: TObject; const NewURL: String)
of object;

property OnRedirected: TacHTTPRedirected;
```

Description

The OnRedirected event occurs if the server has redirected the HTTP query to another location (when redirection has been detected by internal status callback procedure). This means that the data which about to be posted, and the document which about to be downloaded in response of your query, will be taken from another location, specified in *NewURL* parameter.

See also

Location parameter in OnHeaderInfo event.

9.31.4.15OnUploadCGITimeoutFailed

Applies to

acHTTP component.

Declaration

```
procedure OnUploadCGITimeoutFailed: TNotifyEvent;
```

Description

The OnUploadCGITimeoutFailed event occurs if server has dropped connection while uploading some files to CGI script.

If the CGI program is PHP script, then sometime this can be fixed by increasing of "*max_execution_time*" and "*max_input_time*" variables in configuration (PHP.ini).

See also

Upload method.

9.31.4.16OnUploadFieldRequest

Applies to

acHTTP component.

Declaration

```
type
  TacHTTPUploadFieldRequest = procedure(Sender: TObject;
    FileIndex: Word; UploadStream: TStream;
    var FieldName, FileName: String) of object;

procedure OnUploadFieldRequest: TacHTTPUploadFieldRequest;
```

Description

The OnUploadFieldRequest should be used to put the *FieldName*, *FileName* (if required), and data to the stream (*UploadStream* parameter) for further uploading to the CGI application.

The acHTTP passes to the OnUploadFieldRequest event handler following parameters:

Parameter	Meaning
FileIndex	parameter specifies the index of data-field/file which should be uploaded. (Note: Total number of fields/files which should be uploaded must be specified on call of Upload method. This parameter is the index of file in queue.)
UploadStream	parameter is the empty stream which should be used to write data for uploading.

© 1998-2006, UtilMind Solutions®

(Use Stream.Write() method to put data to stream, however, since this is TMemoryStream you can use other methods of TMemoryStreams).

FieldName should be specified in this event handler. This is the name of field of the "web form".

FileName is the optional parameter used to specify local path and filename of uploaded file. It does not transmitted to CGI if empty. Use it only if your CGI application should know the real filename. Leave the *FileName* empty if it's not required for your CGI.

Example

```
procedure TForm1.acHTTP1UploadFieldRequest(Sender: TObject;
  FileIndex: Word; UploadStream: TStream; var FieldName, FileName:
String);
begin
  if FileIndex = 0 then // first file
   begin
    FieldName := 'img1';
    FileName := 'c:\1.jpg';
   end
  else // second file, if FileIndex = 1
   begin
    FieldName := 'img2';
    FileName := 'c:\2.jpg';
   end;

  // put file data to stream
   TMemoryStream(UploadStream).LoadFromFile(FileName);
end;
```

Remarks

Unfortunately the web server itself can NOT receive files by HTTP protocol. For this purpose you should use some intermediate CGI program, in example, written in C, Perl or PHP (or even in Delphi, if you're running Windows server). If you would like to get examples on how to create scripts which can receive files by HTTP protocol, please check out PHP.net (PHP manuals), or www.cgi-resources.com (CGI Resource Index).

Here is an example of PHP script which we are using to upload picture from one our program:

```
<?php

$picdir  = '/data/www/domains/images.utilmind.com/images/';

if (($index == '') || ($user == '')) die('0'); // not enough
parameters POST'ed

$big   = $picdir.$user.$index.'.big'; // big picture
$small = $picdir.$user.$index.'.small'; //thumbnailed image

if ((isset($pic)) && (isset($smallpic))) { // upload (else -- delete
it)
  copy($pic,      $big)   or die('2'); // store big picture
  copy($smallpic, $small) or die('3'); // store thumbnailed image
} else {
  if (file_exists($big))   unlink($big); // delete big picture
  if (file_exists($small)) unlink($small); //delete thumbnailed image
}
```

© 1998-2006, UtilMind Solutions®

```
?>
1
```

If script above does not work for some reason — use $_FILES predefined variable ($_FILES['fieldname']['name'] returns the name of uploaded file, $_FILES['fieldname']['tmp_name'] returns the location where uploaded data temporary stored. Following PHP function moves temporary file to permanent location:

```
move_uploaded_file($_FILES['field_name']['tmp_name'],
$_SERVER['DOCUMENT_ROOT'].'/permanent_location/file.dat') or
die('Cannot copy file!');
```

Here is the link to nice file uploader class in PHP: http://dave.imarc.net/downloads/fileupload.zip

And on Client side we are using following code to POST required data:

```
procedure  TMEMPicUploader.UploadPictureHTTPFieldRequest(Sender:
TObject;
  FileIndex: Word; UploadStream: TStream; var FieldName,
  FileName: String);
const
  FieldNames: Array[0..3] of String = ('user', 'index', 'pic',
'smallpic');
var
  W, H: Integer;
  PicIndexStr: String;
  BigImage, SmallImage: TacProportionalImage;
begin
  FieldName := FieldNames[FileIndex];

  with UploadStream, Client.MyProfile do
   case FileIndex of
     0: Write(Username[1], Length(Username));
     1: begin
          PicIndexStr := IntToStr(FPictureIndex);
          UploadStream.Write(PicIndexStr[1], Length(PicIndexStr));
        end;
     else
       FileName := Username;

       if FileIndex = 2 then
         FPicture.Graphic.SaveToStream(UploadStream) // normal picture
       else
        begin // thumbnailed picture
          BigImage := TacProportionalImage.Create(Self);
          try
            BigImage.Picture.Assign(FPicture);
            BigImage.Width := FThumbnailWidth;
            BigImage.Height := FThumbnailHeight;
            SmallImage := TacProportionalImage.Create(Self);
           try
             // CREATE THUMBNAILED IMAGE
            with BigImage, DrawRect do
             begin
              W := Right - Left;
              H := Bottom - Top;
              with SmallImage.Picture.Bitmap do
               begin
```

© 1998-2006, UtilMind Solutions®

```
                    Width  := W;
                    Height := H;
                  end;
                 SmallImage.Canvas.StretchDraw(Rect(0, 0, W, H),
  Picture.Graphic);
               end;

              with TJPEGImage.Create do
               try
                  Assign(SmallImage.Picture.Bitmap);
                  SaveToStream(UploadStream);
               finally
                 Free;
               end;
            finally
              SmallImage.Free;
            end;
          finally
            BigImage.Free;
          end;
        end;
    end;
end;
```

If you want to implement uploading of some bulky data to *password protected* directories, you must know that all uploading data is sent in the headers of HTTP request, before the HTTP server returns the notice that that login information is required. So you must specify Username and Password before uploading, otherwise, if you specifying login information in the OnPasswordRequest event handler, all request headers (all uploading files) will be sent more than once, every time when you set *TryAgain* parameter of OnPasswordRequest event handler to True.

See also

Upload method;
OnUploadProgress and OnPasswordRequest events.

9.31.4.17OnUploadProgress

Applies to

acHTTP component.

Declaration

```
type
  TacHTTPUploadProgressEvent = procedure(Sender: TObject;
              DataSize, BytesTransferred,
              ElapsedTime, EstimatedTimeLeft: Integer;
              PercentsDone: Byte; TransferRate: Single) of object;

procedure OnUploadProgress: TacHTTPUploadProgressEvent;
```

Description

The OnUploadProgress event occurs every time when acHTTP uploaded the block of data (size of block equal to TransferBufferSize) to the CGI application. Use it to display the progress of uploading files.

Write OnUploadProgress event handler to show the upload progress (*PercentsDone* parameter indicate the progress in percents), elapsed and estimated time before finishing download and speed

© 1998-2006, UtilMind Solutions®

of data transfer.

The acHTTP passes to the OnUploadProgress event handler following parameters:

Parameter	Meaning
DataSize	total size of data which we currently downloading, in bytes (if possible to determinate).
BytesRead	size of already received data, in bytes;
ElapsedTime	time elapsed from beginning of download (in seconds);
EstimatedTimeLeft	estimated time left before finishing of download (Formula: `X := FileSize / BytesRead * ElapsedTime - ElapsedTime`);
PercentsDone	progress in percents (0%..100%);
TransferRate	speed of data transfer (in Kb/s).

Example

Delphi:

```
procedure TForm1.acHTTP1UploadProgress(Sender: TObject; DataSize,
   BytesRead, ElapsedTime, EstimatedTimeLeft: Integer; PercentsDone:
   Byte; TransferRate: Single; Stream: TStream);
begin
  // progress bar position
  CurrentFileProgressBar.Position := PercentsDone;

  // file size
  FileSizeLabel.Caption :=
       Format('File size: %.1f Kb', [FileSize / 1024]);

  // downloaded (in Kb)
  DownloadedLabel.Caption :=
      Format('Downloaded: %.1f Kb:', [BytesRead / 1024]);

  // transfer rate
  TransRateLabel.Caption :=
       Format('Transfer rate: %.1f Kb/s', [TransferRate]);

  // estimated time left
  EstTimeLeftLabel.Caption :=
       Format('Estimated time left: %d:%.2d:%.2d',
          [EstimatedTimeLeft div 60 div 60,   // hours
           EstimatedTimeLeft div 60 mod 60,   // minutes
           EstimatedTimeLeft mod 60 mod 60]); // seconds
end;
```

C++ Builder:

```
void __fastcall TForm1::acHTTP1UploadProgress(TObject *Sender,
      AnsiString ContentType, int FileSize, int BytesRead,
      int ElapsedTime, int EstimatedTimeLeft,
      BYTE PercentsDone, float TransferRate, TStream *Stream)
{
  // progress bar position
  ProgressCurrentFile->Position = PercentsDone;

  // file size
  FileSizeLabel->Caption =
      Format("File size: %.1f Kb:",
```

© 1998-2006, UtilMind Solutions®

```
        ARRAYOFCONST(((float)FileSize / 1024)));

    // downloaded (in Kb)
    DownloadedLabel->Caption =
        Format("Downloaded: %.1f Kb:",
        ARRAYOFCONST(((float)BytesRead / 1024)));

    // transfer rate
    TransRateLabel->Caption =
        Format("Transfer rate: %.1f Kb/s",
        ARRAYOFCONST(((float)TransferRate)));

    // estimated time left
    EstTimeLeftLabel->Caption =
        Format("Estimated time left: %d:%.2d:%.2d",
        ARRAYOFCONST((EstimatedTimeLeft / 60 / 60,     // hours
                      EstimatedTimeLeft / 60 % 60,     // minutes
                      EstimatedTimeLeft % 60 % 60))); // seconds
}
```

See also
Upload method;
OnUploadFieldRequest and OnUploadCGITimeoutFailed events.

9.31.4.18 OnWaitTimeoutExpired

Applies to
acHTTP, acSendMail, acThread and acSQLQueryThread components.

Declaration

```
var
  TacThreadWaitTimeoutExpired = procedure(Sender: TObject; var
TerminateThread: Boolean) of object;

property OnWaitTimeoutExpired: TacThreadWaitTimeoutExpired;
```

Description
The OnWaitTimeoutExpired event occurs when the component could not complete HTTP request for the time interval specified in WaitTimeout property (when WaitTimeout is expired).

The *TherminateThread* is optional parameter (True by default), which allows to terminate, or prevent termination of thread in case if specified time interval is expired before completion of execution of the thread.

Notes
The WaitTimeout only works together with WaitThread property, only when it set to True.

See also
WaitThread and WaitTimeout properties;
Abort method.

9.31.5 Appendix: HTTP Status Codes

The following table contains the constants and corresponding values for the HTTP status codes returned by HTTP servers. Following constants defined in "WinInet.pas" module.

Constants

© 1998-2006, UtilMind Solutions®

// 1xx: Informational - Request received, continuing process

HTTP_STATUS_CONTINUE (100)
The request can be continued.

HTTP_STATUS_SWITCH_PROTOCOLS (101)
The server has switched protocols in an upgrade header.

// 2xx: Success - The action was successfully received, understood, and accepted

HTTP_STATUS_OK (200)
The request completed successfully.

HTTP_STATUS_CREATED (201)
The request has been fulfilled and resulted in the creation of a new resource.

HTTP_STATUS_ACCEPTED (202)
The request has been accepted for processing, but the processing has not been completed.

HTTP_STATUS_PARTIAL (203)
The returned meta information in the entity-header is not the definitive set available from the origin server.

HTTP_STATUS_NO_CONTENT (204)
The server has fulfilled the request, but there is no new information to send back.

HTTP_STATUS_RESET_CONTENT (205)
The request has been completed, and the client program should reset the document view that caused the request to be sent to allow the user to easily initiate another input action.

HTTP_STATUS_PARTIAL_CONTENT (206)
The server has fulfilled the partial GET request for the resource.

// 3xx: Redirection - Further action must be taken in order to complete the request

HTTP_STATUS_AMBIGUOUS (300)
The server couldn't decide what to return.

HTTP_STATUS_MOVED (301)
The requested resource has been assigned to a new permanent URI (Uniform Resource Identifier), and any future references to this resource should be done using one of the returned URIs.

HTTP_STATUS_REDIRECT (302)
The requested resource resides temporarily under a different URI (Uniform Resource Identifier).

HTTP_STATUS_REDIRECT_METHOD (303)
The response to the request can be found under a different URI (Uniform Resource Identifier) and should be retrieved using a GET HTTP verb on that resource.

HTTP_STATUS_NOT_MODIFIED (304)
The requested resource has not been modified.

HTTP_STATUS_USE_PROXY (305)
The requested resource must be accessed through the proxy given by the location field.

HTTP_STATUS_REDIRECT_KEEP_VERB (307)
The redirected request keeps the same HTTP verb. HTTP/1.1 behavior.

4xx: Client Error - The request contains bad syntax or cannot be fulfilled

© 1998-2006, UtilMind Solutions®

HTTP_STATUS_BAD_REQUEST (400)
The request could not be processed by the server due to invalid syntax.

HTTP_STATUS_DENIED (401)
The requested resource requires user authentication.

HTTP_STATUS_PAYMENT_REQ (402)
Not currently implemented in the HTTP protocol.

HTTP_STATUS_FORBIDDEN (403)
The server understood the request, but is refusing to fulfill it.

HTTP_STATUS_NOT_FOUND (404)
The server has not found anything matching the requested URI (Uniform Resource Identifier).

HTTP_STATUS_BAD_METHOD (405)
The HTTP verb used is not allowed.

HTTP_STATUS_NONE_ACCEPTABLE (406)
No responses acceptable to the client were found.

HTTP_STATUS_PROXY_AUTH_REQ (407)
Proxy authentication required.

HTTP_STATUS_REQUEST_TIMEOUT (408)
The server timed out waiting for the request.

HTTP_STATUS_CONFLICT (409)
The request could not be completed due to a conflict with the current state of the resource. The user should resubmit with more information.

HTTP_STATUS_GONE (410)
The requested resource is no longer available at the server, and no forwarding address is known.

HTTP_STATUS_LENGTH_REQUIRED (411)
The server refuses to accept the request without a defined content length.

HTTP_STATUS_PRECOND_FAILED (412)
The precondition given in one or more of the request header fields evaluated to false when it was tested on the server.

HTTP_STATUS_REQUEST_TOO_LARGE (413)
The server is refusing to process a request because the request entity is larger than the server is willing or able to process.

HTTP_STATUS_URI_TOO_LONG (414)
The server is refusing to service the request because the request URI (Uniform Resource Identifier) is longer than the server is willing to interpret.

HTTP_STATUS_UNSUPPORTED_MEDIA (415)
The server is refusing to service the request because the entity of the request is in a format not supported by the requested resource for the requested method.

HTTP_STATUS_RANGE_NOT_SATISFIABLE (416)
Requested Range not satisfiable.

HTTP_STATUS_RETRY_WITH (449)
The request should be retried after doing the appropriate action.

5xx: Server Error - The server failed to fulfill an apparently valid request

© 1998-2006, UtilMind Solutions®

HTTP_STATUS_SERVER_ERROR (500)
The server encountered an unexpected condition that prevented it from fulfilling the request.

HTTP_STATUS_NOT_SUPPORTED (501)
The server does not support the functionality required to fulfill the request.

HTTP_STATUS_BAD_GATEWAY (502)
The server, while acting as a gateway or proxy, received an invalid response from the upstream server it accessed in attempting to fulfill the request.

HTTP_STATUS_SERVICE_UNAVAIL (503)
The service is temporarily overloaded.

HTTP_STATUS_GATEWAY_TIMEOUT (504)
The request was timed out waiting for a gateway.

HTTP_STATUS_VERSION_NOT_SUP (505)
The server does not support, or refuses to support, the HTTP protocol version that was used in the request message.

9.31.6 HTTPReadString

Unit
acHTTP

Declaration
```
function HTTPReadString(const URL: String; Timeout: Integer = 0):
String;
```

Description
The HTTPReadString function provides extremely simple way to receive some data from the Web by HTTP protocol, without using acHTTP component and specifying its properties and handling the events. You just need to specify the URL (and optionally Timeout) and function will return downloaded data (or empty string, if remote host are unreachable or connection failed).

Return value is the downloaded string, or empty string, which means that download failed for some reason.

Parameters

URL	specifies the URL of document which you wish to download;
Timeout	optional parameter, which specifies the time-out for downloading (in milliseconds). If a connection request takes longer than this time-out value, the request is cancele and function returns empty string (which means that download failed). Zero timeout (0) means infinite, thus the function will try to download the data without any forced interrupts.

Remarks
1. Don't forget to add "acHTTP" into uses clause of your unit before using this function.
2. When you call this function, the execution of procedure from which the call are made, will suspended for some time, until the HTTP request will be complete or failed (it looks just like if you'd used acHTTP component with WaitThread property set to True).

Example
```
var
  Config: String;
begin
```

© 1998-2006, UtilMind Solutions®

```
  Config := HTTPReadString('www.yourdomain.com/some_configuration.ini');
  if Config <> '' then
   begin
    // deal with downloaded string
   end;


  // ANOTHER EXAMPLE: read the same config file with 10 seconds timeout
  Config := HTTPReadString('www.yourdomain.com/some_configuration.ini',
10000);
  if Config <> '' then
   begin
    // deal with downloaded string
   end;
end;
```

See also

Timeouts and WaitThread properties of acHTTP component.
FileReadString function from acUtils unit.

9.32 acHTTPDownloadQueue

9.32.1 TacHTTPDownloadQueue

Overview

The acHTTPDownloadQueue component

How to use?

See also

acHTTP and acThread components.

9.33 acIESniffer

9.33.1 TacIESniffer

Overview

The acIESniffer monitors all running instances of Internet Explorer (or just some certain browser window(s)), sniffs the URL address from address line and gives an access to all their properties, methods and events (without implementing/installing ANY Browser Helper Objects). The component can work either in stand-alone Delphi/BCB application or can be put to Explorer Toolbar written with Delphi/BCB. The component allows:

- list the URLs which currently are available in the address lines of each browser window, and to be notified when the address changes;
- detect when new Internet Explorer window appears and when user close the window;
- hook each event of the Internet Explorer, modify its properties and call their methods, just like you are using usual TWebBrowser component;
- retrieve and modify the content of each page (its text, tags, links, images etc), for example you may highlight some text, remove or change the text, tags etc;
- automatically fill the Web forms (with acIESnifferAutoFillUserInfo component);
- redirect navigation to different URL when some keywords are detected in the address line;
- automatically redirect searches from default MSN to another specified search engine (it works when user enters search terms right in address line without specifying correct address);
- prevent popup windows to be opened, and so on…

© 1998-2006, UtilMind Solutions®

This component available only in Delphi/BCB 5 and higher versions and works with Internet Explorer 4 and higher (unfortunately IWebBrowser2 and DWebBrowserEvents2 interfaces does not supported in earlier versions).

How does it works?

The IE Sniffer contantly monitors the shell for new instances of Internet or Windows explorer (with the interval specified in MonitorInterval property). When it detects new instance of Explorer, it triggers OnWindowLoad event, adds the detected instance to the internal list, and starts hooking all events of that browser, by connecting to its IWebBrowser2 interface *(Note: the component does NOT hooks events of Windows Explorer, you can receive events only from IE)*. When user close the Explorer window, it triggers OnWBQuit (window is about to be closed), then the OnWindowUnload event (window are closed and all its handles already destroyed).

How to use?

First drop the component onto your form and specify the interval between checking for new Explorer windows to MonitorInterval property. If you just want to detect when the new URL appears in some browser window, and to be notified when the address changes — write OnURLChange event handler. To be notified when the new Explorer window opens — use OnWindowLoad event, to take some specific actions when user closes the window — write OnWindowUnload event handler, or use OnWBQuit event instead when the Internet Explorer window is only about to be closed. If you want to detect only windows with "http" prefix (with content received by HTTP or HTTPS protocols — set SniffWithHTTPrefixOnly property to True). Also you can always receive the full list of URLs which currently are available in all Explorer windows by reading the URLs property.

If you don't need to hook events from ALL instances of Internet Explorer and just want to receive events from some certain browser window — set Active property to False (component will not detect IE windows automatically), and add some certain browser window for monitoring with AddBrowser method.

If you want to perform some more complicated tasks (like grabbing the content of browser window, or modifying it, or something else that can be done with browser), you should access its IWebBrowser2 interface. To access it, you can use the numerous events (each event passes the pointer to IWebBrowser2 instance), or use IEList property to enumerate each instance of Internet Explorer and play with their properties and methods. If you wish to replace or highlight some text when the page is downloaded — use MarkText or ReplaceText methods in the OnWBDownloadComplete and OnWindowLoad event handlers.

Here is simple example which demonstrates how to highlight some word on the downloaded HTML document:

```
procedure TForm1.acIESniffer1WBDownloadComplete(Sender: TObject;
  const URL: String; const Browser: IWebBrowser2);
begin
  acIESniffer1.MarkText(Browser, 'Delphi', clBlack, clYellow);
end;
```

To clear the marks — use ClearMarks method:

```
procedure TForm1.ClearBtnClick(Sender: TObject);
begin
  acIESniffer1.ClearMarks(Browser);
end;
```

If you wish to redirect the navigation to another URL when some keywords are detected in the address line you can use following example:

```
procedure TBandForm.acIESniffer1WBBeforeNavigate2(Sender: TObject;
  const URL: String; const Browser: IWebBrowser2; const pDisp:
```

© 1998-2006, UtilMind Solutions®

```
IDispatch;
  var NewURL, Flags, TargetFrameName, PostData, Headers: OleVariant;
  var Cancel: WordBool);
begin
  if Pos('porn', LowerCase(NewURL)) <> 0 then
   begin
    Cancel := True;
    Browser.Navigate('http://www.disney.com', Flags, TargetFrameName,
PostData, Headers);
   end;
end;
```

Here is another example how to elliminate all popup windows before they will be displayed. This is VERY simple example, but it works and kills ALL popup windows (which don't have the toolbar):

```
  procedure TBandForm.acIESniffer1WBBeforeNavigate2(Sender: TObject;
    const URL: String; const Browser: IWebBrowser2; const pDisp:
  IDispatch;
    var NewURL, Flags, TargetFrameName, PostData, Headers: OleVariant;
    var Cancel: WordBool);
  begin
    Cancel := Browser.Toolbar = 0;
  end;
```

Also if your popup killer software operates with "white" and "black" lists, you can additionally check the URL parameter and if it contains some unwanted text (domain name or keyword), you can allow or disallow the navigation accordingly.
Additionally, you can check the state of Ctrl key (like Google toolbar does) and always allow the navigation when Ctrl key pressed.

Note for C++ Builder developers

When you will build the project with IESniffer, you may get following compiler error:
[C++ Error] SHDocVw.hpp(893): E2293) expected, at the following line:

```
/* TWinControl.CreateParented */ inline __fastcall TWebBrowser(HWND ParentWindow)
: Olectrls::TOleControl(ParentWindow) { }
```

Please manually edit the SHDocVw.hpp and change this line to

```
/* TWinControl.CreateParented */ inline __fastcall TWebBrowser(HANDLE
ParentWindow) : Olectrls::TOleControl(ParentWindow) { }
```

Code example

In Delphi: http://www.appcontrols.com/demos/aciesnifferdemo-delphi.zip
Compiled executable: http://www.appcontrols.com/download/exe/IESnifferDemo.exe

See also

acIESnifferAutoFillUserInfo and acHTMLViewer components.

9.33.2 Properties

9.33.2.1 Active

Applies to

acIESniffer, acAwayTimer, acCursorLocker, acFormBackground, acFormHelp, acFormMagnet and acFormSystemMenu components.

Declaration

```
property Active: Boolean; // True by default
```

© 1998-2006, UtilMind Solutions®

Description

The Active property controls whether the acIESniffer is active to detect when the new instances of Internet or Windows Explorer appears and disappears (when user opens and closes the Internet or Windows Explorer windows).

Set Active property to True to enable the timer and check the shell for instances of Windows or Internet Explorer, and to trigger OnWindowLoad event when new window appears and OnWindowUnload event when user closes the Explorer window.

The detection are performed with intervals specified in MonitorInterval property.

If you don't want to detect new instances of Internet Explorer on timer-based schedule, you can just periodically call Refresh method and set Active to False.
The Refresh method detects each instance of Internet Explorer and adds it to the internal list. Alternatively, if you don't need to operate with ALL instances of Explorer, you can use AddBrowser method to add some certain intance of IE to the list and hook all its events.

See also

MonitorInterval property;
Refresh method.

9.33.2.2 FormAutoFill

Applies to

acIESniffer component.

Declaration

```
property FormAutoFill: TacIESnifferAutoFill;
```

Description

The FormAutoFill property used to specify custom component which allows to fill the Web forms.

Currently AppControls includes only one "form filler", acIESnifferAutoFillUserInfo component. Drop that component on your form together with acIESniffer and point FormAutoFill to the *acIESnifferAutoFillUserInfo* component, if you want to automatically fill the Web forms with some basic user details, like name, phone and address.

See also

acIESnifferAutoFillUserInfo component.

9.33.2.3 IEList

Applies to

acIESniffer component.

Declaration

```
property URLList: TacIEList;
```

Description

The URLList property is the list of containers for each detected instance of Explorer and their IWebBrowser2 interface. This is just descendant of usual TList which contains TacIEHelper objects. Here is how TacIEHelper object are introduced in acIESniffer unit:

```
type
  TacIEHelper = class(TInterfacedObject, IUnknown, IDispatch)
  public
    property Browser: IWebBrowser2; // pointer to IWebBrowser2
```

© 1998-2006, UtilMind Solutions®

```
  interface
      property IsCompleted: Boolean;  // whether ReadyState of Browser
  is in READYSTATE_COMPLETE
      property URL: String;           // current URL
      property ZoomFactor: Integer;   // zoom factor of HTMLViewer, 100
  by default (100%)
    end;
```

Example (demonstrates how to fill some ListView with URLs and the page titles grabbed from each Explorer window)

```
procedure TForm1.RefreshActiveInstancesInListView;
var
  I: Integer;
  ListItem: TListItem;
begin
  with ListView1, Items do
   try
     BeginUpdate;
     Clear;

     I := acIESniffer1.IEList.Count;
     if I <> 0 then
      for I := 0 to I - 1 do
       with acIESniffer1.IEList[I] do
        try
          ListItem := Items.Add;
          ListItem.Caption := URL;
          ListItem.SubItems.Add(Browser.LocationName);
        except
        end;
   finally
     EndUpdate;
   end;
end;
```

See also

SniffWithHTTPPrefixOnly property;
OnWindowLoad and OnWindowUnload events.

9.33.2.4 MonitorInterval

Applies to

acIESniffer and acAwayTimer components.

Declaration

```
property MonitorInterval: Integer; // 2 seconds by default
```

Description

The MonitorInterval property specifies the time intervals, in units of milliseconds, between each automatic call of Refresh method, which detects all new Internet Explorer windows.

By default, the MonitorInterval set to 2000, that means that component is checking the shell for new Explorer windows every 2 seconds. Feel free to increase or decrease this value, however it's not recommended to decrease it less than 500 milliseconds, because it can lead to overloading of CPU.

See also

© 1998-2006, UtilMind Solutions®

Active property.

9.33.2.5 SearchBar

Applies to
acIESniffer component.

Declaration
```
type
  TacIESnifferSearchBar = class
  published
    property Enabled: Boolean stored False; // basically run-time only
    property URL: String;
  end;

property SearchBar: TacIESnifferSearchBar;
```

Description
The SearchBar property used to specify custom URL for search panel (which visible at the left side of Internet Explorer window, when user clicks "Search" button on the toolbar).

The *Enabled* is dynamic property which should be specified in run-time only! It determines whether the search bar of Internet Explorer currently points to specified URL. Set *Enabled* to True at run-time to specify the custom page for search bar, or set it to False, to restore default Windows Search Assistant.

The *URL* property specifies the address to the web page which can be used instead of default Windows Search Assistant. The custom page for search bar can looks like http://www.google.com/ie.

See also
SearchRedirect property.

9.33.2.6 SearchRedirect

Applies to
acIESniffer component.

Declaration
```
type
  TacIESnifferSearchRedirect = class
  published
    property Enabled: Boolean;
    property RedirectURL: String;
  end;

property SearchRedirect: TacIESnifferSearchRedirect;
```

Description
The SearchRedirect property used to specify the custom search engine for default searches (when user specifies search terms in the address line instead of correct address).

The *Enabled* property specifies whether the acIESniffer should redirect default searches to another URL when it notice that browser is about to navigate somewhere like "http://auto.search.msn.com/response.asp" and redirect the search to the URL specified in *RedirectURL* property. Set Enabled to True, if you would like to use custom search engline, or leave it False, if you're happy with default MSN engine (or some another search engine installed as

© 1998-2006, UtilMind Solutions®

Browser Helper Object plug-in to IE).

The *RedirectURL* property specifies the URL to some custom search engline interface. The property can contain string like "http://www.yoururl.com/**%s**", where **%s** will be automatically replaced by the search terms typed by user in the address line.

See also
SearchBar property;
acIESnifferAutoFillUserInfo component.

9.33.2.7 SniffWithHTTPPrefixOnly

Applies to
acIESniffer component.

Declaration
```
property SniffWithHTTPPrefixOnly: Boolean;
```

Description
The SniffWithHTTPPrefixOnly property controls whether the component should allow detect only that windows of Internet Explorer which have 'http' prefix in its address line.

Set SniffWithHTTPPrefixOnly to True, to prevent hooking events when user surfing local intranet pages or just use Windows Explorer instead of Internet, and want to retrieve content and events only of those IE instances which currently displays the Web content received by HTTP or HTTPS protocols. Otherwise, it will detect all addresses, event when user using Windows Explorer to browse local directories (though, the component is unable to hook the events of Windows Explorer it gives access only to IE).

See also
URLs property;
OnWindowLoad and OnWindowUnload events.

9.33.2.8 URLs

Applies to
acIESniffer component.

Declaration
```
property URLs: TStrings;
```

Description
The URLs property used to sniff the text from address line of each instance of Internet Explorer (and even Windows Explorer, if SniffWithHTTPPrefixOnly is False).

When you reading URLs property, you don't need to call Refresh method, it's called before sniffing

See also
SniffWithHTTPPrefixOnly property;
OnWindowLoad and OnWindowUnload events.

© 1998-2006, UtilMind Solutions®

9.33.3 Methods

9.33.3.1 AddBrowser

Applies to
acIESniffer component.

Declaration
```
function AddBrowser(Browser: IWebBrowser2): Boolean;
```

Description
The AddBrowser method used to add some certain *Browser* window to the internal list of component, and want to hook all events from that certain *Browser*.

The AddBrowser should be used only if Active property is False, so you don't want to detect the IE windows automatically and just want to receive events from some certain *Browser* (for example if you write the toolbar for Internet Explorer and want to receive events only from that browser inside the IE window below your toolbar).

Don't worry about adding one Browser instance several times, function will add only ONE instance of browser, and will return False if the *Browser* already exists in the internal list of component. Also you don't need to remove browser from list, component will handle this automatically upon receiving OnWBQuit event (when IE window are closed).

Note that if you want to operate with only ONE browser, do not call Refresh method (which detects IE instances automatically and add all them to the internal list to hook their events), plus make sure that Active property is False (so Refresh method will not called automatically on timer-based schedule).

Example
```
procedure TBandForm.NavigateFromBand(const URL: String);
var
  _Url, X: OleVariant;
begin
  acIESniffer1.AddBrowser(IE);
  _Url := Url; X := 0;
  IE.Navigate(Url, X, X, X, X);
end;

// another example how to add TWebBrowser to the internal list of
IESniffer
procedure TForm1.Button1Click(Sender: TObject);
begin
  IESniffer1.AddBrowser(WebBrowser1.ControlInterface);
end;
```

See also
Active and IEList properties;
Refresh method;
OnWindowLoad, OnWindowUnload, OnWBDownloadComplete and OnWBQuit events.

9.33.3.2 ClearMarks

Applies to
acIESniffer component.

© 1998-2006, UtilMind Solutions®

Declaration

```
function ClearMarks(Browser: IWebBrowser2; const MarkID: String =
'IESnifferMark'): Integer;
```

Description

The ClearMarks method clear the marks from keywords, which has been highlighted with MarkText method.

When you call ClearMarks you just need to specify the Browser window and *MarkID*, which should be the same which has been used when you called MarkText method.

Example

```
procedure TBandForm.SetHighlighter(Value: Boolean);
begin
  if Value then
    acIESniffer1.MarkText(IE, FKeywords, clBlack, clYellow)
  else
    acIESniffer1.ClearMarks(IE);
end;
```

See also

MarkText and ReplaceText methods.

9.33.3.3 CloseBrowsers

Applies to

acIESniffer component.

Declaration

```
procedure CloseBrowsers(const URL: String = 'about:blank');
```

Description

The CloseBrowsers method closes ALL detected Internet Explorer windows which currently displaying the location specified in *URL* parameter.

Calling of this method without any parameters will close all empty browser windows (with `about:blank` text in the address line).

See also

MarkText and ReplaceText methods.

9.33.3.4 MarkText

Applies to

acIESniffer component.

Declaration

```
function TacIESniffer.MarkText(Browser: IWebBrowser2; const Text:
String;
  FontColor: TColor = clBlack; BackgroundColor: TColor = clYellow;
  WholeWords: Boolean = False; MatchCase: Boolean = False; ScanFrames:
Boolean = True;
  const MarkID: String = 'IESnifferMark'): Integer; // returns number of
marked words
```

Description

The MarkText method searches for all occurences of the *Text* on the Web page and marks it with

© 1998-2006, UtilMind Solutions®

specified foreground (*FontColor* parameter) and background (*BackgroundColor* parameter) colors.

Parameter	Meaning
Browser	is the IWebBrowser2 interface with some text which should be marked.
Text	is the word or phrase which should be marked in the HTML document.
FontColor	specifies the font color to mark the found text.
BackgroundColor	specifies the background color to highlight the text.
WholeWords	whether the *Text* should match whole words only.
MatchCase	whether the *Text* should match case.
ScanFrames	whether to scan subframes of the web page (True by default).
MarkID	identifier of marked text which must be used if you plan to clear marks. (ClearMarks method can clear the marks from keywords with specified *MarkID*).

Function returns the number of marked text fragments, or 0 if specified Text was not found on the page.

Example

```
procedure TForm1.acIESniffer1WBDownloadComplete(Sender: TObject;
  const URL: String; const Browser: IWebBrowser2);
begin
  // highlights AppControls keyword with yellow color
  acIESniffer1.MarkText(Browser, 'AppControls', clBlack, clYellow);
end;
```

See also

ReplaceText method;
OnWindowLoad and OnWBDownloadComplete events.

9.33.3.5 Refresh

Applies to

acIESniffer, acProcessList and acConnectionList components (used to refresh the content of the internal data list).

Declaration

```
procedure Refresh;
```

Description

The Refresh method refreshes the internal list of active Internet Explorer windows.

You can use Refresh method to detect new instances of Internet Explorer if you don't want to detect it automatically on timer-based schedule (if Active property is False).

Also, to refresh the list of active IE windows, you can simply read the URLs property, it will call Refresh method automatically.

Alternatively, if you don't need to operate with ALL instances of Explorer, you can use AddBrowser method to add some certain intance of IE to the list and hook all its events.

See also

Active, MonitorInterval and URLs properties;

© 1998-2006, UtilMind Solutions®

AddBrowser method;
OnWindowLoad and OnWindowUnload events.

9.33.3.6 ReplaceText

Applies to
acIESniffer component.

Declaration

```
function TacIESniffer.ReplaceText(Browser: IWebBrowser2; const OldText,
NewText: String;
  WholeWords: Boolean = False; MatchCase: Boolean = False; ScanFrames:
Boolean = True): Integer; // returns number of marked words
```

Description
The ReplaceText method searches for all occurences of the *OldText* on the Web page and replaces it with *NewText* (can contain HTML tags).

Parameter	Meaning
Browser	is the IWebBrowser2 interface with some text which should be marked.
OldText	is the word or phrase which should be replaced in the HTML document by *NewText*.
NewText	the new word or phrase to which the *OldText* should be replaced to.
WholeWords	whether the *Text* should match whole words only.
MatchCase	whether the *Text* should match case.
ScanFrames	whether to scan subframes of the web page (True by default).

Function returns the number of marked text fragments, or 0 if specified Text was not found on the page.

Example

```
procedure TForm1.IESniffer1WBDownloadComplete(Sender: TObject;
  const URL: String; const Browser: IWebBrowser2);
begin
  // change all occurances of word "Delphi" to bold "Delphi Rules!" ;-)
   IESniffer1.ReplaceText(Browser, 'Delphi', '<b>Delphi Rules!</b>');
end;
```

See also
MarkText method;
OnWindowLoad and OnWBDownloadComplete events.

9.33.4 Events

9.33.4.1 OnURLChange

Applies to
acIESniffer component.

Declaration

```
type
  TacIESnifferEvent = procedure(Sender: TObject; const URL: String;
const Browser: IWebBrowser2) of object;
```

© 1998-2006, UtilMind Solutions®

```
property OnURLChange: TacIESnifferEvent;
```

Description

The OnURLChange event occurs when the URL in the address line of Explorer changes.

Use OnURLChange event to be notified when user viewing the content from some particular location and grab that *URL* address from certain *Browser*.

Normally the OnURLChange event occurs at once after detecting the new instance of Internet Explorer (after OnWindowLoad), to notify your application about new URL address, and when the user navigates not another location (before OnWBTitleChange and OnWBNavigateComplete2 occur, if the URL is different than before).

See also

SniffWithHTTPPrefixOnly property;
OnWindowLoad, OnWBTitleChange, OnWBBeforeNavigate2, OnWBNavigateComplete2, OnWBDownloadComplete and OnWBDocumentComplete events.

9.33.4.2 OnWBBeforeNavigate2

Applies to

acIESniffer component.

Declaration

```
type
  TacIEWebBrowserBeforeNavigate2 = procedure(Sender: TObject; const URL:
String; const Browser: IWebBrowser2;
    const pDisp: IDispatch; var NewURL: OleVariant; var Flags:
OleVariant;
    var TargetFrameName: OleVariant; var PostData: OleVariant;
    var Headers: OleVariant; var Cancel: WordBool) of object;

property OnWBBeforeNavigate2: TacIEWebBrowserBeforeNavigate2;
```

Description

The OnWBBeforeNavigate2 event occurs immediately before some instance of Internet Explorer navigates to a new resource.

Write an OnWBBeforeNavigate2 event handler to redirect or cancel a change to a different URL. This event may occur as the result of a call to the Navigate or Navigate2 method of *Browser* control, or the user clicking a link.

The *URL* parameter is the current location shown in the address line of browser window.

Browser is the pointer to IWebBrowser2 control which performs the current operation (you can use its properties and methods inside the event handler).

pDisp is the Automation interface of the Web browser control specified by *Browser*.

NewURL is the Uniform Resource Locator of the resource the Web browser is looking up. Change this value to redirect the navigation operation to a different resource.

Flags is not currently used.

TargetFrameName is the name of the frame in which the resource will be displayed, or NULL if the resource should not be displayed in a named frame. Change this value to change where the resource is displayed. See the Navigate method of IWebBrowser2 interface for a list of possible values.

© 1998-2006, UtilMind Solutions®

PostData contains the data sent to the server when the underlying operation is an HTTP post message. The event handler can change this value before it is sent to the target URL. The PostData can be extracted to normal string using following code:

```
function PostDataToStr(PostData: WideString): String;
begin
  SetLength(Result, Length(PostData) shl 1 - 1); // -1 tailing #0
  Move(PostData[1], Result[1], Length(Result));
end;
```

Headers contains any headers sent to the servers when the URL represents an HTTP message. HTTP headers specify such things as the intended action required of the server, the type of data, and so on. (See TWebRequest object, whose properties represent many of the more common headers). The event handler can change this value before it is sent to the target URL.

Cancel determines whether the Web browser looks up the specified resource after the event handler exits. Change *Cancel* to True to cancel the navigation operation.

Example

```
// demonstrates how to redirect navigation to another URL
procedure TBandForm.acIESniffer1WBBeforeNavigate2(Sender: TObject;
  const URL: String; const Browser: IWebBrowser2; const pDisp:
IDispatch;
  var NewURL, Flags, TargetFrameName, PostData, Headers: OleVariant;
  var Cancel: WordBool);
begin
  if Pos('porn', LowerCase(NewURL)) <> 0 then
   begin
    Cancel := True;
    Browser.Navigate('http://www.disney.com', Flags, TargetFrameName,
PostData, Headers);
   end;
end;
```

See also

OnWBNavigateComplete2 event.

9.33.4.3 OnWBCommandStateChange

Applies to

acIESniffer component.

Declaration

```
type
  TacIEWebBrowserCommandStateChange = procedure(Sender: TObject; const
URL: String; const Browser: IWebBrowser2;
    Command: Integer; Enable: WordBool) of object;

property OnWBCommandStateChange: TacIEWebBrowserCommandStateChange;
```

Description

The OnWBCommandStateChange event occurs when the ability to execute certain IWebBrowser2 methods changes.

Write an OnWBCommandStateChange to update any controls in the application whose state depends on the ability to execute the Web browser's methods.

© 1998-2006, UtilMind Solutions®

Browser is the Web browser control whose capabilities have changed.

Command indicates what has changed. The following table lists the possible values:

Constant	Value	Meaning
CSC_UPDATECOMMANDS	-1	Any change not covered by the other constants. The application must check the properties of the Web browser to update its controls. For example, the event handler might check the Busy property to update a Stop button.
CSC_NAVIGATEFORWARD	1	The history list changed the ability of the GoForward method to navigate to a new URL. The Enable parameter indicates whether GoForward now navigates to a new URL (True), or not (False).
CSC_NAVIGATEBACK	2	The history list changed the ability of the GoBack method to navigate to a new URL. The Enable parameter indicates whether GoBack now navigates to a new URL (True), or not (False).

Enable indicates whether the CSC_NAVIGATEFORWARD or CSC_NAVIGATEBACK commands should now be enabled (True), or not (False).

See also
OnWBPropertyChange and OnWBProgressChange events.

9.33.4.4 OnWBDocumentComplete

Applies to
acIESniffer component.

Declaration

```
type
  TacIEWebBrowserDocumentComplete = procedure(Sender: TObject; const
URL: String; const Browser: IWebBrowser2;
    const pDisp: IDispatch; var NewURL: OleVariant) of object;

property OnWBDocumentComplete: TacIESnifferEvent;
```

Description
The OnWBDownloadComplete Occurs when the document being navigated to reaches ReadyState_Complete.

Write an OnWBDocumentComplete event handler to take specific action when a frame or document is FULLY loaded into the Web browser. For a document without frames, this event occurs once when the document finishes loading. On a document containing multiple frames, this event occurs once for each frame. When the multiple-frame document finishes loading, the Web browser fires the event one final time.

Browser is the IWebBrowser2 that is loading the document.

pDisp is the Automation interface of the top-level frame or browser. When loading a document without frames, *pDisp* is the interface of the Web browser. When loading a document with multiple frames, this is the interface of the containing frame, except for the very last time the event occurs, when it is the interface of the Web browser.

© 1998-2006, UtilMind Solutions®

NewURL is the URL, UNC file name, or PIDL that to which the Web browser navigated. This URL can be different from the URL to which the browser was told to navigate. For example, the browser may have been redirected by the target resource or by an OnWBBeforeNavigate2 event handler. In addition, the value of URL is the canonicalized and qualified URL: for example, if an application specified a URL of "www.appcontrols.com" in a call to the Navigate or Navigate2 method of *Browser* object, the URL in the OnWBDocumentComplete event handler is "http://www.appcontrols.com/".

If you wish to modify downloaded page — use OnWBDownloadComplete event, to be notified when the page is refreshed by user.

See also
OnWBBeforeNavigate2, OnWBDownloadBegin, OnWBDownloadComplete and OnWBNavigateComplete2 events.

9.33.4.5 OnWBDownloadBegin

Applies to
acIESniffer component.

Declaration
```
type
  TacIESnifferEvent = procedure(Sender: TObject; const URL: String;
const Browser: IWebBrowser2) of object;

property OnWBDownloadBegin: TacIESnifferEvent;
```

Description
The OnWBDownloadBegin event occurs when the Web browser starts downloading a document.

Write an OnWBDownloadBegin event handler to take specific action after the IWebBrowser2 has located a document and immediately before it starts downloading the document. For example, use the OnWBDownloadBegin event to launch an animation control the represents downloading or a progress bar that is updated by an OnWBProgressChange event handler. The control can then be stopped in an OnWBDownloadComplete event handler.

Note
To take specific action when the Web browser looks up the resource, rather than when it begins downloading, use the OnWBBeforeNavigate2 event. OnWBDownloadBegin occurs shortly after OnWBBeforeNavigate2.

See also
OnWBBeforeNavigate2, OnWBDocumentComplete, OnWBDownloadComplete, OnWBNavigateComplete2 and OnWBProgressChange events.

9.33.4.6 OnWBDownloadComplete

Applies to
acIESniffer component.

Declaration
```
type
  TacIESnifferEvent = procedure(Sender: TObject; const URL: String;
const Browser: IWebBrowser2) of object;

property OnWBDownloadComplete: TacIESnifferEvent;
```

© 1998-2006, UtilMind Solutions®

Description

The OnWBDownloadComplete event occurs when a navigation operation finishes, is halted, or fails.

Write an OnWBDownloadComplete event handler to take specific action after the Web browser stops a downloading operation. For example, use the OnWBDownloadComplete event to stop an download indication that is started in an OnWBDownloadBegin event handler.

You can use this event to read or modify the content of downloaded Web page, or just highlight some keywords on the page.

Note

Unlike the OnWBNavigateComplete2 event, OnWBDownloadComplete occurs even if the Web browser does not successfully navigate to an URL.

Example 1 (demonstrates how to read all content of downloaded HTML page)

```
uses MSHTML; // introduces IHTMLDocument2 interface

procedure TForm1.acIESniffer1WBDownloadComplete(Sender: TObject;
  const URL: String; const Browser: IWebBrowser2);
var
  doc: IHTMLDocument2;
  Collection: IHTMLElementCollection;
  Element: IHTMLElement;
  HTMLPage: String;
  PlainText: String;
begin
  try
    doc := (Browser.Document as IHTMLDocument2);
    Collection := doc.all;
    Collection := Collection.Tags('BODY') as IHTMLElementCollection;
    Element := Collection.Item(NULL, 0) as IHTMLElement;

    HTMLPage := Element.OuterHTML; // read the HTML page
    PlainText := Element.OuterText; // or just plain text
  except
  end;
end;
```

Example 2 (demonstrates how to highlight some text keywords in the downloaded page)

(Alternatively you can use MarkText or ReplaceText methods!)

```
uses MSHTML; // introduces IHTMLDocument2 interface

procedure TForm1.acIESniffer1WBDownloadComplete(Sender: TObject;
  const URL: String; const Browser: IWebBrowser2);
var
  Doc: IHTMLDocument2;
  BodyElement: IHTMLBodyElement;
  TextRange: IHTMLTxtRange;
  SearchFlag: Integer;
begin
  try
    //
http://msdn.microsoft.com/library/default.asp?url=/workshop/author/dhtml
/reference/methods/findtext.asp
    SearchFlag := 2 + 4; // 2 = Whole words only, 4 = Match case
```

© 1998-2006, UtilMind Solutions®

```
      Doc := Browser.Document as IHTMLDocument2;
      BodyElement := Doc.body as IHTMLBodyElement;
      TextRange := BodyElement.CreateTextRange;
      try
        while TextRange.findText('Delphi', MaxInt, SearchFlag) do
         TextRange.pasteHTML('Delphi Rules!');
      finally
        TextRange._Release;
      end;
    except
    end;
end;
```

See also

OnWBDocumentComplete, OnWBDownloadBegin and OnWBNavigateComplete2 events; MarkText and ReplaceText methods.

9.33.4.7 OnWBFileDownload

Applies to

acIESniffer component.

Declaration

```
type
  TacIEWebBrowserOnFileDownload = procedure(Sender: TObject;
    const Browser: IWebBrowser2; var Cancel: WordBool) of object;

property OnWBFileDownload: TacIEWebBrowserOnFileDownload;
```

Description

The OnWBFileDownload event occurs before displaying the FileDownload dialog box. You can cancel downloading setting the *Cancel* parameter to False.

This event is useful if you would like to implement your own download manager, without built-in IE download dialogs. To hook the location of the file which is about to be downloaded — write OnWBBeforeNavigate2 event handler.

This event supported by IE5.5+ only! In lower versions please use OnWBBeforeNavigate2 event and check whether the location have *.zip, *.exe or some another extension supported by your download manager and set *Cancel* parameter to False.

See also

OnWBBeforeNavigate2 event.

9.33.4.8 OnWBFullScreen

Applies to

acIESniffer component.

Declaration

```
type
  TacIEWebBrowserOnFullScreen = procedure(Sender: TObject; const URL:
String; const Browser: IWebBrowser2;
    FullScreen: WordBool) of object;

property OnWBFullScreen: TacIEWebBrowserOnFullScreen;
```

© 1998-2006, UtilMind Solutions®

Description
The OnWBFullScreen event occurs when the Internet Explorer become maximized to full screen or restored from maximized state.

See also
OnWBVisible, OnWBToolBar, OnWBStatusBar, OnWBMenuBar and OnWBTheaterMode events.

9.33.4.9 OnWBMenuBar

Applies to
acIESniffer component.

Declaration
```
type
  TacIEWebBrowserOnMenuBar = procedure(Sender: TObject; const URL:
String; const Browser: IWebBrowser2;
    MenuBar: WordBool) of object;

property OnWBMenuBar: TacIEWebBrowserOnMenuBar;
```

Description
The OnWBMenuBar event occurs when the Internet Explorer shows or hides its menu bar.

See also
OnWBVisible, OnWBToolBar, OnWBStatusBar, OnWBTheaterMode and OnWBFullScreen events.

9.33.4.10 OnWBNavigateComplete2

Applies to
acIESniffer component.

Declaration
```
type
  TacIEWebBrowserNavigateComplete2 = procedure(Sender: TObject; const
URL: String; const Browser: IWebBrowser2;
    const pDisp: IDispatch; var NewURL: OleVariant) of object;

property OnWBNavigateComplete2: TacIEWebBrowserNavigateComplete2;
```

Description
The OnWBNavigateComplete2 event occurs immediately after the Web browser successfully navigates to a new location.

Write an OnWBNavigateComplete2 event handler to take specific action when the Web browser successfully navigates to a new resource. The event can occur before the document is fully downloaded, but when it occurs at least part of the document must be received and a viewer for the document created.

Browser is the IWebBrowser2 interface that navigated to the new resource.

pDisp is the Automation interface of the browser.

URL is the URL, UNC file name, or PIDL that to which the Web browser navigated. This URL can be different from the URL to which the browser was told to navigate. For example, the browser may have been redirected by the target resource or by an OnWBBeforeNavigate2 event handler. In

© 1998-2006, UtilMind Solutions®

addition, the value of URL is the canonicalized and qualified URL: for example, if an application specified a URL of "www.appcontrols.com" in a call to the Navigate or Navigate2 method of the IWebBrowser2 interface, the URL in the OnWBDocumentComplete event handler is "http://www.appcontrols.com/".

Note

Unlike the OnWBDownloadComplete event, OnWBNavigateComplete2 does not occur if the operation is not successful.

The OnURLChange event will occur if the URL in the browser address line has been changed.

See also

OnWBBeforeNavigate2, OnWBDocumentComplete, OnWBDownloadBegin and OnWBDownloadComplete events.

9.33.4.11 OnWBNewWindow2

Applies to

acIESniffer component.

Declaration

```
type
  TacIEWebBrowserNewWindow2 = procedure(Sender: TObject; const URL:
String; const Browser: IWebBrowser2;
    var ppDisp: IDispatch; var Cancel: WordBool) of object;

property OnWBNewWindow2: TacIEWebBrowserNewWindow2;
```

Description

The OnWBNewWindow2 event occurs when a new, hidden, non-navigated Web browser window is needed.

Write an OnWBNewWindow2 to take specific action immediately before the Web browser creates a new window for displaying a resource. This window may be needed because the user shift-clicked on a link, the user right-clicked on a link and chose "open in new window", the frame for the target URL does not yet exist, or the Navigate or Navigate2 methods of IWebBrowser2 interface requested that the target document appear in a new window.

Browser is the IWebBrowser2 interface that needs a new window to display its target resource.

ppDisp optionally returns the interface for a newly created, hidden, TWebBrowser component that can act as the new window. The Web browser configures this component and navigates to the target URL, starting with an OnWBBeforeNavigate2 event. If the event handler does not create a component and return its interface as the ppDisp parameter, the Web browser generates a top-level window as a separate, nonhosted process.

Cancel allows the event handler to block the creation of a new window. When the event handler sets *Cancel* to True, the Web browser tries to display the target resource in its current window, starting with an OnWBBeforeNavigate2 event.

Note

The event handler should not return a value for *ppDisp* when setting *Cancel* to True.

See also

OnWBBeforeNavigate2 event.

© 1998-2006, UtilMind Solutions®

9.33.4.12OnWBProgressChange

Applies to
acIESniffer component.

Declaration

```
type
  TacIEWebBrowserProgressChange = procedure(Sender: TObject; const URL:
String; const Browser: IWebBrowser2;
    Progress, ProgressMax: Integer) of object;

property OnWBProgressChange: TacIEWebBrowserProgressChange
```

Description
The OnWBProgressChange event occurs when the progress of a download operation is updated.

Write an OnWBProgressChange event handler to provide visual feedback about the download process. For example, an OnWBProgressChange event handler can update a TProgressBar component or display the number of bytes downloaded so far.

Browser is the IWebBrowser2 interface that is in the process of downloading a document.

Progress indicates how much of the document has already been downloaded, on a scale of 0 to ProgressMax. When Progress is –1, the operation is finished.

ProgressMax indicates the total size of the download operation.

To calculate the percentage of progress to show in a progress indicator (when *Progress* is not –1), multiply the value of *Progress* by 100 and divide by the value of *ProgressMax*.

See also
OnWBDocumentComplete, OnWBDownloadBegin and OnWBDownloadComplete events.

9.33.4.13OnWBPropertyChange

Applies to
acIESniffer component.

Declaration

```
type
  TacIEWebBrowserPropertyChange = procedure(Sender: TObject; const URL:
String; const Browser: IWebBrowser2;
    const szProperty: WideString) of object;

property OnWBPropertyChange: TacIEWebBrowserPropertyChange;
```

Description
The OnWBPropertyChange event occurs with Internet Explorer when a property is modified using the PutProperty method.

See also
OnWBCommandStateChange and OnWBProgressChange events.

9.33.4.14OnWBQuit

Applies to
acIESniffer component.

© 1998-2006, UtilMind Solutions®

Declaration

```
type
  TacIESnifferEvent = procedure(Sender: TObject; const URL: String;
const Browser: IWebBrowser2) of object;

property OnWBQuit: TacIESnifferEvent;
```

Description

The OnWBQuit event occurs when the Internet Explorer is about to shut down (when user close the window or the window closed automatically by some script).

Write the OnWBQuit event handler to be notified when the Internet Explorer window is about to be closed. After receiving OnWBQuit event you can not use *Browser* anymore since its handle will be destroyed after this event. The instance of this Browser object already removed from internal list of acIESniffer component.

See also

OnWindowUnload and OnWindowLoad events.

9.33.4.15OnWBStatusBar

Applies to

acIESniffer component.

Declaration

```
type
  TacIEWebBrowserOnStatusBar = procedure(Sender: TObject; const URL:
String; const Browser: IWebBrowser2;
    StatusBar: WordBool) of object;

property OnWBStatusBar: TacIEWebBrowserOnStatusBar;
```

Description

The OnWBStatusBar event occurs when the Internet Explorer shows or hides its status bar.

See also

OnWBVisible, OnWBToolBar, OnWBMenuBar, OnWBTheaterMode and OnWBFullScreen events.

9.33.4.16OnWBStatusTextChange

Applies to

acIESniffer component.

Declaration

```
type
  TacIEWebBrowserStatusTextChange = procedure(Sender: TObject; const
URL: String; const Browser: IWebBrowser2;
    var Text: WideString) of object;

property OnWBStatusTextChange: TacIEWebBrowserStatusTextChange;
```

Description

The OnStatusTextChange occurs when the text displayed in the Internet Explorer's status bar changes.

Browser is the IWebBrowser2 interface that needs a new window to display its target resource.

© 1998-2006, UtilMind Solutions®

Text is the text which currently displayed in the status line of *Browser*. You can change this value and update the text visible in IE status bar.

Example

```
procedure TForm1.acIESniffer1WBStatusTextChange(Sender: TObject;
  const URL: String; const Browser: IWebBrowser2; const Text:
WideString);
begin
  StatusLabel.Caption := Text;
end;
```

See also

OnWBTitleChange event.

9.33.4.17OnWBTheaterMode

Applies to

acIESniffer component.

Declaration

```
type
  TacIEWebBrowserOnTheaterMode = procedure(Sender: TObject; const URL:
String; const Browser: IWebBrowser2; TheaterMode: WordBool) of object;

property OnWBFullScreen: TacIEWebBrowserOnFullScreen;
```

Description

The OnWBTheaterMode event occurs when the Internet Explorer changes into or out of theater mode.

See also

OnWBVisible, OnWBToolBar, OnWBStatusBar, OnWBMenuBar and OnWBFullScreen events.

9.33.4.18OnWBTitleChange

Applies to

acIESniffer component.

Declaration

```
type
  TacIEWebBrowserTextChange = procedure(Sender: TObject; const URL:
String; const Browser: IWebBrowser2;
    const Text: WideString) of object;

property OnWBTitleChange: TacIEWebBrowserTextChange;
```

Description

The OnWBTitleChange event occurs when the title of a document in the IWebBrowser2 interface becomes available or changes.

Write an OnWBTitleChange event handler to respond when the Web browser obtains information about the document title. Because the title can change while an HTML page is downloading, LocationName is originally set to the URL of the document. After the title specified in the HTML page, if any, becomes available, LocationName is changed to reflect the actual title.

The *Browser* parameter is the IWebBrowser2 interface that is unloading a document.

© 1998-2006, UtilMind Solutions®

Text is the newly-available title of the document.

See also
OnWBStatusTextChange event.

9.33.4.19OnWBToolBar

Applies to
acIESniffer component.

Declaration
```
type
  TacIEWebBrowserOnToolBar = procedure(Sender: TObject; const URL:
String; const Browser: IWebBrowser2;
    ToolBar: WordBool) of object;

property OnWBToolBar: TacIEWebBrowserOnToolBar;
```

Description
The OnWBToolBar event occurs when the Internet Explorer changes which tool bars are visible.

See also
OnWBVisible, OnWBStatusBar, OnWBMenuBar, OnWBTheaterMode and OnWBFullScreen events.

9.33.4.20OnWBVisible

Applies to
acIESniffer component.

Declaration
```
type
  TacIEWebBrowserOnVisible = procedure(Sender: TObject; const URL:
String; const Browser: IWebBrowser2;
    Visible: WordBool) of object;

property OnWBVisible: TacIEWebBrowserOnVisible;
```

Description
The OnWBVisible event occurs when the Web browser window is about to be shown or hidden.

Write an OnWBVisible event handler to make adjustments to compensate for the appearance or disappearance of the Web browser window.

The *Browser* parameter is the Web browser (IWebBrowser2 interface) whose *Visible* property is changing.

Visible is True if the Web browser is about to appear, False if the Web browser is about to be hidden.

See also
OnWBToolBar, OnWBStatusBar, OnWBMenuBar, OnWBTheaterMode and OnWBFullScreen events.

© 1998-2006, UtilMind Solutions®

9.33.4.21 OnWindowLoad

Applies to

acIESniffer component.

Declaration

```
type
  TacWindowLoadState = (lsExistsLoading, lsExistsCompleted, lsLoading,
lsCompleted);

  TacIESnifferLoadEvent = procedure(Sender: TObject; const URL: String;
const Browser: IWebBrowser2;
    State: TacWindowLoadState) of object;

property OnWindowLoad: TacIESnifferLoadEvent;
```

Description

The OnWindowLoad event occurs when the component detects new Internet Explorer window.

The *URL* parameter is the text sniffed from the address line, *Browser* is the pointer to the TWebBrowser2 interface, detected by component. You can use the properties and methods of *Browser* right in this event handler.

The *State* parameter determines whether the Internet Explorer window was already exists when your application started and whether the content of the Web page are completely loaded. There is 4 possible values for this parameter:

Value	Meaning
lsExistsLoading	the IE window was already running before your application started. Its content was not completely loaded (still downloading);
lsExistsCompleted	the IE window was already running before your application started and all its content was completely loaded (so you can change);
lsLoading	this is new IE window detected by program. The Web page still loading;
lsCompleted	this is new IE window, and the Web page already completed (even ready to be modified!, see OnWBDownloadComplete event for more details and sample code which shows how to dynamically modify the Web page content);

If SniffWithHTTPPrefixOnly property is True, the component will detect only that windows which already have "http" prefix in the address line (shows content received by HTTP or HTTPS protocols). Otherwise, if SniffWithHTTPPrefixOnly is False, the component can detect windows with any text in the address line, even if it's "about:blank" or path to some local directory in Windows Explorer.

The OnWBDownloadComplete event will be triggered if the Explorer window was detected with IsCompleted status (if the window was not existeted on application startup but the content was already completed upon detecting the IE instance).

The OnURLChange event will occur at once after detecting of new IE window.

Example (demonstrates how to replace the text if the page already completed, but you can do the same in OnWBDownloadComplete event handler)

```
procedure TForm1.acIESniffer1WindowLoad(Sender: TObject; const URL:
String;
  const Browser: IWebBrowser2; State: TacWindowLoadState);
const
```

© 1998-2006, UtilMind Solutions®

```
  StateStr: Array[TacWindowLoadState] of String = ('Exists Loading',
'Exists Completed',
                                                   'New Loading', 'New
Completed');
begin
  Label1.Caption := StateStr[State];
  if (State = lsExistsCompleted) or (State = lsCompleted) then
    IESniffer.ReplaceText(Browser, 'Delphi', 'Delphi Rules!');
end;
```

See also
SniffWithHTTPPrefixOnly property;
OnWindowUnload, OnWBDownloadComplete and OnURLChange events.

9.33.4.22OnWindowUnload

Applies to
acIESniffer component.

Declaration

```
type
  TacIESnifferUnloadEvent = procedure(Sender: TObject; const URL:
String) of object;

property OnWindowUnload: TacIESnifferUnloadEvent;
```

Description
The OnWindowUnload event notifies that the Explorer window was closed (disappears from screen), process has been terminated and all its handles was destroyed, so you can not use its IWebBrowser2 interface anymore.

However, you should stop using that browser instance once it only was about to be destroyed, after receiving OnWBQuit event.

See also
OnWindowLoad and OnWBQuit events.

9.34 acIESnifferAutoFillUserInfo

9.34.1 TacIESnifferAutoFillUserInfo

Overview
The acIESnifferAutoFillUserInfo is the "plug-in" for acIESniffer component which allows to automatically fill the Web forms with specified information, when it connected to the FormAutoFill property of acIESniffer.

How does it works?
When the this component is connected to acIESniffer, it hooks the DownloadComplete event of each browser (listed in the internal list of acIESniffer). When it detects the DownloadComplete event (it occurs when the Web page is completely downloaded), the acIESnifferAutoFillUserInfo are looking in the Web forms for the input fields which can be automatically filled with information provided in Fields structure.

To find the input boxes which could be filled, the component uses so-called FillTokens. When the component find on the Web page some input box, it checks whether its name attribute (in the HTML tag) and the text prior to that input box. If the text nearby contains one of the words described

© 1998-2006, UtilMind Solutions®

in "Possible" property AND do NOT have the words described in "Wrong" property, OR contains one of the word specified in "Super" property, OR the field name is the one of the word specified in "ExactNameField", then the input field can be automatically marked, or filled with the proper text, taken from Fields structure.

new! To specify custom fields/tokens for filling of the web forms — specify them to the CustomFields property (collection of the custom fields/tokens).

How to use?

If you want to automatically fill that fields with provided information — set AutoFill property to True. In case if you just want to "highlight" that fields which possibly could be filled — set AutoHighlight property to True, and specify the background color for the marked input boxes to HighlightColor property.

To fill the form fields programmatically (for example, when user clicks some button, not on each DownloadComplete event) — call the **Fill** method.

See also

acIESniffer component.

9.34.2 Properties

9.34.2.1 AutoFill

Applies to

acIESnifferAutoFillUserInfo component.

Declaration

```
property AutoFill: Boolean;
```

Description

The AutoFill property controls whether the Web forms should be automatically filled with information specified in Fields structure, when the IE browser generates the DownloadComplete event.

See also

AutoHighlight and Fields properties;
Fill method.

9.34.2.2 AutoHighlight

Applies to

acIESnifferAutoFillUserInfo component.

Declaration

```
property AutoHighlight: Boolean;
```

Description

The AutoFill property controls whether the Web forms which *could be* automatically filled should be "highlighted" (with colors specified in HighlightColor and HighlightTextColor properties), when the IE browser generates the DownloadComplete event.

The empty fields, where the text is not specified, will NOT be highlighted.

See also

AutoFill and Fields properties;
Fill method.

© 1998-2006, UtilMind Solutions®

9.34.2.3 CustomFields

Applies to
acIESnifferAutoFillUserInfo component.

Declaration

```
type
  TIESnifferAutoFillCustomFieldType = (ftAny, ftEditBox, ftComboBox);
  TIESnifferAutoFillCustomField = class
  published
    property FieldType: TIESnifferAutoFillCustomFieldType default ftAny;
    property Name: String;
    property Value: String;
    property Tokens: TIESnifferAutoFillTokens;
  end;

property CustomFields: TIESnifferAutoFillCustomFields;
```

Description
The CustomFields property is the collection of the custom fields/tokens. Each item of this collection respresents *4 tokens* of the field on the Web forms and the *Value*, which should be specified to that field on filling.

To add custom fields at design-time — click on the property name and add the custom fields in special designer.

See also
Fields and FillTokens properties;
Fill method.

9.34.2.4 Fields

Applies to
acIESnifferAutoFillUserInfo component.

Declaration

```
type
  TacIESnifferAutoFillUserFields = class(TPersistent)
  published
    property FullName: String;
    property Company: String;
    property JobTitle: String;
    property Email: String;
    property Phone: String;
    property Fax: String;
    property TaxIDNumber: String;

    property AddressLine1: String;
    property AddressLine2: String;
    property City: String;
    property State: String;
    property ZipCode: String;
    property Country: String;
  end;

property Fields: TacIESnifferAutoFillUserFields;
```

© 1998-2006, UtilMind Solutions®

Description

The Fields structure used to specify the user details information: full name, email, phone, fax and full address information. This information can be automatically put by the component to Web forms if AutoFill property is True, or after calling the Fill method.

The component don't highlight and don't fill that fields where the text is empty.

To specify custom fields, i.e, bank account information, social security number, or something else — use CustomFields property.

See also

AutoFill, AutoHighlight, CustomFields, FillTokens and RegistrySaver properties;
Fill method.

9.34.2.5 FillTokens

Applies to

acIESnifferAutoFillUserInfo component.

Declaration

```
type
  TacIESnifferAutoFillTokens = class(TPersistent)
  published
    property Super: String;
    property ExactNameField: String;
    property Possible: String;
    property Wrong: String;
  end;

  TacIESnifferAutoFillUserTokens = class(TPersistent)
  published
    property FullName: TacIESnifferAutoFillTokens;
    property FirstName: TacIESnifferAutoFillTokens;
    property LastName: TacIESnifferAutoFillTokens;
    property Company: TacIESnifferAutoFillTokens;
    property JobTitle: TacIESnifferAutoFillTokens;
    property Email: TacIESnifferAutoFillTokens;
    property Phone: TacIESnifferAutoFillTokens;
    property Fax: TacIESnifferAutoFillTokens;
    property TaxIDNumber: TacIESnifferAutoFillTokens;
    property AddressLine1: TacIESnifferAutoFillTokens;
    property AddressLine2: TacIESnifferAutoFillTokens;
    property City: TacIESnifferAutoFillTokens;
    property State: TacIESnifferAutoFillTokens;
    property ZipCode: TacIESnifferAutoFillTokens;
    property Country: TacIESnifferAutoFillTokens;
  end;

property FillTokens: TacIESnifferAutoFillUserTokens;
```

Description

The FillTokens structure can be used to specify custom "tokens" of the fields which can be automatically filled with values in Fields structure.

When the component find on the Web page some input box, it checks whether its name attribute (in HTML tag) and the text prior to that input box. If the text nearby contains one of the words

© 1998-2006, UtilMind Solutions®

described in "**Possible**" property AND do NOT have the words described in "**Wrong**" property, OR contains one of the word specified in "**Super**" property, OR the field name is the one of the word specified in "**ExactNameField**", then the input field can be automatically marked, or filled with the proper text, taken from Fields structure.

To specify custom fields/tokens, i.e, bank account information, social security number, or something else — use CustomFields property.

See also
AutoFill, AutoHighlight, Fields and CustomFields properties;
Fill method.

9.34.2.6 HighlightColor

Applies to
acIESnifferAutoFillUserInfo component.

Declaration
```
property HighlightColor: TColor; // $00A0FFFF by default
```

Description
The HighlightColor property specifies the *background color* for the fields on the Web forms, which could be automatically filled with text specified in Fields structure.

If you specifying too dark background color, then specify lighter text color in HighlightTextColor property.

See also
AutoHighlight, HighlightTextColor and Fields properties;
Fill method.

9.34.2.7 HighlightTextColor

Applies to
acIESnifferAutoFillUserInfo component.

Declaration
```
property HighlightTextColor: TColor; // black by default
```

Description
The HighlightTextColor property specifies the *text color* for the fields on the Web forms, which could be automatically filled with text specified in Fields structure.

To specify the *background color* for fields which could be filled — use HighlightColor property.

See also
AutoHighlight, HighlightColor and Fields properties;
Fill method.

9.34.2.8 RegistrySaver

Applies to
acIESnifferAutoFillUserInfo component.

Declaration
```
type
```

© 1998-2006, UtilMind Solutions®

```
  TacIESnifferAutoFillRegistrySaver = class
  published
    property Enabled: Boolean;
    property RegKey: String;
    property RegLocation: TacRegLocation;
  end;

property RegistrySaver: TacIESnifferAutoFillRegistrySaver;
```

Description

The RegistrySaver structure used to automatically save and retreive the values for Fields structure from registry.

To let the component automatically retreive the Fields from registry on program startup — set *Enabled* property to True. To specify the registy key and location where the text should be stored — specify *RegKey* and *RegLocation* properties.

To save the Fields to registry after their modification — call **Save** method.

See also

Fields property;
Save method.

9.34.3 Methods

9.34.3.1 Fill

Applies to

acIESnifferAutoFillUserInfo component.

Declaration

```
procedure Fill(const Browser: IWebBrowser2; BackgroundColor: TColor =
clNone; TextColor: TColor = clNone;
  HighlightOnly: Boolean = False; ScanFrames: Boolean = True); overload;

procedure Fill(const Document: IHTMLDocument2; BackgroundColor: TColor =
clNone; TextColor: TColor = clNone;
  HighlightOnly: Boolean = False; ScanFrames: Boolean = True); overload;
```

Description

The Fill method used to mark or fill the fields of the Web forms with text specified in Fields structure.

Browser (or *Document*) parameter points to the interface which holds the web page.

BackgroundColor and *TextColor* is the optional parameters used to specify custom background and text colors for marked form fields.

HighlightOnly parameter can be set to True, if you just want to highlight fields which could be filled, without filling them.

ScanFrames is also optional parameter, which specifies whether the method should scan all frames inside the specified *Browser* interface or *Document*.

Example

```
procedure TIEBandForm.AutoFillBtnClick(Sender: TObject);
begin
```

© 1998-2006, UtilMind Solutions®

```
    acIESnifferAutoFillUserInfo1.Fill(FBrowser);
end;
```

See also
AutoFill, AutoHighlight, HighlightColor, HighlightTextColor and Fields properties;
acIESniffer component.

9.34.3.2 Save

Applies to
acIESnifferAutoFillUserInfo component.

Declaration
```
procedure Save;
```

Description
The Save method saves the values of Fields structure to the registry (or INI-file), to the registry keys specified in RegistrySaver structure.

Example
```
procedure TAutoFillFrm.ButtonsPanel1ApplySettings(Sender: TObject);
begin
  with acIESnifferAutoFillUserInfo1, Fields do
   begin
    FullName := EName.Text;
    Email := EEMail.Text;
    Company := ECompany.Text;
    JobTitle := EJobTitle.Text;
    Phone := EPhone.Text;
    Fax := EFax.Text;

    AddressLine1 := ELine1.Text;
    AddressLine2 := ELine2.Text;
    City := ECity.Text;
    State := EState.Text;
    ZIPCode := EZIP.Text;
    Country := ECountry.Text;

    Save;
   end;
end;
```

See also
Fields and RegistrySaver structures;
acIESniffer component.

9.35 acInfoBox

9.35.1 TacInfoBox

Overview
The acInfoBox component is a quick and simpe implementation of smart dialog boxes. You probably already seen these smart dialog boxes with some notification message and *"Don't show me this message again"* option below text. If user will find message annoying and don't want to see this notification again, it simply check this box and message will not appear anymore.

© 1998-2006, UtilMind Solutions®

The acInfoBox can show the message with rich-formatted text with graphic icon at the left side of dialog, and contains the corresponding check box (with customizeable caption, i.e: *"Don't display me this again"*). The state for this check box is kept in registry. All implementation with just one line of code.

Snapshot

How to use?

ss

You can test how your InfoBox will looks like even at design time, just specify all needed properties, right-click on the component icon on your form and select "Test!" menu item from popup menu. (screenshot required?)

See also

9.35.2 Properties

9.35.2.1 ButtonCaptions

Applies to

acInfoBox component.

Declaration

```
type
  TacInfoBoxButtons = (ibOK, ibOKCancel, ibYesNo, ibYesNoCancel);

property ButtonCaptions: TacInfoBoxButtonCaptions;
```

Description

See also

9.35.2.2 CheckBoxCaption

Applies to

acInfoBox component.

Declaration

```
property ButtonCaptions: TCaption; // String
```

Description

See also

© 1998-2006, UtilMind Solutions®

9.35.2.3 Font

Applies to
acInfoBox component.

Declaration
```
property Font: TFont;
```

Description

See also
Text and TextStyle properties.

9.35.2.4 Icon

Applies to
acInfoBox component.

Declaration
```
property Icon: TPicture;
```

Description

See also

9.35.2.5 InfoBoxButtons

Applies to
acInfoBox component.

Declaration
```
property InfoBoxButtons: TacInfoBoxButtons;
```

Description

See also

9.35.2.6 Picture

Applies to
acInfoBox component.

Declaration
```
property Picture: TPicture;
```

Description

See also

9.35.2.7 PictureResID

Applies to
acInfoBox component.

© 1998-2006, UtilMind Solutions®

Declaration

```
property PictureResID: Integer;
```

Description

See also

9.35.2.8 Position

Applies to

acInfoBox component.

Declaration

```
type
  TPosition = (poDesigned, poDefault, poDefaultPosOnly,
poDefaultSizeOnly, poScreenCenter, poDesktopCenter, poMainFormCenter,
poOwnerFormCenter);

property Position: TPosition;
```

Description

Use Position to get or set the size and placement of the form. Position can have one of the following TPosition values:

Value Meaning

poDesigned The form appears positioned on the screen and with the same height and width as it had at design time.
poDefault The form appears in a position on the screen and with a height and width determined by Windows. Each time you run the application, the form moves slightly down and to the right. The right side of the form is always near the far right side of the screen, and the bottom of the form is always near the bottom of the screen, regardless of the screen's resolution.

poDefaultPosOnly The form displays with the size you created it at design time, but Windows chooses its position on the screen. Each time you run the application, the form moves slightly down and to the right. When the form can no longer move down and to the right and keep the same size while remaining entirely visible on the screen, the form displays at the top-left corner of the screen.
poDefaultSizeOnly The form appears in the position you left it at design time, but Windows chooses its size. The right side of the form is always near the far right side of the screen, and the bottom of the form is always near the bottom of the screen, regardless of the screen's resolution.

poScreenCenter The form remains the size you left it at design time, but is positioned in the center of the screen. In multi-monitor applications, the form may be moved from this center position so that it falls entirely on one monitor, as specified by the DefaultMonitor property.
poDesktopCenter The form remains the size you left it at design time, but is positioned in the center of the screen. No adjustments are made for multi-monitor applications.
poMainFormCenter The form remains the size you left it at design time, but is positioned in the center of the application's main form. No adjustments are made for multi-monitor applications. This position should only be used with secondary forms. If set for a main form, it acts like poScreenCenter.

poOwnerFormCenter The form remains the size you left it at design time, but is positioned in the center of the form specified by the Owner property. If the Owner property does not specify a form, this position acts like poMainFormCenter.

See also

© 1998-2006, UtilMind Solutions®

9.35.2.9 RegKey

Applies to
acInfoBox component.

Declaration
```
property RegKey: String;
```

Description

See also
RegLocation and RegValue properties.

9.35.2.10 RegLocation

Applies to
acInfoBox component.

Declaration
```
type
  TacRegLocation = (rlCurrentUser, rlLocalMachine)

property RegLocation: TacRegLocation;
```

Description

See also
RegKey and RegValue properties.

9.35.2.11 RegValue

Applies to
acInfoBox component.

Declaration
```
property RegValue: String;
```

Description

See also
RegKey and RegLocation properties.

9.35.2.12 ShowCheckBox

Applies to
acInfoBox component.

Declaration
```
property ShowCheckBox: Boolean;
```

Description

See also

© 1998-2006, UtilMind Solutions®

9.35.2.13Text

Applies to
acInfoBox component.

Declaration
```
property Text: String;
```

Description
The Text property specifies the text

See also
TextStyle, Font and Title properties.

9.35.2.14TextStyle

Applies to
acInfoBox, acRichLabel, acFormHelp, acHeader, acHeaderPanel and acPicturePanel components.

Declaration
```
type
  TacTextStyle = class(TPersistent)
  published
    property PlainText: Boolean default False;
    property TagOpen: Char default '[';
    property TagClose: Char default ']';
  end;
```

Description
The TextStyle structure used to specify the style of text displayed by this component.

The PlainText property specifies whether you would like to use right-formatted text or just flat, plain text. TagOpen and TagClose properties used to specify prefered signs for brackets which identifies begin and end of tags used for rich formatting. By default every tag beginning with '**[**' sign and ending with '**]**' sign.

For example, when PlainText is False, TagOpen is '[' and TagClose is ']', following text:

```
[i]Hello[]  [red][b]World[def]!
```

will looks like:

Hello **World**!

Otherwise, if PlainText is True, the tags will not be used and users will see just flat text, including all tags.

See also
Text property;
acRichLabel component and Rich-formatting tags overview.

9.35.2.14.1 PlainText

Applies to
acInfoBox, acRichLabel, acFormHelp, acHeader, acHeaderPanel and acPicturePanel components as subproperty of TextStyle structure.

Declaration
```
property PlainText: Boolean;
```

© 1998-2006, UtilMind Solutions®

Description
The PlainText property controls whether the text should be displayed as rich text (if PlainText is False) or as plain, non-formatted text (if PlainText is True). When PlainText is True, the text will be displayed as usual flat text and nor tags will be used.

See also
TagOpen and TagClose properties;
Using rich-formatting tags topic.

9.35.2.14.2 TagClose

Applies to
acInfoBox, acRichLabel, acFormHelp, acHeader, acHeaderPanel and acPicturePanel components as subproperty of TextStyle structure.

Declaration
```
property TagClose: Char;
```

Description
The TagClose property specifies the sign which marks the ending of formatting area of rich-text. The rich formatting tags of AppControls pack are similar to brackets "<>" of tags in HTML format. Default TagClose value is "**]**" sign.

Examples of standard tags
`[b]` - makes text after this tag **bold**.
`[i]` - makes text after this tag *italic*
`[biu]` - makes text after this tag ***bold, italic and underlined***.
`[]` - clears text formatting.
`[red]` - makes text after this tag red.
`[lime]` - makes text after this tag light green (lime).
`[def]` or `[default]` - returns the text formatting to default state as specified in Font property.

Note
If PlainText property is True, nor rich text will be displayed. All formating will be shown as usual flat text.

See also
TagOpen and PlainText properties and example of using rich-formatting tags.

9.35.2.14.3 TagOpen

Applies to
acInfoBox, acRichLabel, acFormHelp, acHeader, acHeaderPanel and acPicturePanel components as subproperty of TextStyle structure.

Declaration
```
property TagOpen: Char;
```

Description
The TagOpen property specifies the sign which marks the beginning of rich-text formatting area. The rich formatting tags of AppControls pack are similar to brackets "<>" of tags in HTML format. Default TagOpen value is "**[**" sign.

Examples of standard tags
`[b]` - makes text after this tag **bold**.

© 1998-2006, UtilMind Solutions®

[i] - makes text after this tag *italic*
[biu] - makes text after this tag ***bold, italic and underlined***.
[] - clears text formatting.
[red] - makes text after this tag red.
[lime] - makes text after this tag light green (lime).
[def] or [default] - returns the text formatting to default state as specified in Font property.

See also
TagClose and PlainText properties and example of using rich-formatting tags.

9.35.2.15Title

Applies to
acInfoBox component.

Declaration
```
property Title: TCaption;
```

Description
The Title property specifies the text string for the title bar of dialog box.

See also
Text property.

9.35.3 Methods

9.35.3.1 Execute

Applies to

Declaration
```
function Execute(OnlyIfPossible: Boolean): Integer;
```

Description

See also

9.35.4 Events

9.35.4.1 OnCheckBoxChecked

Applies to
acInfoBox component.

Declaration
```
type
  TacInfoBoxCheckBoxCheckedEvent = procedure(Sender: TObject; Checked:
Boolean) of object;

property OnCheckBoxChecked: TacInfoBoxCheckBoxCheckedEvent;
```

Description

See also

© 1998-2006, UtilMind Solutions®

9.36 acOnlyOne

9.36.1 TacOnlyOne

Overview

The acOnlyOne component detects already running instances of the application and disallow to start the second copy of the same program.

After detecting of the second application instance, acOnlyOne will automatically terminate program execution and switch the tasks to previous program instance (if SwitchToFirst property is True). If you wish to take some specific actions if another application instance detected (application already running) — write OnAlreadyExists event handler.

---

new! The acOnlyOne unit now exports several functions which can be used outside of the component, in the initialization section of your application (*.DPR unit), so you can close second application instance and switch the input focus to previous, before the second instance will create any form.

Here is an example (procedures exported from acOnlyOne unit marked with red color):

```
program YourAppTitle;

uses
  acOnlyOne,
  acUtils,
  Main in 'Main.pas' {MainForm};

{$R *.res}

begin
  if HasParamStr('/close') then
   begin
    CloseFirstAppInstance;
    Exit;
   end;

  // The Application.Title should be initialized before checking whether
the application instance exists
  Application.Initialize;
  Application.Title := 'YourAppTitle';

  if IsAppInstanceAlreadyExists then
   begin
    InfoBox('The application already running!');
    SwitchToFirstAppInstance;
    Exit;
   end;

  Application.CreateForm(TMainForm, MainForm);
  Application.Run;
end.
```

© 1998-2006, UtilMind Solutions®

9.36.2 Properties

9.36.2.1 CloseFirst

Applies to
acOnlyOne component.

Declaration
```
property CloseFirst: Boolean; // default False
```

Description
The CloseFirst property used to close first application instance on the startup of second instance.

Set CloseFirst property to True, and your application will "kick" previous program instances on startup.

See also
CloseSecond and SwitchToFirst properties;
CloseFirstInstance method;
OnAlreadyExist event.

9.36.2.2 CloseSecond

Applies to
acOnlyOne component.

Declaration
```
property CloseSecond: Boolean; // default True
```

Description
The CloseSecond property controls whether the second (and greater) instance of application should be closed if the application already running. However, if you wish to allow to run more than one instance of your application — set CloseSecond to False. Otherwise, if you wish to kill all superfluous instances — leave it True.

See also
CloseFirst and SwitchToPrevious properties;
OnAlreadyExist event.

9.36.2.3 SwitchToFirst

Applies to
acOnlyOne component.

Declaration
```
property SwitchToFirst: Boolean; // default True
```

Description
If SwitchToFirst property is True and another copy of the same application is detected, acOnlyOne will switch the tasks to previous program instance.

Note
You don't need to change value of this property at run-time. acOnlyOne use this property on program startup only.

See also

© 1998-2006, UtilMind Solutions®

CloseFirst and CloseSecond properties;
OnAlreadyExist event.

9.36.3 Methods

9.36.3.1 CloseFirstInstance

Applies to
acOnlyOne component.

Declaration
```
procedure CloseFirstInstance;
```

Description
The CloseFirstInstance method closes the previos instance of the application.

Alternatively you can set CloseFirst property on startup and first program instance will be closed automatically.

However, if you would like to close ALL instances of your application when, for example, your program started with "/close" command-line parameter, you can use **CloseFirstAppInstance** procedure, exported from acOnlyOne unit, and use following code in the main ".DPR" unit of your program. This example demonstrates how to kill previous program instance and close itself without creating any forms:

```
program YourProgramName;

uses
  acOnlyOne,
  acUtils,
  Main in 'Main.pas' {MainForm};

{$R *.RES}

begin
  if HasParamStr('/close') then
   begin
    CloseFirstAppInstance;
    Exit;
   end;

  Application.Initialize;
  Application.Title := 'YourProgramTitle';
  Application.CreateForm(TMainForm, MainForm);
  Application.Run;
end.
```

Also you can check out the **CloseApp** command-line tool in the "Tools" directory of distribution package. That utility can be used to close ANY application which uses TacOnlyOne component, all you need is just specify the the application title as command-line parameter. This can be very useful for silent installations which should close the installation instance before overwrite it.
Usage of CloseApp.exe is following:

```
CloseApp.exe [ApplicationTitle]
```

For example:

```
CloseApp.exe MyAppTitle
```

© 1998-2006, UtilMind Solutions®

See also
CloseFirst and CloseSecond properties;
OnAlreadyExist event.

9.36.4 Events

9.36.4.1 OnAlreadyExist

Applies to
acOnlyOne component.

Declaration
```
property OnAreadyExist: TNotifyEvent;
```

Description
The OnAlreadyExist event will occurs on program startup if the second instance of the same program is already running.

See also
SwitchToFirst property.

9.37 acProcessList

9.37.1 acProcessList

Overview
The acProcessList component used to retrieve the list of all running system processes. You can retrieve the full list simly calling the Refresh method and gain access to the information about each process, using the component as usual TList. TacProcess objects contains the information about process and has Kill method which able to imediately terminate the process.

How does it works and how to use?
The component is the container of the list of **TacProcess** objects. Each TacProcess object contains brief information about the process (it's executable file name, description, application type, process identifier, icon index in the system image list and the list of windows which belongs to the process). Also, the TacProcess object allows to terminate the process using its Kill method.

The acProcessList component already contains the list of currently running processes when your application starts. However, to keep the list fresh, you should call Refresh method, so the component will re-retreive the list of current processes.

Following example demonstrates how to retrieve the list of processes and display them in the ListView:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  I: Integer;
  ListItem: TListItem;
begin
  acProcessList1.Refresh; // refreshing the list of processes

  ListView1.Items.BeginUpdate;
  try
    ListView1.Items.Clear;
    I := acProcessList1.Count;
    if I <> 0 then
     for I := 0 to I - 1 do
```

© 1998-2006, UtilMind Solutions®

```
        begin
          ListItem := ListView1.Items.Add;
          ListItem.Caption := acProcessList1[I].ExeName;
          ListItem.ImageIndex := acProcessList1[I].SystemIconIndex;
          ListItem.SubItems.Add(acProcessList1[I].Description);

          // ...optionally, let's associate the Data pointer to
    TacProcess object
          // and Data pointer of Process object to ListItem
          ListItem.Data := acProcessList1[I];
          acProcessList1[I].Data := ListItem;
        end;
      finally
        ListView1.Items.EndUpdate;
      end;
    end;

    // This code demonstrates how to display the process description and
    its 32x32 icon
    // when user selects the list item of ListView
    procedure TForm1.ListView1SelectItem(Sender: TObject; Item: TListItem;
      Selected: Boolean);
    begin
      if Item <> nil then
       with TacProcess(Item.Data) do
        begin
          Label1.Caption := Description;
          Image1.Picture.Icon.Handle := SystemIcon32Handle;
        end;
    end;
```

See also
acConnectionList component.

9.37.2 Properties

9.37.2.1 Count

Applies to
acProcessList and acConnectionList components.

Declaration
```
property Count: Integer; // read-only!
```

Description
The Count is the read-only property which determines the number of Items (TacProcess objects) which are available in the list.

To get access to the list items — use Items property.

See also
Items property.

9.37.2.2 Items

Applies to
acProcessList component.

© 1998-2006, UtilMind Solutions®

Declaration

```
property Items[Index: Integer]: TacProcess read Get; default; // read-only!
```

Description

The Items is the *run-time only* property, which can be used to get an access to each instance of TacProcess object (which contains the information about the process) at particular position. *Index* gives the position of the TacProcess object, where 0 is the position of the first process, 1 is the position of the second process, and so on ("Count - 1" is the position of last process in the list).

Items is the default property of acProcessList component. The Items identifier can be omitted when accessing the TacProcess objects. For example, the following two lines of code are both acceptable and do the same thing:

```
FirstProcess := acProcessList1.Items[0];
FirstProcess := acProcessList1[0];
```

To determinate the number of items which currently are available in the list — read Count property.

Example

```
// Demonstrates how to retrieve the list of processes and display them
in the ListView:
procedure TForm1.Button1Click(Sender: TObject);
var
  I: Integer;
  ListItem: TListItem;
begin
  acProcessList1.Refresh; // refreshing the list of processes

  ListView1.Items.BeginUpdate;
  try
    ListView1.Items.Clear;
    I := acProcessList1.Count;
    if I <> 0 then
     for I := 0 to I - 1 do
      begin
        ListItem := ListView1.Items.Add;
        ListItem.Caption := acProcessList1[I].ExeName;
        ListItem.ImageIndex := acProcessList1[I].SystemIconIndex;
        ListItem.SubItems.Add(acProcessList1[I].Description);

        // ...optionally, let's associate the Data pointer to TacProcess
object
        // and Data pointer of Process object to ListItem
        ListItem.Data := acProcessList1[I];
        acProcessList1[I].Data := ListItem;
      end;
  finally
    ListView1.Items.EndUpdate;
  end;
end;
```

See also

Count property.

© 1998-2006, UtilMind Solutions®

9.37.2.3 ProcessList

Applies to
acProcessList component.

Declaration
```
type
  TacProcesses = class(TacObjectList)
  public
    function FindProcessByPID(PID: DWord; MultiPass: Boolean = True):
TacProcess;
    function FindProcessByWnd(Wnd: hWnd; MultiPass: Boolean = True):
TacProcess;
    function FindExeNameByPID(PID: DWord): String;
    function FindExeNameByWnd(Wnd: hWnd): String;
    function FindDescriptionByPID(PID: DWord): String;
    function FindDescriptionByWnd(Wnd: hWnd): String;
    function FindWindowTitleByPID(PID: DWord): String;
    function FindWindowTitleByWnd(Wnd: hWnd): String;

    procedure ExportToListView(ListView: TCustomListView);
    procedure Refresh;

    property Items[Index: Integer]: TacProcess read Get; default;
  end;

property ProcessList: TacProcesses;
```

Description
The ProcessList is the internal list which contains the information about each system process.

The ProcessList is descendant of TList object, and contains only TacProcess objects.

The component does not refreshes this list automatically. To refresh this list — use Refresh method.

See also
TacProcess object;
Refresh method.

9.37.3 Methods

9.37.3.1 Refresh

Applies to
acProcessList, acConnectionList and acIESniffer components (used to refresh the content of the internal data list).

Declaration
```
procedure Refresh;
```

Description
The Refresh method retrieves the list of all currently running system processes and fills the internal list with new information about each TCP/UDP connection.

Use Refresh method to keep the list of connections always fresh. The component does NOT update

© 1998-2006, UtilMind Solutions®

its list automatically (it does it only once on startup).

See also
Count and Items properties.

9.37.3.2 FindProcessByPID

Applies to
acProcessList component.

Declaration
```
function FindProcessByPID(PID: DWord; RefreshIfNotFound: Boolean =
True): TacProcess;
```

Description
The FindProcessByPID method retrieves the pointer to TacProcess object by its process identifier. Set *RefreshIfNotFound* parameter to True, if you would like to refresh the list automatically if the process was not found in first try.

See also
Refresh, FindProcessByPID, FindProcessByWnd, FindExeNameByPID, FindExeNameByWnd, FindDescriptionByPID, FindDescriptionByWnd, FindWindowTitleByPID and FindWindowTitleByWnd methods;
TacProcess object.

9.37.3.3 FindProcessByWnd

Applies to
acProcessList component.

Declaration
```
function FindProcessByWnd(Wnd hWnd; RefreshIfNotFound: Boolean = True):
TacProcess;
```

Description
The FindProcessByWnd method retrieves the pointer to TacProcess object by the window handle which belongs to the process. Set *RefreshIfNotFound* parameter to True, if you would like to refresh the list automatically if the process was not found in first try.

See also
Refresh, FindProcessByPID, FindProcessByWnd, FindExeNameByPID, FindExeNameByWnd, FindDescriptionByPID, FindDescriptionByWnd, FindWindowTitleByPID and FindWindowTitleByWnd methods;
TacProcess object.

9.37.3.4 FindExeNameByPID

Applies to
acProcessList component.

Declaration
```
function FindExeNameByPID(PID: DWord; RefreshIfNotFound: Boolean =
True): String;
```

Description
The FindExeNameByPID method retrieves the name of the executable by the process identifier. Set *RefreshIfNotFound* parameter to True, if you would like to refresh the list automatically if the process

© 1998-2006, UtilMind Solutions®

was not found in first try.

See also
Refresh, FindProcessByPID, FindProcessByWnd, FindExeNameByPID, FindExeNameByWnd, FindDescriptionByPID, FindDescriptionByWnd, FindWindowTitleByPID and FindWindowTitleByWnd methods;
TacProcess object.

9.37.3.5 FindExeNameByWnd

Applies to
acProcessList component.

Declaration
```
function FindExeNameByWnd(Wnd hWnd; RefreshIfNotFound: Boolean = True):
TacProcess;
```

Description
The FindExeNameByWnd method retrieves the name of the executable by the window handle which belongs to the process. Set *RefreshIfNotFound* parameter to True, if you would like to refresh the list automatically if the process was not found in first try.

See also
Refresh, FindProcessByPID, FindProcessByWnd, FindExeNameByPID, FindExeNameByWnd, FindDescriptionByPID, FindDescriptionByWnd, FindWindowTitleByPID and FindWindowTitleByWnd methods;
TacProcess object.

9.37.3.6 FindDescriptionByPID

Applies to
acProcessList component.

Declaration
```
function FindDescriptionByPID(PID: DWord; RefreshIfNotFound: Boolean =
True): String;
```

Description
The FindDescriptionByPID method retrieves the process description by the process identifier. Set *RefreshIfNotFound* parameter to True, if you would like to refresh the list automatically if the process was not found in first try.

See also
Refresh, FindProcessByPID, FindProcessByWnd, FindExeNameByPID, FindExeNameByWnd, FindDescriptionByPID, FindDescriptionByWnd, FindWindowTitleByPID and FindWindowTitleByWnd methods;
TacProcess object.

9.37.3.7 FindDescriptionByWnd

Applies to
acProcessList component.

Declaration
```
function FindDescriptionByPID(Wnd: hWnd; RefreshIfNotFound: Boolean =
True): String;
```

© 1998-2006, UtilMind Solutions®

Description

The FindDescriptionByWnd method retrieves the process description by the window handle which belongs to the process. Set *RefreshIfNotFound* parameter to True, if you would like to refresh the list automatically if the process was not found in first try.

See also

Refresh, FindProcessByPID, FindProcessByWnd, FindExeNameByPID, FindExeNameByWnd, FindDescriptionByPID, FindDescriptionByWnd, FindWindowTitleByPID and FindWindowTitleByWnd methods;
TacProcess object.

9.37.3.8 FindWindowTitleByPID

Applies to

acProcessList component.

Declaration

```
function FindWindowTitleByPID(PID: DWord; RefreshIfNotFound: Boolean =
True): String;
```

Description

The FindWindowTitleByPID method retrieves the title of first visible window of the process by the process identifier. Set *RefreshIfNotFound* parameter to True, if you would like to refresh the list automatically if the process was not found in first try.

See also

Refresh, FindProcessByPID, FindProcessByWnd, FindExeNameByPID, FindExeNameByWnd, FindDescriptionByPID, FindDescriptionByWnd, FindWindowTitleByPID and FindWindowTitleByWnd methods;
TacProcess object.

9.37.3.9 FindWindowTitleByWnd

Applies to

acProcessList component.

Declaration

```
function FindWindowTitleByWnd(PID: hWnd; RefreshIfNotFound: Boolean =
True): String;
```

Description

The FindWindowTitleByWnd method retrieves the title of first visible window of the process by the window handle which belongs to the process. Set *RefreshIfNotFound* parameter to True, if you would like to refresh the list automatically if the process was not found in first try.

See also

Refresh, FindProcessByPID, FindProcessByWnd, FindExeNameByPID, FindExeNameByWnd, FindDescriptionByPID, FindDescriptionByWnd, FindWindowTitleByPID and FindWindowTitleByWnd methods;
TacProcess object.

9.37.4 TacProcess object

Unit

acProcessList

Declaration

© 1998-2006, UtilMind Solutions®

```
type
  TacProcess = class(TObject)
  public
    Data: Pointer;

    function Kill: Boolean;  // returns True if succeed

    property AppType: TacAppType;
    property AppTypeName: String;
    property Description: String;
    property ExeName: String;
    property ProcID: DWord;
    property Windows: hWndArray;
    property WindowTitle: String;
    property SystemIconIndex: Integer;
    property SystemIcon16Handle: hIcon;
    property SystemIcon32Handle: hIcon;
  end;
```

Description

The TacProcess object is the item of acProcessList component, which holds the information about the system process.

Each TacProcess object contains brief information about the process (it's executable file name, description, application type, process identifier, icon index in the system image list and the list of windows which belongs to the process). Also, it allows to terminate the process using its Kill method.

9.37.5 Properties

9.37.5.1 AppType

Applies to

acProcess object.

Declaration

```
type
  TacAppType =
      (atUnknown,         // type not yet determined
       atVisibleWindows,  // application with visible windows
       atHiddenWindows,   // application with only hidden windows
       atExplorer,        // Windows Explorer
       atNoWindows);      // application with no windows

property AppType: TacAppType; // read-only!
```

Description

The AppType determines the type of the application (whether it is GUI application with visible windows, hidden application, Explorer window, or just some system process).

The application type can be one of following values:

Value	Meaning
atVisibleWindows	the application have one or more visible windows
atHiddenWindows	all windows of the application are hidden
atExplorer	the application is Windows Explorer

© 1998-2006, UtilMind Solutions®

atNoWindows	the application has no windows

See also
AppTypeName property.

9.37.5.2 AppTypeName

Applies to
acProcess object.

Declaration
```
property AppTypeName: String; // read-only!
```

Description
The AppTypeName property is just human readable string representation of AppType property.

The value of this property can be one of following values:

Value	Meaning
Application	the application have one or more visible windows
Hidden	all windows of the application are hidden
Explorer	the application is Windows Explorer
Process	the application has no windows

See also
AppType property.

9.37.5.3 Data

plies to
acProcess and acConnection objects.

Declaration
```
property Data: Pointer;
```

Description
The Data property specifies any application-specific data associated with the TacProcess object.

Use Data to associate arbitrary data structure with the TacProcess object.

Example
```
// Demonstrates how to fill the ListView with process entries
// and associate ListItem with TacProcess object and otherwise,
TacProcess with ListItem...
procedure TForm1.Button1Click(Sender: TObject);
var
  I: Integer;
  ListItem: TListItem;
begin
  ListView1.Items.BeginUpdate;
  try
    ListView1.Items.Clear;
    I := acProcessList1.Count;
    if I <> 0 then
     for I := 0 to I - 1 do
      begin
```

© 1998-2006, UtilMind Solutions®

```
        ListItem := ListView1.Items.Add;
        ListItem.Caption := acProcessList1[I].ExeName;

        // associate the Data pointer to TacProcess object and Data
pointer of Process object to ListItem
        ListItem.Data := acProcessList1[I];
        acProcessList1[I].Data := ListItem;
      end;
  finally
    ListView1.Items.EndUpdate;
  end;
end;
```

9.37.5.4 Description

Applies to
acProcess object.

Declaration
```
property Description: String; // read-only!
```

Description
The Description property is the string which represents the description of the application, retrieved from the header of its executable file. If the file does not contains any description (i.e: 'Windows Executable' file does not contains version information), property returns the name of executable (see also ExeName).

Example *// shows the description of first process available in the list*
```
procedure TForm1.acButton1Click(Sender: TObject);
begin
  Label1.Caption := acProcessList1.Items[0].Description;
end;
```

See also
ExeName property.

9.37.5.5 ExeName

Applies to
acProcess object.

Declaration
```
property ExeName: String; // read-only!
```

Description
The ExeName is the name of the executable file, to which the process belongs.

To retrieve the description of the executable file from its header — use Description property. To retrieve the index of file icon from system image list — read SystemIconIndex property.

See also
Description, SystemIconIndex, SystemIcon16Handle and SystemIcon32Handle properties; acSystemImageList component.

© 1998-2006, UtilMind Solutions®

9.37.5.6 ProcID

Applies to
acProcess object.

Declaration
```
property ProcID: DWord; // read-only!
```

Description
The ProcID is the unique identifier of the process throughout the system.

Each process have its unique identifier (like each window have its unique handle). For example, if you wish to determinate the ID of the current process of your application, you can use `GetCurrentProcessID` function from WinAPI.

See also
Description, SystemIconIndex, SystemIcon16Handle and SystemIcon32Handle properties; acSystemImageList component.

9.37.5.7 SystemIcon16Handle

Applies to
acProcess object.

Declaration
```
property SystemIcon16Handle: hIcon; // read-only!
```

Description
The SystemIcon16Handle determines the handle of the 16x16 icon of the application process.

Example
```
// Demonstrates how to assign the handle to icon object and display it
in the TImage component
procedure TForm1.ListView1SelectItem(Sender: TObject; Item: TListItem;
  Selected: Boolean);
begin
  if Item <> nil then
    Image1.Picture.Icon.Handle :=
TacProcess(Item.Data).SystemIcon16Handle;
end;
```

See also
SystemIconIndex and SystemIcon32Handle properties; acSystemImageList component.

9.37.5.8 SystemIcon32Handle

Applies to
acProcess object.

Declaration
```
property SystemIcon16Handle: hIcon; // read-only!
```

Description
The SystemIcon32Handle determines the handle of the 32x32 icon of the application process.

© 1998-2006, UtilMind Solutions®

Example

```
// Demonstrates how to assign the handle to icon object and display it
in the TImage component
procedure TForm1.ListView1SelectItem(Sender: TObject; Item: TListItem;
  Selected: Boolean);
begin
  if Item <> nil then
    Image1.Picture.Icon.Handle :=
TacProcess(Item.Data).SystemIcon32Handle;
end;
```

See also

SystemIconIndex and SystemIcon16Handle properties;
acSystemImageList component.

9.37.5.9 SystemIconIndex

Applies to

acProcess object.

Declaration

```
property SystemIconIndex: Integer; // read-only!
```

Description

The SystemIconIndex determines the index of the application icon within the system image list.

To retrieve the full list of system images you can use acSystemImageList component.

Example

```
// Demonstrates how to fill the ListView with process entries and
display the icon of the process.
// Let's consider that ViewStyle = vsReport
// and SmallImages assigned to some acSystemImageList component dropped
onto the same form.
procedure TForm1.Button1Click(Sender: TObject);
var
  I: Integer;
  ListItem: TListItem;
begin
  ListView1.Items.BeginUpdate;
  try
    ListView1.Items.Clear;
    I := acProcessList1.Count;
    if I <> 0 then
     for I := 0 to I - 1 do
      begin
        ListItem := ListView1.Items.Add;
        ListItem.Caption := acProcessList1[I].ExeName;
        ListItem.ImageIndex := acProcessList1[I].SystemIconIndex;

        // associate the Data pointer to TacProcess object and Data
pointer of Process object to ListItem
        ListItem.Data := acProcessList1[I];
        acProcessList1[I].Data := ListItem;
      end;
  finally
    ListView1.Items.EndUpdate;
```

© 1998-2006, UtilMind Solutions®

```
  end;
end;
```

See also
Description, SystemIconIndex, SystemIcon16Handle and SystemIcon32Handle properties; acSystemImageList component.

9.37.5.10Windows

Applies to
acProcess object.

Declaration
```
property Windows: Array of hWnd; // read-only!
```

Description
The Windows property is the list of all windows handles which belongs to the process.

To enumerate all windows programmatically you can use following code example:
```
  if High(Windows) >= 0 then // if application have windows
   // Return the window title of the first visible window
   // whose title is non-empty
   for I := 0 to High(Windows) do
    if IsWindowVisible(Windows[I]) then
     begin
      Result := acUtils.GetWindowTitle(Windows[I]); // retrieve the
title of Window by handle
      if Result <> '' then Exit;
     end;
```

To determinate the title of first visible window of the process — read WindowTitle property.

See also
WindowTitle property.

9.37.5.11WindowTitle

Applies to
acProcess object.

Declaration
```
property WindowTitle: String; // read-only!
```

Description
The WindowTitle property detemines the caption of the first visible window which belongs to process.

To retrieve the full list of windows which belongs to the process — use Windows property.

How does this property works? *(exact code sample)*
```
function TacProcess.GetWindowTitle: String;
var
  I: Integer;
begin
  if AppType = atExplorer then
    Result := 'Windows Explorer'
  else
```

© 1998-2006, UtilMind Solutions®

```
    if CaseAnsiCompareText(ExtractFilename(FExeName), 'systray.exe') = 0
then
      Result := 'System tray'
    else
     begin
      Result := '';
      if High(Windows) >= 0 then // if application have windows
       begin
        // Return the window title of the first visible window
        // whose title is non-empty
        for I := 0 to High(Windows) do
         if IsWindowVisible(Windows[I]) then
          begin
           Result := acUtils.GetWindowTitle(Windows[I]);
           if Result <> '' then Exit;
          end;

        // Nothing yet? Then return the window title of the first
        // window, visible or otherwise, that is non-empty
        for I := 0 to High(Windows) do
         begin
          Result := acUtils.GetWindowTitle(Windows[I]);
          if Result <> '' then Exit;
         end;
       end;
     end;
end;
```

See also
Windows property.

9.37.6 Methods

9.37.6.1 Kill

Applies to
acProcess object.

Declaration
```
function Kill: Boolean;  // returns True if succeed
```

Description
The Kill method can be used to terminate the process associated with the TacProcess object.

Function returns True if succeed or False otherwise.

See also
Description, SystemIconIndex, SystemIcon16Handle and SystemIcon32Handle properties; acSystemImageList component.

9.38 acRSSFeed

9.38.1 TacRSSFeed

Enter topic text here.

© 1998-2006, UtilMind Solutions®

9.39 acRSSFeedContainer

9.39.1 TacRSSFeedContainer

Enter topic text here.

9.40 acSendMail

9.40.1 TacSendMail

Overview

The acSendMail component used for sending e-mail messages via SMTP servers. The messages can be either in plain text or HTML format, contain attachments or *embedded images* for HTML contents. You can use the acSendMail for mass dispatching of messages by mailing lists, specifying multiple recipients.

For more information about SMTP protocol see RFC 821.

How to use?

SMTP server

First you should specify SMTP server which will relay messages to recipients. Set the hostname and port number to SMTPHost and SMTPPort properties. In case if specified SMTP server requires secure authentication — set Authentication.Enabled property to True and provide the Username and Password.

Sender and recipients

The name and email address of sender should be specified in FromName and FromAddress properties (or leave them blank if you sending anonymous message). Recipients should be listed in ToAddr, ToCC or ToBCC properties. (Addresses listed in ToBCC (blind carbon-copy) will not be included to the header of message.)

Message

Set the body of the message to MsgBody and subject of message to MsgSubject property. If you wish to send HTML-formatted message — set MsgContentType to `ctHTML`. If you use foreign language (with non-latin character) — specify content charset in MsgContentCharset property. Also you can specify the priority (level of importance) of message in MsgPriority property.

Attachment

All files which should be attached to message should be listed in Attach property. There are two attachment types — normal and embedded (see AttachType property). Use `ctEmbedded` type if you wish to use some attached pictures in HTML-based message (see also MsgContentType property).

Sending or saving the message

Use Send method to dispatch the message to SMTP server or SaveToFile to save generated message to disk. Use Abort to terminate connection with SMTP server.

Event handlers

Write OnProgress and OnResponse event handlers to display the progress and status of dispatching. Hook OnHostUnreachable, OnConnLost and OnSMTPError events to be notified errors (or use OnAnyError event to be notified about ANY fatal error). Write OnSuccess event to notify user about successful dispatching of e-mail message.

new! *Email templates*

The acSendMail can automatically translate the email templates and replace all the keywords like

© 1998-2006, UtilMind Solutions®

%RECIPIENT_NAME% or %RECIPIENT_EMAIL% to their actual values. You just need to specify some keywords which are used in your template and values which should replace the keywords in the outgoing message. See TemplateConverter property for more details.

Usage example
In Delphi: http://www.appcontrols.com/demos/sendmaildemo-delphi.zip
In C++ Builder: http://www.appcontrols.com/demos/sendmaildemo-bcb.zip
Compiled executable: http://www.appcontrols.com/demos/exe/SendMailDemo.exe

See also
acTextTemplateConverter, acThread and acHTTP components;
SMTP protocol reference (rfc821); MIME references (rfc2045, rfc2046, rfc2047, rfc2048, rfc2049).

9.40.2 Properties

9.40.2.1 AddHeaders

Applies to
acSendMail and acHTTP components.

Declaration
```
property AddHeaders: TStringList;
```

Description
The AddHeaders property specifies additional headers of the e-mail message. You can specify ANY optional headers that may be required by mail client to process and display the message.

Specify one header per line. The format should looks like:
FieldName: Value

For example
X-ResentFrom: <email@address.com>
X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2600.0000
X-Antivir-Status: Clean
and so forth…

See also *(fields automatically included to the header of message)*
Agent, FromAddress, FromName, FromOrganization, MsgContentCharset, MsgContentType, MsgPriority, MsgSubject, ReplyTo, ToAddr and ToCC properties.

9.40.2.2 Agent

Applies to
acSendMail and acHTTP components.

Declaration
```
property Agent: String;
```

Description
The Agent property is the value that specifies the name and type of the e-mail client. You can specify ANY value, this is just optional identifier of the program which sends the message.

The value of Agent property automatically includes to the header of message as "X-Mailer" field. Leave the Agent value blank if you don't want to include the program identifier to the message.

See also *(optional headers)*
AddHeaders and MsgPriority properties.

© 1998-2006, UtilMind Solutions®

9.40.2.3 Attach

Applies to
acSendMail component.

Declaration
```
property Attach: TStringList;
```

Description
The Attach property specifies the file names which should be included to the e-mail message on dispatching.

The Attach property is the list of strings (TStringList), where every line is the file name including full path to the file.

The files can be attached as "external" or as "internal" (embedded) data. For example, you might want to embed additional pictures to the body of HTML message. See AttachType property for more details.

If attached file could not be found (moved, renamed or deleted before sending), the OnCantAttach event occurs. However, this is not fatal error and acSendMail usually continue sending the e-mail, skipping the lost file.

Example
```
procedure TForm1.AttachBtnClick(Sender: TObject);
var
  I: Integer;
begin
  with OpenDialog1 do // TOpenDialog on form
   if Execute and (I <> Files.Count) then
     for I := 0 to Files.Count - 1 do
       acSendMail1.Attach.Add(Files[I]);
end;
```

See also
AttachType property;
OnCantAttach event.

9.40.2.4 AttachType

Applies to
acSendMail component.

Declaration
```
type
  TacSMTPAttachType = (atNormal, atEmbedded);

property AttachType: TacSMTPAttachType;
```

Description
The AttachType property controls how the SendMail should envelop the attached files (listed in Attach property).

There are two possible types of attachment:

Value	Meaning
atNormal	normal attachment. User will be able to save attached files to disk or execute

© 1998-2006, UtilMind Solutions®

attached files directly from message;

atEmbedded embed attached pictures to the body of message and "hide" them. Use this attachment type only for HTML-based message (when MsgContentType = ctHTML). *Note: pictures will be embedded only if you're showing them in HTML-based message (use <img src="cid:picture.gif"> tags).*

If you would like to show embedded pictures in the HTML message, use the filename of attached message as "content-identifer". To show embedded pictures use following tag:

```
<img src="cid:filename"> or...
<body background="cid:filename">
```

Example:

```
<body background="cid:background.jpg">
  <img src="cid:Banner.gif" width=468 height=60>
</body>
```

(*Note: "background.jpg" and ""Banner.gif" files should be attached to the message).*

See also

Attach and MsgContentType properties.

9.40.2.5 Authentication

Applies to

acSendMail component.

Declaration

```
type
  TacSendMailAuthentication = class
  published
    property Enabled: Boolean default False;
    property Username: String;
    property Password: String;
  end;
```

Description

The Authentication structure used to specify whether the SMTP server requires authentication (set Enabled to True, if it requires), and to provide the Username and Password for user authentication.

See also

SMTPHost and SMTPPort properties.

9.40.2.5.1 Enabled

Applies to

acSendMail component as sub-property of Authentication structure.

Declaration

```
property Enabled: Boolean default False;
```

Description

The Enabled property specifies whether the component should send Username and Password for secure authentication with SMTP server.

Set Enabled to True if authentication required, or False otherwise.

See also

Enabled, Username and Password properties.

© 1998-2006, UtilMind Solutions®

9.40.2.5.2 Password

Applies to
acSendMail component as sub-property of Authentication structure.

Declaration
```
property Password: String;
```

Description
The Password property specifies the password required for secure authentication with SMTP server.

See also
Enabled, Username and Password properties.

9.40.2.5.3 Username

Applies to
acSendMail component as sub-property of Authentication structure.

Declaration
```
property Username: String;
```

Description
The Username property specifies the username required for secure authentication with SMTP server.

See also
Enabled, Username and Password properties.

9.40.2.6 Busy

Applies to
acSendMail and acHTTP components.

Declaration
```
property Busy: Boolean; // Read-only !!
```

Description
The Busy property determines whether the SendMail component (its thread) is busy on some operations. When Busy property is True, the SendMail currently sending e-mail messages to recipients.

Notes
You can NOT send any new e-mail message when the component is busy, and must wait until component done all operations. The Send method will return False when the component is busy.

Busy property are useless, in case if your program checking Busy status from the same thread which sends email (when Threaded = False).

See also
Send method; OnSuccess and OnAnyError events;
Threaded property and Running property of acThread component.

© 1998-2006, UtilMind Solutions®

9.40.2.7 FromAddress

Applies to
acSendMail component.

Declaration
```
property FromAddress: String;
```

Description
The FormAddress property specifies the name of sender (i.e: 'john@doe.com'). Leave it blank if you don't want to include the e-mail address to the header of message.

Note
Some SMTP servers don't relay anonymous messages and requires e-mail address of sender to be specified.

See also
FromName, FromOrganization and ReplyTo properties.

9.40.2.8 FromName

Applies to
acSendMail component.

Declaration
```
property FromName: String;
```

Description
The FormName property specifies the name of sender (i.e: 'John Doe'). Leave it blank if you don't want to include the sender's name to the header of e-mail message.

See also
FromAddress and FromOrganization properties.

9.40.2.9 FromOrganization

Applies to
acSendMail component.

Declaration
```
property FromOrganization: String;
```

Description
The FormOrganization property is optional value that specifies the company name of sender. Leave it blank if you don't want to include this value to the header of e-mail message.

If the value of FormOrganization property specified, it automatically includes to the header of message as "Organization" field.

See also
FromAddress and FromName properties.

9.40.2.10 MsgBody

Applies to
acSendMail component.

© 1998-2006, UtilMind Solutions®

Declaration

```
property MsgBody: String;
```

Description

The MsgBody property specifies the text of e-mail message.

The text can be either in plain or HTML format (depending on MsgContentType value). To specify the language of message (content encoding) — use MsgContentCharset property.

See also

MsgSubject and TemplateConverter properties.

9.40.2.11 MsgContentCharset

Applies to

acSendMail component.

Declaration

```
property MsgContentCharset: String; // 'iso-8859-1' by default
```

Description

The MsgContentCharset property determines the charaters set used in the text of e-mail message. This parameter always included to the message to allow the e-mail client to determinate which charset should be used to display the message. Use appropriate definition of charset for language used in email message

Note

You should programmatically handle conversion of text to required code table if the content charset of e-mail is different than charset used on typing a message. Otherwise the text mey appears to be broken. (FYI: 'iso-8859-1' is the standard Western European code table, AFAIK, 'iso-8859-1' is the same as 'us-ascii'.)

The acSendMail automatically converts the text from Windows to KOI8 table ('koi8-r' is the standard Russian charset used in e-mail messages), if the When MsgContentCharset = 'koi8-r'. (So Russian programmers may do not worry about this.)

See also

MsgContentType, MsgBody and MsgSubject properties.

9.40.2.12 MsgContentType

Applies to

acSendMail component.

Declaration

```
type
  TacSMTPContentType = (ctPlainText, ctHTML);

property MsgContentType: TacSMTPContentType;
```

Description

The MsgContentType property specifies the type of message content. The MsgContentType allows to email reader to determinate which format should be used to display the message.

Set MsgContentType to `ctPlainText` if you wish the MsgBody contains only plain text, or set it to `ctHTML` to send HTML-formatted message.

© 1998-2006, UtilMind Solutions®

See also
MsgContentCharset, MsgBody and MsgSubject properties.

9.40.2.13MsgDateTime

Applies to
acSendMail component.

Declaration
```
property MsgDateTime: TDateTime;
```

Description
The MsgDateTime property specifies the date and time of outgoing message.

Usually you don't need to specify this property, because if the MsgDateTime is 0, the component will use current date/time on sending the message. Also the component automatically specifies the shift from GMT, using the international settings specifies on sender's machine.

See also
MsgContentCharset, MsgBody and MsgSubject properties.

9.40.2.14MsgPriority

Applies to
acSendMail component.

Declaration
```
type
  TacSMTPPriority = (mpHighest, mpHigh, mpNormal, mpLow, mpLowest);

property MsgPriority: TacSMTPPriority; // mpNormal by default
```

Description
The MsgPriority property specifies an importance and priority of e-mail message. The priority of message will be shown by e-mail reader program. Also, most SMTP servers relays messages with the higher priorities earlier than messages with low priorities.

The value of MsgPriority property automatically includes to the header of message as "X-Priority" field.

See also *(optional headers)*
AddHeaders and Agent properties.

9.40.2.15MsgSubject

Applies to
acSendMail component.

Declaration
```
property MsgSubject: String;
```

Description
The MsgSubject property specifies the subject of e-mail message.

See also
MsgBody, MsgContentType and MsgContentCharset properties.

© 1998-2006, UtilMind Solutions®

9.40.2.16ReplyTo

Applies to
acSendMail component.

Declaration
```
property ReplyTo: String;
```

Description
The ReplyTo is the optional property which specifies that you want replies to your outgoing messages sent to a different e-mail address, specified in this property. Specify the ReplyTo address if you wish to get replies to different address than specified in FromAddress property.

Usually when the message could not be delivered to recipient it will be returned to the sender, by address specified in FromAddress property. So you can specify "trashcan" address in FromAddress but your real address in ReplyTo.

If the value of ReplyTo property specified, it automatically includes to the header of message as "ReplyTo" field. Leave the ReplyTo value blank if you don't want to include this field to the header of message.

See also
FromAddress property.

9.40.2.17SMTPHost

Applies to
acSendMail component.

Declaration
```
property SMTPHost: String;
```

Description
The SMTPHost property specifies the hostname or IP address of the SMTP server which should be used to relay the e-mail messages to recipients.

See also
SMTPPort property.

9.40.2.18SMTPPort

Applies to
acSendMail component.

Declaration
```
property SMTPPort: Integer; // 25 by default (standard SMTP port)
```

Description
The SMTPPort specifies the port number of SMTP server (specifies the port which listen the server). The standard port number for SMTP protocol is 25.

See also
SMTPHost property.

© 1998-2006, UtilMind Solutions®

9.40.2.19Suspended

Applies to
acSendMail, acHTTP and acThread components.

Declaration
```
property Suspended: Boolean;
```

Description
The Suspended property indicates whether a thread (used for downloading) is currenly suspended.

Set Suspended to True to suspend the process temporary; set it False to resume it.

See also
Suspended property of acThread component

9.40.2.20TemplateConverter

Applies to
acSendMail component.

Declaration
```
property TemplateConverter: TacTextTemplateConverter;
```

Description
The TemplateConverter can be pointed to acTextTemplateConverter component, if you are using the template keywords in your email messages, to translate all the %keywords% in message body and headers of outgoing email, to replace them with actual "Values", specified as the parameters of converter component.

For example, if your message contains keywords like %RECIPIENT_NAME% or %RECIPIENT_EMAIL%, the component will automatically replace them to the actual name and email address of recipient, and the message like
```
To: %RECIPIENT_NAME% <%RECIPIENT_EMAIL%>
Dear %RECIPIENT_NAME%,
etc, etc...
```

will be automatically translated to
```
To: Bill Gates <billgates@microsoft.com>
Dear Bill Gates,
etc, etc...
```

See also
TacTextTemplateConverter component.

9.40.2.21Thread

Applies to
acSendMail and acHTTP components.

Declaration
```
property Thread: TacCustomThread; // Read-only !!
```

Description
The Thread property is the pointer to the process thread, which used for sending the data to SMTP server, avoiding suspending of main application thread. This is read-only public property.

© 1998-2006, UtilMind Solutions®

⚠ This property can be used only in when Threaded property is True, since the component does not create any separate thread in non-threaded model. Otherwise this property will return **nil** (NULL), because the thread not yet created.

See also
Suspended, ThreadPriority and Threaded properties;
acThread component.

9.40.2.22Threaded

Applies to
acSendMail component.

Declaration
```
property Threaded: Boolean; // True by default
```

Description
The Threaded property used to choose between two working models of acSendMail component.

This property specifies whether the message should be prepared and sent via Internet to SMTP server in *current, caller's* thread, or should create *separate* thread which will process the message delivery. Set Threaded to True to create separate thread, or to False if the part of code which calls the Send method are already in separate, non-interface thread.

Basically there is two possible working models:

1. **Threaded**: when Threaded is True, the component creates new separate thread to produce and send email message to specified SMTP host. In this case the acSendMail does not locks the application interface when it connects to the SMTP server and sends the message via Internet, and only produce events synchronized with main application thread (interface).

   ⚠ However, in this model the component can send ONLY ONE message simultaneously, since it can create only one thread to send message. If you will try to send additional message when the component is busy, next email will not be sent, and Send method will just return False. You should implement some message queue to operate with acSendMail in this threading model and don't call Send method while previous message not yet sent. Otherwise — set Threaded to False and use non-threaded model to send any number of email messages simultaneously from any number of different threads.

2. **Non-threaded**: when Threaded is False, the component does not creates any separate thread to produce and send email message. This model should be used only if your application uses many threads which could send email message using this component. This model should NOT be used in single-threaded application, since when you call the Send method, all your application interface will be locked while the component producing and sending the email message.

See also
Thread property;
Send method.

9.40.2.23ThreadPriority

Applies to
acSendMail, acHTTP, acHostnameResolver and acAutoUpgrader components.

Declaration
```
property ThreadPriority: TThreadPriority;
```

© 1998-2006, UtilMind Solutions®

Description

ThreadPriority indicates the priority used when scheduling the thread. Adjust the priority higher or lower as needed.

TThreadPriority type defines the possible values for the Priority property of the acThread component, as defined in the following table. The system schedules CPU cycles to each thread based on a priority scale; the Priority property adjusts a thread's priority higher or lower on the scale.

Values	Meaning
tpIdle	The thread executes only when the system is idle. The system will not interrupt other threads to execute a thread with tpIdle priority.
tpLowest	The thread's priority is two points below normal.
tpLower	The thread's priority is one point below normal.
tpNormal	The thread has normal priority.
tpHigher	The thread's priority is one point above normal.
tpHighest	The thread's priority is two points above normal.
tpTimeCritical	The thread gets highest priority.

Warning

Boosting the thread priority of a CPU intensive operation may "starve" the other threads in the application. Only apply priority boosts to threads that spend most of their time waiting for external events.

See also

Thread, Threaded and WaitThread properties;
acThread component.

9.40.2.24ToAddr

Applies to

acSendMail component.

Declaration

```
property ToAddr: String;
```

Description

The ToAddr property specifies the recipients of the message. You can specify any number of e-mail addresses and recipient names.

The value of ToAddr property must have following format:
"Name1" <address1>; "Name2" <address2> , "NameN" <addressN>

The name of the recipient marked blue (must be between quotes ("), i.e: "John Smith"). You can leave the name blank and use just email address;
E-mail address of recipient marked green (must be between brackets "<>", i.e: <john@smith.com>).
Multiple recipients should be separated by semicolon (;) or comma (,).

Examples

<john@smith.com>
"John Smith" <john@smith.com>, <info@appcontrols.com>, <webmaster@utilmind.com>
"Aleksey Kuznetsov" <aleksey@utilmind.com>; "Webmaster" <webmaster@utilmind.com>

Alternatively, if you wish to specify multiple recipients, you can use ToCC property (carbon-copy). If you wish to hide some email addresses from message header — use ToBCC property (blind carbon-copy).

© 1998-2006, UtilMind Solutions®

Warning

Please don't use brackets (< and >) in the "Name" fields (between quotes). Otherwise the address will be parsed incorrectly. For example, following address will be parsed incorrectly!!:
"John<g> Smith" <john@smith.com>

However, the parser is smart enough to parse definitions like these:
"John Smith, Inc;" <john@smith.com>
"Smith, "A" John" <john@smith.com>

See also

ToCC and ToBCC properties.

9.40.2.25ToBCC

Applies to

acSendMail component.

Declaration

```
property ToBCC: String;
```

Description

The ToBCC property used to specify recipients of the message which addresses should NOT be visible in the message header. This is field known as blind carbon-copy and only the message sender knows to whom the message was sent. All e-mails used in ToBCC will be hidden and recipients will NOT see these e-mail addresses.

The value of ToBCC property must have following format:
"Name1" <address1>; "Name2" <address2> , "NameN" <addressN>

The name of the recipient marked blue (must be between quotes ("), i.e: "John Smith"). You can leave the name blank and use just email address;
E-mail address of recipient marked green (must be between brackets "<>", i.e: <john@smith.com>).
Multiple recipients should be separated by semicolon (;) or comma (,).

Examples

<john@smith.com>
"John Smith" <john@smith.com>, <info@appcontrols.com>, <webmaster@utilmind.com>
"Aleksey Kuznetsov" <aleksey@utilmind.com>; "Webmaster" <webmaster@utilmind.com>

Alternatively, if you wish to specify multiple recipients, you can use ToCC property (visible carbon-copy).

Warning

Please don't use brackets (< and >) in the "Name" fields (between quotes). Otherwise the address will be parsed incorrectly. For example, following address will be parsed incorrectly!!:
"John<g> Smith" <john@smith.com>

However, the parser is smart enough to parse definitions like these:
"John Smith, Inc;" <john@smith.com>
"Smith, "A" John" <john@smith.com>

See also

ToAddr and ToCC properties.

© 1998-2006, UtilMind Solutions®

9.40.2.26ToCC

Applies to
acSendMail component.

Declaration
```
property ToCC: String;
```

Description
The ToCC property used to specify multiple recipients of the message. This is standard field known as carbon-copy and recipient will be able to reply to all others recipients.

The value of ToCC property must have following format:
"Name1" <address1>; "Name2" <address2> , "NameN" <addressN>

The name of the recipient marked blue (must be between quotes ("), i.e: "John Smith"). You can leave the name blank and use just email address;
E-mail address of recipient marked green (must be between brackets "<>", i.e: <john@smith.com>).
Multiple recipients should be separated by semicolon (;) or comma (,).

Examples
<john@smith.com>
"John Smith" <john@smith.com>, <info@appcontrols.com>, <webmaster@utilmind.com>
"Aleksey Kuznetsov" <aleksey@utilmind.com>; "Webmaster" <webmaster@utilmind.com>

If you wish to hide recipients from the message header — use ToBCC property (blind carbon-copy).

Warning
Please don't use brackets (< and >) in the "Name" fields (between quotes). Otherwise the address will be parsed incorrectly. For example, following address will be parsed incorrectly!!:
"John<g> Smith" <john@smith.com>

However, the parser is smart enough to parse definitions like these:
"John Smith, Inc;" <john@smith.com>
"Smith, "A" John" <john@smith.com>

See also
ToAddr and ToBCC properties.

9.40.2.27WaitThread

Applies to
acSendMail, acHTTP, acThread and acSQLQueryThread components.

Declaration
```
property WaitThread: Boolean;
```

Description
The WaitThread property controls whether the procedure that calls the Send method (which downloads the data from the Web) should be suspended and wait until the scanning process will be done.

Set the WaitThread to True, if you would like to read the data from the Web so that the application does not continue with next lines of code after calling the Send method. Your application will done download (or inform about error) before continuing to next step.

© 1998-2006, UtilMind Solutions®

If your application can wait, until the the email will be sent, only for some limited time interval — specify WaitTimeout property.

See also
WaitTimeout, Thread, Threaded and Suspended properties;
Send method.

9.40.2.28 WaitTimeout

Applies to
acSendMail, acHTTP, acThread and acSQLQueryThread components.

Declaration
```
property WaitTimeout: Integer;
```

Description
The WaitTimeout property specifies the time interval (limit), in milliseconds unit, which application able to wait while the mail will be sent.

For example, if the maximum time which you can allow to send an email is 5 seconds, set this value to 5000 (milliseconds). If application can wait infinitely, set WaitTimeout to 0.

When the timeout is expired, the component automatically terminates the process of sending an email. To be notified when the WaitTimeout is expired — write OnWaitTimeoutExpired event handler.

Notes
The WaitTimeout only works together with WaitThread property, only when it set to True.

See also
WaitThread, Threaded, ThreadPriority and Suspended properties;
Send and Abort methods;
OnWaitTimeoutExpired event.

9.40.3 Methods

9.40.3.1 SaveToFile

Applies to
acSendMail component.

Declaration
```
function SaveToFile(const FileName: String): Boolean;
```

Description
The SaveToFile property generates the e-mail message (including encoded attachment) and stores it to the file (specified in *FileName* parameter). Set *Unsent* parameter to True to add "X-Unsent: 1" field (which means that message not yet sent) to the stored message.

Function returns True if message has successfully saved, or False if it could not be stored by specified *FileName*.

The created message has standard format (.eml) which can be easily viewed and sent by MS-Outlook e-mail clients.

See also

© 1998-2006, UtilMind Solutions®

Send property.

9.40.3.2 Send

Applies to
acSendMail component.

Declaration
```
function Send: Boolean; // returns False if busy or WaitTimeout is expired
```

Description
The Send method generates and sends the e-mail message to all recipients specified in ToAddr, ToCC and ToBCC properties.

If you are using threaded model (Threaded property is True), this method may returns False if component currently busy (already sending the message), OR WaitTimeout is expired (if you waiting for completion of request in the function that calls this medod, using WaitThread property).

See also
Abort method;
Threaded, ToAddr, ToCC and ToBCC properties.

9.40.3.3 Abort

Applies to
acSendMail, acHTTP, acAutoUpgrader, acWebImage and acSQLQueryThread components.

Declaration
```
procedure Abort(HardTerminate: Boolean = True);
```

Description
The Abort method imediately terminates the thread which sends the data via SMTP protocol. After calling the Abort method, the OnAborted event occurs.

The Abort method contains 1 optional parameter, *HardTerminate* (True by default), which controls whether the application should imediately terminate thread which sends data to SMTP server, or wait when server terminate connection itself.

See also
Send method; OnAborted event.

9.40.4 Events

9.40.4.1 OnAborted

Applies to
acSendMail, acHTTP, acAutoUpgrader and acWebImage components.

Declaration
```
property OnAborted: TNofifyEvent;
```

Description
The OnAborted event occurs when user interrupts the process of sending the e-mail to SMTP server, after calling the Abort method.

See also

© 1998-2006, UtilMind Solutions®

Abort method;
OnWaitTimeoutExpired event.

9.40.4.2 OnAnyError

Applies to
acSendMail and acHTTP components.

Declaration
```
property OnAnyError: TNotifyEvent;
```

Description
The OnAnyError event occurs when ANY fatal error has occured on delivering the message to server *(except OnCantAttach)*.

The fatal error means that message could not be delivered. Such errors can be also handled by following events: OnConnLost, OnHostUnreachable and OnSMTPError.

See also
OnConnLost, OnHostUnreachable and OnSMTPError events.

9.40.4.3 OnCantAttach

Applies to
acSendMail component.

Declaration
```
type
  TacSMTPCantAttachEvent = procedure(Sender: TObject; FileName: String)
of object;

property OnCantAttach: TacSMTPCantAttachEvent;
```

Description
The OnCantAttach event occurs if the component could not read the file which should be attached (listed in Attach property).

This is not fatal error, so SendMail will try to continue sending the message without missed file. If you don't want to continue process — just call Abort method in the event handler.

See also
Attach property; Abort method.

9.40.4.4 OnConnLost

Applies to
acSendMail and acHTTP components *(but with different parameters)*.

Declaration
```
property OnConnLost: TNotifyEvent;
```

Description
The OnConnLost event occurs when the connection with remote server lost for some reason, at the moment of sending or receiving the data by SMTP protocol.

This is fatal error and OnAnyError event will occured after this event.

© 1998-2006, UtilMind Solutions®

See also
OnAnyError event.

9.40.4.5 OnHostUnreachable

Applies to
acSendMail, acHTTP and acAutoUpgrader components.

Declaration
```
property OnHostUnrachable: TNotifyEvent;
```

Description
The OnHostUnreachable event occurs if the acSendMail can not establish connection to SMTP server, specified in SMTPHost property. Possible reasons of this problem is:

1. User currently not connected to the Internet;
2. Hostname is unknown (check spelling of domain name in SMTPHost property);
3. Invalid SMTP port specified (check the number of SMTP port in SMTPPort property);
4. Remote server is down (disconnected from Internet).

Example
Delphi:
```
procedure TForm1.acSendMail1HostUnreachable(Sender: TObject);
begin
  Application.MessageBox(PChar('Specified SMTP server is
unreachable.'#13#10#10'Please check your Internet connection'#13#10'or
specify another SMTP host or/and port.'),
                         PChar(Application.Title),
                         MB_OK or MB_ICONSTOP);
end;
```

C++ Builder:
```
void __fastcall TForm1::acSendMail1HostUnreachable(TObject *Sender)
{
  AnsiString Msg =
      "Specified SMTP server is unreachable.\n\n"
      "Please check your Internet connection\n"
      "or specify another SMTP host or/and port.";
   Application->MessageBox(Msg.c_str(),
                           Application->Title.c_str(),
                           MB_OK | MB_ICONSTOP);
}
```

See also
SMTPHost and SMTPPort properties;
OnAnyError event.

9.40.4.6 OnProgress

Applies to
acSendMail, acHTTP and acAutoUpgrader components *(but with different parameters)*.

Declaration
```
type
  TacSMTPProgressEvent = procedure(Sender: TObject; MessageSize,
BytesSent: Integer; PercentsDone: Byte) of object;

property OnProgress: TacSMTPProgressEvent;
```

© 1998-2006, UtilMind Solutions®

Description

The OnProgress event occurs every time when the component has successfully sent the part of e-mail message to SMTP server.

Write OnProgress event handler to show the download progress. The acSendMail passes to the event handler following parameters:

Parameter	Meaning
MessageSize	total size of e-mail message in bytes;
BytesSent	number of bytes that already sent to SMTP server;
PercentsDone	progress in percents (0%..100%).

Example

```
procedure TForm1.acSendMail1Progress(Sender: TObject; MessageSize,
  BytesSent: Integer; PercentsDone: Byte);
begin
  ProgressBar1.Position := PercentsDone;
end;
```

See also

OnSuccess event.

9.40.4.7 OnResponse

Applies to

acSendMail component.

Declaration

```
type
  TacSMTPResponseEvent = procedure(Sender: TObject; Code: Integer; const
Response: String) of object;

property OnResponse: TacSMTPResponseEvent;
```

Description

The OnResponse event occurs every time when component receives the response for SMTP request from server. The response contains the status information (code and optional text) of request.

For more information about status codes of SMTP protocol see RFC 821.

Example

```
procedure TForm1.acSendMail1Response(Sender: TObject; Code: Integer;
  const Response: string);
begin
  with ListView1 do // status view
   begin
    Items.Add(Response);
    ItemIndex := Items.Count - 1;
   end;
end;
```

See also

OnSMTPError, OnSuccess and OnProgress events.

© 1998-2006, UtilMind Solutions®

9.40.4.8 OnSMTPError

Applies to
acSendMail component.

Declaration
```
type
  TacSMTPResponseEvent = procedure(Sender: TObject; Code: Integer; const
Response: String) of object;

property OnSMTPError: TacSMTPResponseEvent;
```

Description
The OnSMTPError event occurs when the SMTP server returns an error code with response (see OnResponse event). After receiving error code, the acSendMail component automatically terminates connection with SMTP server, then triggers the OnSMTPError event.

For more information about status codes of SMTP protocol see RFC 821.

Example
```
procedure TForm1.acSendMail1SMTPError(Sender: TObject;
  Code: Integer; const Response: string);
begin
  with StatusList do
   begin
    Items.Add('ERROR: ' + Response);
    ItemIndex := Items.Count - 1;
   end;
end;
```

See also
OnResponse and OnAnyError events.

9.40.4.9 OnSuccess

Applies to
acSendMail component.

Declaration
```
property OnSuccess: TNotifyEvent;
```

Description
The OnSuccess event occurs when the e-mail message has been successfully sent to SMTP server. Use OnSuccess event to be notified when the SendMail successfuly dispatched the email message to SMTP server (message will be relayed to recipients shortly by server).

Note
Successful dispatching of message to SMTP server not gives guarantee of successful delivery to recipients. SMTP server may return the message if it could not be relayed to the mailbox.

See also
OnResponse and OnSMTPError events.

9.40.4.10 OnWaitTimeoutExpired

Applies to
acSendMail, acHTTP, acThread and acSQLQueryThread components.

© 1998-2006, UtilMind Solutions®

Declaration

```
property OnWaitTimeoutExpired: TNotifyEvent;
```

Description

The OnWaitTimeoutExpired event occurs when the component could not send email for the time interval specified in WaitTimeout property (when WaitTimeout is expired).

The *TherminateThread* is optional parameter (True by default), which allows to terminate, or prevent termination of thread in case if specified time interval is expired before completion of execution of the thread.

Notes

The WaitTimeout only works together with WaitThread property, only when it set to True.

See also

Theaded, WaitThread and WaitTimeout properties;
Abort method.

9.40.5 QuickSendMail function

Unit

acSendMail

Declaration

```
function SendMail(const FromAddr, FromName, ToAddr, ToName, Subject,
Body: String;
  const SMTPHost: String = 'localhost'; SMTPPort: Word = 25;
  ContentType: TSMTPContentType = ctHTML; Priority: TSMTPPriority =
mpNormal;
  const ContentCharset: String = 'iso-8859-1'; RunInSeparateThread:
Boolean = True): Boolean;
```

Description

The SendMail function is alternative way to send email messages without having the acSendMail component on the form, and without dynamic creation of acSendMail component instances. When you use this function — everything created/released internally.

The SendMail have following set of parameters (some of parameters are optional and may be not specified):

Parameter	Meaning
FromAddr	specifies the email address of sender;
FromName	specifies the name of sender (the name which displayed in email message, can be empty string);
ToAddr	email address of recipient;
ToName	name of recipient (can be empty string);
Subject	the subject of email message;
Body	the text of email message (can contain HTML tags if *ContentType* is ctHTML, or just plain text if *ContentType* is ctPlainText);
SMTPHost	the hostname or IP address of SMTP server;
SMTPPort	the port number of SMTP server (standard port for SMTP servers is 25);

© 1998-2006, UtilMind Solutions®

ContentType	can be ctPlainText if your email message contains just plain text, or ctHTML if it should be HTML-formatted;
Priority	priority of email message (npNormal by default);
ContentCharset	the set of characters of text in email message. Use appropriate definition of charset for language used in email message;
RunInSeparateThread	specifies whether the message should be prepared and sent via Internet to SMTP server in *current, caller's* thread, or should create *separate* thread which will process the message delivery. Set this parameter to True to create separate thread, or to False if the part of code which calls the SendMail function are already in separate, non-interface thread.

See also
acSendMail component.

9.41 acShortcut

9.41.1 TacShortcut

Overview
The **acShortcut** component lets you to quickly and easily create a shortcuts (also known as shell links) to files and folders (in any location) and read the information from shorcut files (.lnk).

How to read the information from shortcut ?
Just point the **FileName** property to the .lnk file and read the information from other properties:

LinkTarget	Points to the file which should be opened on starting s shortcut (on clicking the shortcut icon);
Parameters	The command-line parameters for executable file specified by LinkTarget property. (if LinkTarget is an EXE);
WorkingDirectory	Working directory for executable file specified in LinkTarget property. (if LinkTarget is an EXE);
RunAs	Specifies how the window should be opened: minimized, maximized or in its normal state;
IconLocation, IconIndex	Points the location to icon file or another file which contains the icon resources. IconIndex property specifies the resource index, if IconLocation is an EXE or DLL;
HotKey	Specifies the key combination which executes a shortcut;
LargeIcon, SmallIcon	Read-only properties. Contains the 32x32 and 16x16 icons of shortcut link.

The FILENAME property works either at run- or design-time, so you can play with acShortcut component, pointing the .lnk file at design-time. After specifyng the FILENAME, all other properties will be filled with information about the shortcut.

How to create a shortcut ?
To create a shortcut you must specify the shortcut information to according properties and call **SaveToFile(FileName)** method, where the FileName is the name of shortcut. Function returs False when it fails.

© 1998-2006, UtilMind Solutions®

For example, you would like to create a shortcut file on desktop, which points to your current executable file + want to specify the icon *#4* from *shell32.dll* file.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  acShortcut1.LinkTarget := Application.ExeName;

  // next two lines will set icon #4 from shell32.dll
  acShortcut1.IconLocation := acSystemInfo1.DirSystem +
'\shell32.dll';
  acShortcut1.IconIndex := 4;

  // saving to Test.lnk file on desktop
  acShortcut1.SaveToFile(acSystemInfo1.DirDesktop + '\Test.lnk');
end;
```

How to delete shortcut ?

Shortcuts is usual files, with .LNK extension. To remove a shortcuts — just delete them using standard routines, such like *DeleteFile*…

See also

acSystemInfo component.

9.41.2 Properties

9.41.2.1 FILENAME

Applies to

acShortcut component.

Declaration

```
property FILENAME: String;
```

Description

The FILENAME property specifies the name of shortcut file, to read the information from the shell link (.LNK file). At once after pointing the FILENAME to the shortcut, the acShortcut component will read all shortcut information to fill other properties with determinated values.

The FILENAME property is an alternative way to read the shortcut information and does the same as LoadFromFile method. However, unlike the LoadFromFile, the FILENAME property works either at run- or design-time, so you can play with the component specifying the .LNK files and reading shortcut information even at design-time.

See also

LinkTarget and WorkingDirectory properties.

9.41.2.2 HotKey

Applies to

acShortcut component.

Declaration

```
property HotKey: TShortCut;
```

Description

The HotKey property determines and specifies the key combination which executes a shortcut.

© 1998-2006, UtilMind Solutions®

To convert a text string to the HotKey and vice versa, you can use TextToShortCut and ShortCutToText functions from standard "Menus" unit.

9.41.2.3 IconIndex

Applies to
acShortcut and acExtAssociation components.

Declaration
```
property IconIndex: Integer;
```

Description
The IconIndex property determines or specifies the index of icon within the icon file, specified by IconLocation property.

See also
IconLocation property.

9.41.2.4 IconLocation

Applies to
acShortcut and acExtAssociation components.

Declaration
```
property IconLocation: String;
```

Description
The IconLocation property determines or specifies the location of the file which contains the icon for the shortcut. The icon file can be an EXE, DLL or ICO. To determinate or specify the icon index within the icon file, use IconIndex property.

See also
IconIndex property.

9.41.2.5 LargeIcon

Applies to
acShortcut and acExtAssociation components.

Declaration
```
property LargeIcon: TIcon; // read only !!
```

Description
The SmallIcon property retrieves the 32x32 shortcut icon associated with an .LNK file.

The icon will be retrieved from the file specified in IconLocation property, by index specified in IconIndex property.

See also
SmallIcon property.

9.41.2.6 LinkTarget

Applies to
acShortcut component.

Declaration

© 1998-2006, UtilMind Solutions®

```
property LinkTarget: String;
```

Description
The LinkTarget property determines or specifies the location of the shortcut target. The LinkTarget is the application which should be executed when user starts the the shortcut (clicks the shortcut icon).

See also
Parameters, WorkingDirectory and RunAs properties.

9.41.2.7 Parameters

Applies to
acShortcut component.

Declaration
```
property Parameters: String;
```

Description
The Parameters property determines or specifies the command-line parameters transmitted to the executable file, specified by the LinkTarget property, on starting the shortcut.

See also
LinkTarget property.

9.41.2.8 RunAs

Applies to
acShortcut component.

Declaration
```
type
  TacShortcutRunAs = (raNormal, raMinimized, raMaximized);

property RunAs: TacShortcutRunAs;
```

Description
The RunAs parameter determines or specifies how the the application, executed by shortcut should be started.

Value	Meaning
raNormal	starts the application in normal-size window;
raMinimized	minimize the application minimized;
raMaximized	maximize the application on startup.

See also
LinkTarget property.

9.41.2.9 ShortDescription

Applies to
acShortcut component.

Declaration
```
property ShortDescription: String;
```

Description

© 1998-2006, UtilMind Solutions®

The ShortDescription property determines or specifies the description of the shortcut.

This is not obligatory, optional shortcut parameter and, in most cases, can be empty.

9.41.2.10 SmallIcon

Applies to
acShortcut and acExtAssociation components.

Declaration
```
property SmallIcon: TIcon; // read only !!
```

Description
The SmallIcon property retrieves the 16x16 shortcut icon associated with an .LNK file.

The icon will be retrieved from the file specified in IconLocation property, by index specified in IconIndex property.

See also
LargeIcon property.

9.41.2.11 WorkingDirectory

Applies to
acShortcut component.

Declaration
```
property WorkingDirectory: String;
```

Description
The WorkingDirectory property determines and specifies the working directory for application which will be executed when user clicks a shortcut.

This is the working directory for the program specified in the LinkTarget property.

See also
LinkTarget property.

9.41.3 Methods

9.41.3.1 LoadFromFile

Applies to
acShortcut component.

Declaration
```
function LoadFromFile(const FileName: String): Boolean;
```

Description
The LoadFromFile method reads the shortcut information from an .LNK file and fills the properties with determined values.

Alternatively you can read the information from shortcut file just pointing the FileName property to .LNK file. Unlike the LoadFromFile method, the FileName property can be specified either at run- or design-time, so you can play with the component specifying the .LNK files and reading shortcut information even at design-time.

© 1998-2006, UtilMind Solutions®

See also
FileName property and SaveToFile method.

9.41.3.2 SaveToFile

Applies to
acShortcut component.

Declaration
```
function SaveToFile(const FileName: String): Boolean;
```

Description
The SaveToFile method creates the shell link (shortcut file), in location specified by *FileName* parameter, and writes the shortcut information specified in various properties of acShortcut component.

See also
LoadFromFile method and example.

9.42 acThread

9.42.1 TacThread

Overview
The acThread component is enhancement of standard invisible TThread class. Extremely easy to use, since you can specify properties and events directly in Object Inspector as in any usual component, and contains much more features.

How to use ?
For example, you need to run process which will take a lot of time to be completed. This can be anything, sorting big arrays, copying, searching or downloading the data and so forth… Anything that will take rather much time but should not stop and even slow down user interface and other processes…

Well, simply drop the acThread component onto your form and specify all actions of this thread in the OnExecute event handler. To execute thread — call Execute method. To terminate — call Terminate. If you wish to suspend thread temporary, without total termination — use Suspended property (set it to True to suspend and to False to resume thread), or Suspend/Resume methods.

If your thread needs to send output to the main VCL thread, or just make some specific actions in your forms (i.e: fill the list view or increase position of the progress bar), you must synchronize actions between different threads to avoid multi-thread conflicts. To synchronize threads and complete some operations in main VCL thread — use Synchronize method.

To change scheduling priority of thread — use Priority property. To be notified when thread terminated — write it to the OnTerminate event handler

Sample code
You can download the example which demonstrates how to use acThread component from http://www.appcontrols.com/demos/threaddemo.zip, or compiled executable from http://www.appcontrols.com/demos/exe/ThreadDemo.exe.

See also other components with built-in acThread component:
acSQLQueryThread, acHTTP and acAutoUpgrader.

© 1998-2006, UtilMind Solutions®

9.42.2 Properties

9.42.2.1 Handle

Applies to
acThread component.

Declaration
```
property Handle: THandle; // read-only!
```

Description
Handle is the read-only property which contains the thread's handle. Use Handle when calling Win32 API functions for thread manipulation.

See also
ThreadID property.

9.42.2.2 HandleExceptions

Applies to
acThread and acSQLQueryThread components.

Declaration
```
property HandleExceptions: Boolean; // True by default
```

Description
The HandleException property determines whether the thread should automatically handle all exceptions which occurs on it's execution, without raising exception, or you would like to handle exceptions "manually" in OnExecute event handler.

Set HandleException to True if you don't want to allow to raise an exceptions and/or handle exceptions in OnException event. Set HandleException to False if you wish to handle exceptions within OnExecute event handler (see excample below).

Example
```
{ "manual" handling of exceptions within OnExecute event handler (when HandleException = False) }
procedure TForm1.QueryThreadExecute(Sender: TObject);
begin
  with Query1 do
   try
     ExecSQL;
   except
     // handle exception "manually"
   end;
end;
```

See also
OnException and OnExecute events.

9.42.2.3 Priority

Applies to
acThread and acSQLQueryThread components.

Declaration
```
property Priority: TThreadPriority;
```

© 1998-2006, UtilMind Solutions®

Description
Priority indicates the priority used when scheduling the thread. Adjust the priority higher or lower as needed.

TThreadPriority type defines the possible values for the Priority property of the acThread component, as defined in the following table. The system schedules CPU cycles to each thread based on a priority scale; the Priority property adjusts a thread's priority higher or lower on the scale.

Values	Meaning
tpIdle	The thread executes only when the system is idle. The system will not interrupt other threads to execute a thread with tpIdle priority.
tpLowest	The thread's priority is two points below normal.
tpLower	The thread's priority is one point below normal.
tpNormal	The thread has normal priority.
tpHigher	The thread's priority is one point above normal.
tpHighest	The thread's priority is two points above normal.
tpTimeCritical	The thread gets highest priority.

Warning
Boosting the thread priority of a CPU intensive operation may "starve" the other threads in the application. Only apply priority boosts to threads that spend most of their time waiting for external events.

9.42.2.4 ReturnValue

Applies to
acThread component.

Declaration
```
property ReturnValue: Integer;
```

Description
The ReturnValue property specifies the value returned to other waiting threads when the thread finishes executing.

Use ReturnValue to indicate success/failure or numeric result/output of the thread to the application or other threads. The WaitFor method returns the value stored in ReturnValue.

See also
WaitFor method.

9.42.2.5 Running

Applies to
acThread component.

Declaration
```
property Running: Boolean; // Read only !!
```

Description
Running is the read-only property used to determinate whether the thread is currently running and processes operations specified in the OnExecute event handler.

See also
Suspended property and OnExecute method.

© 1998-2006, UtilMind Solutions®

9.42.2.6 Suspended

Applies to
acThread, acHTTP and acSendMail components.

Declaration
```
property Suspended: Boolean;
```

Description
The Suspended property indicates whether a thread is suspended.

Set Suspended to True to suspend a thread (suspend processing of OnExecute event handler); set it to False to resume it. Suspended threads do not continue execution until they are resumed.

See also
Suspend and Resume methods.
OnExecute event.

9.42.2.7 Terminated

Applies to
acThread component.

Declaration
```
property Terminated: Boolean;
```

Description
The Terminated property Indicates whether the thread has been asked to terminate.

The thread's Execute method and any methods that Execute calls (in OnExecute event handler) should check Terminated periodically and exit when it's True. The Terminate method sets the Terminated property to True.

The Terminate method is the polite way to abort the execution of a thread, but it requires cooperation from the thread's OnExecute code. Using Terminate is recommended over the TerminateThread Win32 API call.

See also
Execute and Terminate methods;
OnExecute event.

9.42.2.8 ThreadID

Applies to
acThread component.

Declaration
```
property ThreadID: THandle; // Read only !!
```

Description
The ThreadID is the read-only handle which identifies the thread throughout the system.

Use ThreadID, during debugging, to identify the thread in the Threads status box. ThreadID is also useful when calling Win32 API functions for manipulating the thread.

Note

© 1998-2006, UtilMind Solutions®

ThreadID is different than the thread's handle in the Handle property.

See also
Handle property.

9.42.2.9 WaitThread

Applies to
acThread, acSQLQueryThread, acHTTP and acSendMail components.

Declaration
```
property WaitThread: Boolean;
```

Description
The WaitThread property controls whether the procedure that calls the Execute method should be suspended and wait until the scanning process will be done.

Set the WaitThread to True, if you would like to process some operations in separate thread so that the application does not continue with next lines of code after calling the Execute method. Your application will done execution of the thread *before* continuing to next step.

If your application can't infinitely wait, until the the thread execution will be completed, and want to set for some limited time interval for the execution — specify WaitTimeout property.

See also
WaitTimeout and Suspended properties;
Execute method.

9.42.2.10 WaitTimeout

Applies to
acHTTP, acSendMail, acThread and acSQLQueryThread components.

Declaration
```
property WaitTimeout: Integer;
```

Description
The WaitTimeout property specifies the time interval (limit), in milliseconds unit, which application able to wait until the thread will be terminated.

For example, if the maximum time which you can allow for execution of this thread is 5 seconds, set this value to 5000 (milliseconds). If application can wait infinitely, set WaitTimeout to 0.

When the timeout is expired, the component automatically terminates the HTTP request. To be notified when the WaitTimeout is expired — write OnWaitTimeoutExpired event handler.

Notes
The WaitTimeout only works together with WaitThread property, only when it set to True.

See also
WaitThread, ThreadPriority and Suspended properties;
Execute and Execute methods;
OnWaitTimeoutExpired event.

© 1998-2006, UtilMind Solutions®

9.42.3 Methods

9.42.3.1 Execute

Applies to
acThread component.

Declaration
```
function Execute: Boolean; // returns False if WaitTimeout expired
```

Description
Call Execute method to start the thread and initiate the processes described in the OnExecute event handler.

The Execute method will terminate the thread (if it already Running) and restart current thread again.

Function can return False if WaitTimeout is expired (if you waiting for completion of request in the function that calls this medod, using WaitThread property).

See also
WaitThread and WaitTimeout properties;
Suspend, Resume, Synchronize and Terminate methods;
OnExecute event.

9.42.3.2 Resume

Applies to
acThread component.

Declaration
```
procedure Resume;
```

Description
The Resume method resumes a paused (suspended) thread and continue its execution.

See also
Suspended property;
Suspend, Execute, Synchronize and Terminate methods.

9.42.3.3 Suspend

Applies to
acThread component.

Declaration
```
procedure Suspend;
```

Description
The Suspend method pauses a running thread.

Call Suspend to temporarily halt execution of the thread. To resume execution after a call to Suspend, call Resume (or set Suspended property to True). Calls to Suspend can be nested; Resume must be called the same number of times Suspend was called before the thread will resume execution.

See also

© 1998-2006, UtilMind Solutions®

Suspended property;
Execute, Resume, Synchronize and Terminate methods.

9.42.3.4 Synchronize

Applies to
acThread component.

Declaration
```
procedure Synchronize(Method: TThreadMethod);
```

Description
The Synchronize method executes a method call within the main VCL thread. Synchronize causes the call specified by *Method* to be executed using the main VCL thread, thereby avoiding multi-thread conflicts. If you are unsure whether a method call is thread-safe, call it from within the main VCL thread by passing it to the Synchronize or SynchronizeEx methods.

Execution of the thread is suspended while *Method* is executing in the main VCL thread.

Example
```
procedure TForm1.SynchronizedOutput;
begin
  OutputBox1.Items.Add(Variable1);
  Label1.Caption := Variable2;
end;

procedure TForm1.acThread1Execute(Sender: TObject);
begin
  //...
  StrVariable := StrData1;
  IntVariable := IntData2;
  Synchronize(SynchronizedOutput);
  //...
end;
```

See also
SynchronizeEx, Execute, Suspend, Resume and Terminate methods.

9.42.3.5 SynchronizeEx

Applies to
acThread component.

Declaration
```
procedure SynchronizeEx(Method: TNotifyEvent; Params: Pointer);
```

Description
The SynchronizeEx method executes a method call within the main VCL thread. Synchronize causes the call specified by *Method* to be executed using the main VCL thread, thereby avoiding multi-thread conflicts. If you are unsure whether a method call is thread-safe, call it from within the main VCL thread by passing it to the Synchronize or SynchronizeEx methods.

Internally, the SynchronizeEx works exactly like Synchronize, but allows to tramsmit additional parameters (*Params*) to the synchronized method. See example below.

Execution of the thread is suspended while *Method* is executing in the main VCL thread.

© 1998-2006, UtilMind Solutions®

Example

```
type
  TForm1 = class(TForm)
    acThread1: TacThread;
    procedure acThread1Execute(Sender: TObject);
  private
    procedure SyncCall(Sender: TObject);
  public
  end;

implementation

type
  PMyData = ^TMyData;
  TMyData = record
    I: Integer;
    S: String;
  end;

procedure TForm1.acThread1Execute(Sender: TObject);
var
  Data: TMyData;
begin
  // -- snip --
  with Data do
   begin
    S := 'some text';
    I := Random(100); // some number
   end;
  acThread1.SynchronizeEx(SyncCall, @Data);
  // -- snip --
end;

procedure TForm1.SyncCall(Sender: TObject);
begin
  with PMyData(Sender)^ do
   begin
    // Here you getting access to the parameters
    // described in TMyData structure. Example:
    Application.MessageBox(PChar(S), PChar(IntToStr(I)), MB_OK);
   end;
end;
```

See also

Synchronize, Execute, Suspend, Resume and Terminate methods.

9.42.3.6 Terminate

Applies to

acThread component.

Declaration

```
procedure Terminate(Imediately: Boolean);
```

Description

The Terminate method aborts execution of thread.

© 1998-2006, UtilMind Solutions®

See also
Execute, Suspend, Resume and Synchronize methods.

9.42.3.7 WaitFor

Applies to
acThread component.

Declaration
```
function WaitFor: Integer;
```

Description

See also
Execute, Suspend, Resume, Synchronize and Terminate methods.

9.42.4 Events

9.42.4.1 OnException

Applies to
acThread, acSQLQueryThread and acAppEvents components.

Declaration
```
property OnException: TNotifyEvent;
```

Description
The OnException event occurs if something has been failed on execution of thread, in the OnExecute event handler.

Note that OnException event will NOT occurs if HandleExceptions property = False, that means that you would like to handle exceptions "manually" in OnExecute event handler.

See also
OnExecute and OnTerminate events;
HandleExceptions property.

9.42.4.2 OnExecute

Applies to
acThread component.

Declaration
```
property OnExecute: TNotifyEvent;
```

Description
The OnExecute event occurs when the thread ready to execute new process.

Write OnExecute event handler to make some specific operations in the separate process without suspending of main application thread (without blocking of user interface).

Example 1:
```
procedure TForm1.ConnectThreadExecute(Sender: TObject);
begin
{ this can take very long time and suspend the application }
  mySQLDatabase1.Connected := True;
end;
```

© 1998-2006, UtilMind Solutions®

Example 2:

```
procedure TForm1.acThread1Execute(Sender: TObject);

  procedure ScanDir(Dir: String);
  var
    FindHandle: THandle;
    FindData: TWin32FindData;
  begin
    FindHandle := FindFirstFile(PChar(Dir + '*.*'), FindData);
    if FindHandle <> INVALID_HANDLE_VALUE then
     try
       repeat
         if (String(FindData.cFileName) <> '.') and
            (String(FindData.cFileName) <> '..') then
          begin
           if Sender = acThread1 then
            begin
             FileName1 := Dir + FindData.cFileName;
             acThread1.Synchronize(SyncFileFound1);
            end
           else
            begin
             FileName2 := Dir + FindData.cFileName;
             acThread2.Synchronize(SyncFileFound2);
            end;

           if (FindData.dwFileAttributes and FILE_ATTRIBUTE_DIRECTORY =
FILE_ATTRIBUTE_DIRECTORY) then
             ScanDir(Dir + FindData.cFileName + '\');
          end;
       until not FindNextFile(FindHandle, FindData) or (Sender as
TacThread).Terminated
     finally
       Windows.FindClose(FindHandle);
    end;
  end;

begin
  ScanDir('c:\');
end;

// methods synchronized with main application thread
procedure TForm1.SyncFileFound1;
begin
  ListBox1.Items.Add(FileName1);
  Label1.Caption := 'Files: ' + IntToStr(ListBox1.Items.Count);
end;

procedure TForm1.SyncFileFound2;
begin
  ListBox2.Items.Add(FileName2);
  Label2.Caption := 'Files: ' + IntToStr(ListBox2.Items.Count);
end;
```

See also

OnException and OnTerminate events.

© 1998-2006, UtilMind Solutions®

9.42.4.3 OnTerminate

Applies to
acThread component.

Declaration
```
property OnTerminate: TNotifyEvent;
```

Description
The OnTerminate method occurs after the termination of thread, when thread done all processes in the OnExecute event handler

See also
OnException and OnExecute events.

9.42.4.4 OnWaitTimeoutExpired

Applies to
acThread, acHTTP, acSendMail and acSQLQueryThread components.

Declaration
```
var
  TacThreadWaitTimeoutExpired = procedure(Sender: TObject; var
TerminateThread: Boolean) of object;

property OnWaitTimeoutExpired: TacThreadWaitTimeoutExpired;
```

Description
The OnWaitTimeoutExpired event occurs when the component did not finished execution for the time interval specified in WaitTimeout property (when WaitTimeout is expired).

The *TherminateThread* is optional parameter (True by default), which allows to terminate, or prevent termination of thread in case if specified time interval is expired before completion of execution of the thread.

Notes
The WaitTimeout only works together with WaitThread property, only when it set to True.

See also
WaitThread and WaitTimeout properties;
Terminate method.

9.43 acThreadSafeLog

9.43.1 TacThreadSafeLog

Overview
The acThreadSafeLog is the log-writter component, which can be safely accessed from various different threads simultaneously, without additional synchronization.

It can write the logged text both to specified file or "memo" control, and automatically inserts the specified prefix before each line. The prefix of each record can contain the current time, thread identifier and number of CPU ticks passed from application startup.

Also the acThreadSafeLog can automatically send email message with last *N* log entries to administrator in case of fatal error.

© 1998-2006, UtilMind Solutions®

How to use?

In general, all that you need is to specify the file to which the log should be written to the FileName property, then call Log method anywhere its required, without worries about synchronization, lost text, and sudden access violatios even if you call the Log method "simultaneously" from separate threads of your application.

Every time when your program calls Log method, it appends a new line of specified text to the log file, and automatically adds a special prefix, which can contain current time (with format specified in DateTimeFormat property), the identifier of thread which calls the Log method, and number of CPU ticks passed from the application startup. The values which you would like to see before each line of logged text, can be configured in Options property. Alternatively you can write OnCustomPrefix event handler to specify your own prefix before each line of log file dynamically, before the text will be actually written to the log file.

In case if you would like to direct the output of the log file to some "memo" control (TCustomMemo or any its descendant) — point the corresponding control to the Memo property and set LogToMemo property to True.

new! If machine which runs your application have limited disk space and you should limit the maximum size of log file — you can specify this limit in MaxSize property. When the size of the log file outgrows the limit, it calls OnMaxSizeReached event and automatically clears the log.

new! Also the acThreadSafeLog can send email message with last N log entries to administrator in case of fatal error.

Remarks

If the file specified in FileName property is not exists, the component will automatically create it.

Note that the component locks that specified file at run-time, so operating system will prevent to append or modify information in that file by other programs. It just opens the file when program starts, and close it when program terminates. Unfortunately the application can not use the same log file from its several running instances, so use acOnlyOne component to allow to run only one instance of your application.

See also

acThread and acOnlyOne components.

9.43.2 Properties

9.43.2.1 DateTimeFormat

Applies to

acThreadSafeLog component.

Declaration

```
property DateTimeFormat: String; // 'dd/mm/yyyy hh:nn:ss' by default
```

Description

The DateTimeFormat property specifies the format of the date-and-time value which can be automatically stored to the log as prefix of each line, if *putTimeStamp* Option is True.

When the component put current date and time to the log, it uses standard *FormatDateTime* function from SysUtils unit. The following format specifiers are supported:

Specifier	Displays
c	Displays the date using the format given by the ShortDateFormat global variable,

© 1998-2006, UtilMind Solutions®

	followed by the time using the format given by the LongTimeFormat global variable. The time is not displayed if the fractional part of the DateTime value is zero.
d	Displays the day as a number without a leading zero (1-31).
dd	Displays the day as a number with a leading zero (01-31).
ddd	Displays the day as an abbreviation (Sun-Sat) using the strings given by the ShortDayNames global variable.
dddd	Displays the day as a full name (Sunday-Saturday) using the strings given by the LongDayNames global variable.
ddddd	Displays the date using the format given by the ShortDateFormat global variable.
dddddd	Displays the date using the format given by the LongDateFormat global variable.
m	Displays the month as a number without a leading zero (1-12). If the m specifier immediately follows an h or hh specifier, the minute rather than the month is displayed.
mm	Displays the month as a number with a leading zero (01-12). If the mm specifier immediately follows an h or hh specifier, the minute rather than the month is displayed.
mmm	Displays the month as an abbreviation (Jan-Dec) using the strings given by the ShortMonthNames global variable.
mmmm	Displays the month as a full name (January-December) using the strings given by the LongMonthNames global variable.
yy	Displays the year as a two-digit number (00-99).
yyyy	Displays the year as a four-digit number (0000-9999).
h	Displays the hour without a leading zero (0-23).
hh	Displays the hour with a leading zero (00-23).
n	Displays the minute without a leading zero (0-59).
nn	Displays the minute with a leading zero (00-59).
s	Displays the second without a leading zero (0-59).
ss	Displays the second with a leading zero (00-59).
t	Displays the time using the format given by the ShortTimeFormat global variable.
tt	Displays the time using the format given by the LongTimeFormat global variable.
am/pm	Uses the 12-hour clock for the preceding h or hh specifier, and displays 'am' for any hour before noon, and 'pm' for any hour after noon. The am/pm specifier can use lower, upper, or mixed case, and the result is displayed accordingly.
a/p	Uses the 12-hour clock for the preceding h or hh specifier, and displays 'a' for any hour before noon, and 'p' for any hour after noon. The a/p specifier can use lower, upper, or mixed case, and the result is displayed accordingly.
ampm	Uses the 12-hour clock for the preceding h or hh specifier, and displays the contents of the TimeAMString global variable for any hour before noon, and the contents of the TimePMString global variable for any hour after noon.
/	Displays the date separator character given by the DateSeparator global variable.
:	Displays the time separator character given by the TimeSeparator global variable.
'xx'/"xx"	Characters enclosed in single or double quotes are displayed as-is, and do not affect formatting.

Format specifiers may be written in upper case as well as in lower case letters--both produce the same result.

© 1998-2006, UtilMind Solutions®

If the string given by the Format parameter is empty, the date and time value is formatted as if a 'c' format specifier had been given.

This property only take effect if you set *putTimeStamp* option to True in the Options property. Alternatively, if you don't want to use standard date-time convertion functions, you can write your own OnCustomPrefix event handler, to specify the text which will appear before each line of the log file.

See also
Options property and OnCustomPrefix event.

9.43.2.2 FileName

Applies to
acThreadSafeLog component.

Declaration
```
property FileName: String;
```

Description
The FileName property specifies the file to which the log should be written.

The file will be appended with some text every time when the application calls the Log method, if LogToFile property is True.

The component locks the specified file at run-time, so operating system will prevent to append or modify information in that file by other programs. It just opens the file when program starts, and close it when program terminates. However you can change the FileName dynamically at run-time (the component will unlock previous file to use another one).

Also, unfortunately the application can NOT use the same log file from its several running instances (since the log file are locked by first application instance), so use acOnlyOne component to allow to run only one instance of your application.

If you would like to direct the output also to some Memo control, you can specify it to the Memo property and set LogToMemo property to True.

See also
FlushBuffers, Memo, LogToFile, LogToMemo and Options properties;
Log and Flush methods;
acOnlyOne component.

9.43.2.3 FlushBuffers

Applies to
acThreadSafeLog component.

Declaration
```
property FlushBuffers: Boolean;
```

Description
The FlushBuffers property controls whether the component should automatically clear the buffers for the specified log file and cause all buffered data to be written imediately after calling the Log method.

© 1998-2006, UtilMind Solutions®

When you call Log method, it's typically write the data to the internal system buffer that the operating system writes to the disk on a regular basis. When FlushBuffers property is True, the component automatically write all of the buffered information for the file (specified in the FileName property) to disk.

The flush operations takes additional CPU time which can be critical if you running some server application with tons of active threads. It is recommended to leave FlushBuffers property False, and just call Flush method when it's required.

See also
FileName property;
Log and Flush methods.

9.43.2.4 LogToFile

Applies to
acThreadSafeLog component.

Declaration
```
property LogToFile: Boolean;
```

Description
The LogToFile property controls whether the component should append the logged text to specified FileName every time when application calls the Log method.

The component can put logged text to either, the specified file or to Memo control. Set LogToFile to True to store logged text to the file, or set it to False, if you don't want to store the logs to the file. If you just want to direct the output to the Memo control — point it to the Memo property and set LogToMemo to True.

See also
FileName, Memo and LogToMemo properties;
Log method.

9.43.2.5 LogToMemo

Applies to
acThreadSafeLog component.

Declaration
```
property LogToMemo: Boolean;
```

Description
The LogToMemo property controls whether the component should direct the output of the logged text to specified Memo control (TCustomMemo component or any its desendant) every time when application calls the Log method.

The component can put logged text to either, the specified file or to Memo control. Set LogToMemo to True to store logged text to memo, or set it to False, if you want to store the logs only to file.

See also
FileName, Memo and LogToFile properties;
Log method.

© 1998-2006, UtilMind Solutions®

9.43.2.6 MaxSize

Applies to
acThreadSafeLog component.

Declaration
```
property MaxSize: Cardinal; // in KB!
```

Description
The MaxSize property used to set size limit, in kilobytes unit, of the log file.

When size of the log file reaches specified value, the component triggers OnMaxSizeReached event and automatically clears the log both in file and the memo control.

If MaxSize is 0, the log can have any size (unlimited).

See also
ClearLog method;
OnMaxSizeReached event.

9.43.2.7 Memo

Applies to
acThreadSafeLog component.

Declaration
```
property Memo: TCustomMemo;
```

Description
The Memo property can point to some TCustomMemo control or its descendant, if you would like to direct the output to screen together (or instead of) logging text to the file.

See also
LogToMemo, LogToFile and FileName properties;
Log method.

9.43.2.8 Options

Applies to
acThreadSafeLog component.

Declaration
```
type
  TacThreadSafeLogOption = (putTickStamp, putTimeStamp, putThreadID);
  TacThreadSafeLogOptions = set of TacThreadSafeLogOption;

property Options: TacThreadSafeLogOptions;
```

Description
The Options property specifies which dynamic values should be stored to the log file as prefix before each line of logged text.

There is 3 possible options/values which can be automatically put to the prefix before each line:

Option	Description
putTickStamp	puts the number of CPU ticks passed from the application startup;

© 1998-2006, UtilMind Solutions®

putTimeStamp	puts current date and time in format specified in the DateTimeFormat property;
putThreadID	puts the identifier of the thread which called Log method.

Alternatively, if you don't want to put standard values descibed in Options, you can use OnCustomPrefix event to specify your own prefix before each line of logged text.

See also
DateTimeFormat property and OnCustomPrefix event.

9.43.2.9 SendMail

Applies to
acThreadSafeLog component.

Declaration

```
type
  TacSMTPPriority = (mpHighest, mpHigh, mpNormal, mpLow, mpLowest);

  { TacThreadSafeLogSendMailProperties }
  TacThreadSafeLogSendMailProperties = class
  public
    property Limit: Cardinal default 0;
    property MsgContentCharset: String default 'iso-8859-1';
    property MsgFromAddress: String;
    property MsgFromName: String;
    property MsgToAddress: String;
    property MsgToName: String;
    property MsgSubject: String;
    property MsgPriority: TacSMTPPriority default mpNormal;
    property SMTPHost: String default 'localhost';
    property SMTPPort: Word default 25;
    property Threaded: Boolean default False;
  end;

property SandMail: TacThreadSafeLogSendMailProperties;
```

Description
The SendMail structure used to specify the SMTP properties and headers of email message, which could be sent using EmailLastRecords method (this method can be used to send an email message with last N log entries, to inform the administrator about some fatal error).

The SMTPHost and SMTPPort properties specifies the host and port of SMTP server which relays the email messages.

Limit property specifies the maximum number of messages which could be sent per application session (0 is unlimited).

Threaded specifies whether email should be sent in separate or in current application thread (True — use separate thread).

Other properties (with Msg… prefix), used to specify the headers of email message.

See also
EmailLastRecords method;
acSendMail component.

© 1998-2006, UtilMind Solutions®

9.43.2.9.1 Limit

Applies to
acThreadSafeLog component as subproperty of SendMail structure.

Declaration
```
property Limit: Cardinal;
```

Description
The Limit property specifies the maximum number of email nessages which possible to send using EmailLastRecords method.

For example, in case if Limit = 100, then the application can send only 100 email messages with log entries per its session. When the Limit is exceed, no messages will be sent.

To disable the limit — set Limit to 0 (unlimited number of emails with last log entries).

See also
EmailLastRecords method.

9.43.2.9.2 MsgContentCharset

Applies to
acThreadSafeLog component as subproperty of SendMail structure.

Declaration
```
property MsgContentCharset: String; // 'iso-8859-1' by default
```

Description
The MsgContentCharset property determines the charaters set used in the text of e-mail message. This parameter always included to the message to allow the e-mail client to determinate which charset should be used to display the message.

Note
You should programmatically handle conversion of text to required code table if the content charset of e-mail is different than charset used on typing a message. Otherwise the text mey appears to be broken. (FYI: 'iso-8859-1' is the standard Western European code table, AFAIK, 'iso-8859-1' is the same as 'us-ascii'.)

The SendMail automatically converts the text from Windows to KOI8 table ('koi8-r' is the standard Russian charset used in e-mail messages), if the When MsgContentCharset = 'koi8-r'. (So Russian programmers may do not worry about this.)

See also
EmailLastRecords method.

9.43.2.9.3 MsgFromAddress

Applies to
acThreadSafeLog component as subproperty of SendMail structure.

Declaration
```
property MsgFromAddress: String;
```

Description
The MsgFormAddress property specifies the name of sender (i.e: 'john@doe.com'). Leave it blank if

© 1998-2006, UtilMind Solutions®

you don't want to include the e-mail address to the header of message.

Note

Some SMTP servers don't relay anonymous messages and requires e-mail address of sender to be specified.

See also

MsgFromName, MsgToAddress and MsgToName properties;
EmailLastRecords method.

9.43.2.9.4 MsgFromName

Applies to

acThreadSafeLog component as subproperty of SendMail structure.

Declaration

```
property MsgFromName: String;
```

Description

The MsgFormName property specifies the name of sender (i.e: 'My Server'). Leave it blank if you don't want to include the sender's name to the header of e-mail message.

See also

MsgFromName, MsgToAddress and MsgToName properties;
EmailLastRecords method.

9.43.2.9.5 MsgPriority

Applies to

acThreadSafeLog component as subproperty of SendMail structure.

Declaration

```
type
  TacSMTPPriority = (mpHighest, mpHigh, mpNormal, mpLow, mpLowest);

property MsgPriority: TacSMTPPriority; // mpNormal by default
```

Description

The MsgPriority property specifies an importance and priority of e-mail message. The priority of message will be shown by e-mail reader program. Also, most SMTP servers relays messages with the higher priorities earlier than messages with low priorities.

The value of MsgPriority property automatically includes to the header of message as "X-Priority" field.

See also

EmailLastRecords method.

9.43.2.9.6 MsgSubject

Applies to

acThreadSafeLog component as subproperty of SendMail structure.

Declaration

```
property MsgSubject: String;
```

© 1998-2006, UtilMind Solutions®

Description
The MsgSubject property specifies the subject of e-mail message.

See also
EmailLastRecords method.

9.43.2.9.7 MsgToAddress

Applies to
acThreadSafeLog component as subproperty of SendMail structure.

Declaration
```
property MsgToAddress: String;
```

Description
The MsgToAddress property specifies the recipient of the message (email address).

See also
MsgFromAddress, MsgFromName and MsgToName properties;
EmailLastRecords method.

9.43.2.9.8 MsgToName

Applies to
acThreadSafeLog component as subproperty of SendMail structure.

Declaration
```
property MsgToName: String;
```

Description
The MsgToName property specifies the name of recipient of the email message (i.e: 'John Doe').

See also
MsgFromAddress, MsgFromName and MsgToAddress properties;
EmailLastRecords method.

9.43.2.9.9 SMTPHost

Applies to
acThreadSafeLog component as subproperty of SendMail structure and acSendMail component.

Declaration
```
property SMTPHost: String; // 'localhost' by default
```

Description
The SMTPHost property specifies the hostname or IP address of the SMTP server which should be used to relay the e-mail messages to recipients.

See also
EmailLastRecords method.

9.43.2.9.10 SMTPPort

Applies to
acThreadSafeLog component as subproperty of SendMail structure and acSendMail component.

Declaration

© 1998-2006, UtilMind Solutions®

```
property SMTPPort: Word; // 25 by default (standard STMP port number)
```

Description
The SMTPPort specifies the port number of SMTP server (specifies the port which listen the server). The standard port number for SMTP protocol is 25.

See also
EmailLastRecords method.

9.43.2.9.11 Threaded

Applies to
acThreadSafeLog component as subproperty of SendMail structure, acSendMail and acHTTP components.

Declaration
```
property Threaded: Boolean; // False by default
```

Description
The Threaded property used to choose between two working models of SendMail feature of acThreadSafeLog component.

Basically there is two possible working models:

1. **Threaded**: when Threaded is True, the component creates new separate thread to produce and send email message to specified SMTP host. In this case the SendMail does not locks the application interface when it connects to the SMTP server and sends the message via Internet, and only produce events synchronized with main application thread (interface).

   ⚠ However, in this model the component can send ONLY ONE message simultaneously, since it can create only one thread to send message. If you will try to send additional message when the component is busy, next email will not be sent, and EmailLastRecords method will just return False. You should implement some message queue to operate with SendMail in this threading model and don't call EmailLastRecords method while previous message not yet sent.
   Otherwise — set Threaded to False and use non-threaded model to send any number of email messages simultaneously from any number of different threads.

2. **Non-threaded**: when Threaded is False, the component does not creates any separate thread to produce and send email message. This model should be used only if your application uses many threads which could send email message using this component. This model should NOT be used in single-threaded application, since when you call the EmailLastRecords method, all your application interface will be locked while the component producing and sending the email message.

See also
EmailLastRecords method.

9.43.3 Methods

9.43.3.1 ClearLog

Applies to
acThreadSafeLog component.

Declaration
```
procedure ClearLog;
```

© 1998-2006, UtilMind Solutions®

Description
The ClearLog method recreates the log file and clears all previous logged records both in file and memo control.

See also
MaxSize property and OnMaxSizeReached event.

9.43.3.2 Flush

Applies to
acThreadSafeLog component.

Declaration
```
procedure Flush;
```

Description
The Flush method clears the buffers for the specified log file and cause all buffered data to be imediately written.

When you call Log method, it's typically write the data to the internal system buffer that the operating system writes to the disk on a regular basis. When FlushBuffers property is True, the component automatically write all of the buffered information for the file (specified in the FileName property) to disk.

The flush operations takes additional CPU time which can be critical if you running some server application with tons of active threads. It is recommended to leave FlushBuffers property False, and just call Flush method when it's required.

See also
FileName and FlushBuffers properties;
Log method.

9.43.3.3 Log

Applies to
acThreadSafeLog component.

Declaration
```
procedure Log(const Text: String); overload;
procedure Log(Int: Int64); overload;
procedure Log(Ext: Extended); overload;
```

Description
The Log method appends new line of *Text* to the log file specified in the FileName property, plus can add the same line to some Memo control if it's specified.

You can call this method from any place and any thread of your application without worrying about synchronization between threads. Everything will be logged as charm, none byte will be lost, none memory access violation will be occur.

To call Log method, you need to specify some *Text* (or integer value which will be automatically converted to text). The component will append this text to the file specified in the FileName property (if LogToFile is True), or to some Memo control (if the controls specified and LogToMemo is True).

When the component is about to append specified text to the log, it also puts special prefix before the text. The prefix may contain current date and time, the identifier of calling thread, or

© 1998-2006, UtilMind Solutions®

number of CPU ticks passed from application startup (see Options property for more details). If you don't want to automatically add the prefix before logged text, you may set all of Options to False, or specify your own custom prefix in the OnCustomPrefix event handler.

See also
FileName, Memo, LogToFile, LogToMemo and Options property;
OnCustomPrefix event.

9.43.3.4 GetLastRecords

Applies to
acThreadSafeLog component.

Declaration
```
function GetLastRecords(Count: Cardinal = 100): String;
```

Description
The GetLastRecords method retreives last *N* log entries (*N* specified by Count parameter, 100 by default), and returns them as string.

See also
EmailLastRecords method.

9.43.3.5 EmailLastRecords

Applies to
acThreadSafeLog component.

Declaration
```
function EmailLastRecords(Count: Cardinal = 100): Boolean;
```

Description
The EmailLastRecords method retreives last N logged entries and sends them as email message, to address specified in SendMail structure, via specified SMTP host and port.

See also
SendMail structure;
GetLastRecords method.

9.43.4 Events

9.43.4.1 OnCustomPrefix

Applies to
acThreadSafeLog component.

Declaration
```
type
  TacThreadSafeLogCustomPrefixEvent = procedure(Sender: TObject; var
Prefix: String) of object;

property OnCustomPrefix: TacThreadSafeLogCustomPrefixEvent;
```

Description
The OnCustomPrefix event occurs when the component is about to append new line to the log file. It allows to specify custom prefix before each line of logged text. For example, you can specify current time, or identifier or calling thread or something else, which can not be specified in Options

© 1998-2006, UtilMind Solutions®

property and can not be put as prefix automatically.

Example

```
procedure TForm1.acThreadSafeLog1CustomPrefix(Sender: TObject;
  var Prefix: string);
var
  TickDiff: DWord;
begin
  // basically to specify the prefix, just put required text to Prefix
parameter
  Prefix := 'My custom prefix';

  // here is more complicated example
  TickDiff := GetTickCount - FStartTime;
  Prefix := Format('[%d.%3.3d %s %d] %s', [TickDiff div 1000, TickDiff
mod 1000,
                    FormatDateTime(FDateTimeFormat, Now),
GetCurrentThreadID, Text]);
end;
```

See also

Options and DateTimeFormat properties.

9.43.4.2 OnFileWriteError

Applies to

acThreadSafeLog component.

Declaration

```
property OnFileWriteError: TNotifyEvent;
```

Description

The OnFileWriteError occurs if component can't append new line to the log. One of possible reasons — disk is full.

See also

ClearLog method.

9.43.4.3 OnMaxSizeReached

Applies to

acThreadSafeLog component.

Declaration

```
type
  TacThreadSafeLogMaxSizeReachedEvent = procedure(Sender: TObject; var
ClearLog: Boolean) of object;

property OnMaxSizeReached: TacThreadSafeLogMaxSizeReachedEvent;
```

Description

The OnMaxSizeReached event occurs when the size of log file outgrows to limit specified in MaxSize property. After log reaches specified value, the component automatically clears the log, if your application not override *ClearLog* parameter in this event handler.

Write the OnMaxSizeReached event handler to take some specific actions when the log file outgrows the limit, or to prevent automatic clearing of the log file by specifying False to *ClearLog*

© 1998-2006, UtilMind Solutions®

parameter. Alternatively you can simply call ClearLog method to empty the log manually.

See also
ClearLog method.

9.44 acTrayIcon

9.44.1 TacTrayIcon

Overview
The acTrayIcon is handy component for manipulating the system tray icons in the taskbar status area. Allow to use the animated icons and contains the number of great features such like additional caption button on form's title bar and menu item in the system menu. acTrayIcon is fully automated and can be easily customized with number of various properties. It contains many others properties which allow you to associate your forms with tray icons on taskbar status area without ANY line of additional code!

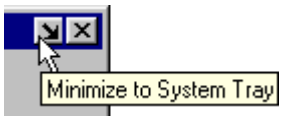
A non intrusive way to display an explanatory message for a status bar icon is to implement a balloon tooltip with its stem pointing to the icon. The acTrayIcon supports the cartoon-style balloon tooltips which appears after minimizing of the form to the system tray. Check out Win2kInfoBalloon structure for more details. The Win2kInfoBalloon can be specified in any Windows version but works (can be appears) in Windows 2000 and later.

new! The acTrayIcon automatically detects when Windows Explorer was crashed or restarted and automatically restores the icon in the tray.

Key properties

Animate	animate the tray icon, if multiple icons specified in ImageList.
Icon	specifies the Icon to display in system tray.
ImageList	specifies the list of icon images, if you would like to make tray icon Animated.
Options	number of various options that set behavior of tray icon
PopupMenu	assigns any popup menus to the tray icon. (acTrayIcon can automatically handle clicks on "Minimize-To-Tray", "Restore-From-Tray" and "What's This?" menu items — see PopupMenuRestoreItem, PopupMenuTrayMinimizeItem and PopupMenuWhatsThisItem properties)
Hint	the text string which appears when mouse moves over the tray icon.
Visible	shows or hides the tray icon.
Win2kInfoBalloon	the cartoon-style balloon window with some important information once the tray icon minimized to the system tray. This feature works on Windows 2000 and later.

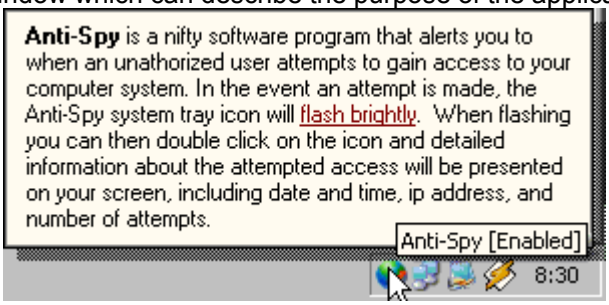
Cool features
acTrayIcon is successor of acCustomCaptionButton class. It have a built-in support of the corresponding caption button on the form's title bar and the menu item in the system menu, associated with this button.



Once the tray icon minimized to the system tray, it can show the balloon-style information window with some important information. Check out Win2kInfoBalloon property.

© 1998-2006, UtilMind Solutions®

new! The acTrayIcon have built-in "What's This?" popup window. Just specify the WhatsThisHint property and point the menu item (in the PopupMenu) which will execute this window. When user clicks the menu item specified in PopupMenuWhatsThisItem, the tray icon will show the popup window which can describe the purpose of the application and its tray icon.



The acTrayIcon automatically minimizes the application to the system tray on program startup, if the application was executed with "/minimized" command-line parameter.

How to change the glyphs for "Minimize to tray" button ?

To change the glyph for the "Minimize to tray" button, you have to modify the ACCB_MINIMIZETOTRAY bitmap resources in the "CaptionButtons.res" file of the AppControls pack. You can change them using standard Image Editor program which supplies with every Delphi/BCB version.

After modifying of these glyphs in the "CaptionButtons.res", they will always with you in any programs, so you will do not need to specify the glyphs every time when you drop component onto form.

See also:

Other components which contains a built-in caption button: acAppAutoRun, acFormHelp, acFormTopmost, acFormRoller.

9.44.2 Properties

9.44.2.1 Animate

Applies to

acTrayIcon component.

Declaration

```
property Animate: Boolean;
```

Description

The Animate property controls whether you would like to animate images, assigned to ImageList property. If Animate is True, acTrayIcon will increase ImageIndex value, with interval specified in AnimateInterval property. If ImageList is not assigned and you have specified just one icon (in the Icon property), setting True to Animate will take not effect.

Icons can be animated only if

© 1998-2006, UtilMind Solutions®

1. AnimateInterval value is higher than 0 (milliseconds), *and…*
2. Animate property is True, *and…*
3. Multiple images assigned to ImageList.

See also
AnimateInterval, ImageList, ImageIndex properties and OnAnimate event.

9.44.2.2 AnimateInterval

Applies to
acAppEvents and acTrayIcon components.

Declaration
```
property AnimateInterval: Word; // in milliseconds
```

Description
The AnimateInterval property determines in milliseconds the tim interval that passes before the acTrayIcon component increases the ImageIndex value to animate images assigned to ImageList.

Icons can NOT be animated when
1. AnimateInterval value is 0;
2. Animate property is False;
3. No images assigned to ImageList.

See also
Animate, ImageList, ImageIndex properties.

9.44.2.3 CaptionButton

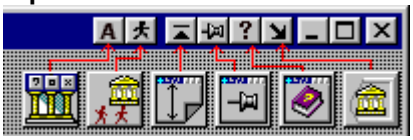
Applies to
acTrayIcon, acAppAutoRun, acFormHelp, acFormTopmost and acFormRoller components.

Declaration
```
property CaptionButton: TacACaptionButton;
```

Description
The CaptionButton is the set of properties for managing an additonal buttons on the title bar:

BtnOrder	controls the button placement order on the title bar;
Cursor	specifies the cursor image when button released;
CursorDown	specifies the cursor image when button pressed;
Enabled	enables or disables button;
Hint	specifies the button's tooltip;
SeparatorWidth	width between current and previous buttons;
ShowHint	enables or disables the tooltip (hint);
Visible	shows or hides the button from the title bar.

Snapshot



9.44.2.3.1 BtnOrder

Applies to
acTrayIcon, acAppAutoRun, acFormHelp, acFormTopmost, acFormRoller components as

© 1998-2006, UtilMind Solutions®

subproperty of CaptionButton.

Declaration

```
property BtnOrder: TacBtnOrder;
```

Description

The BtnOrder property indicates position of caption button, the order in which buttons will be displayed. BtnOrder calculates from right side of title bar to left side (see snapshot below).

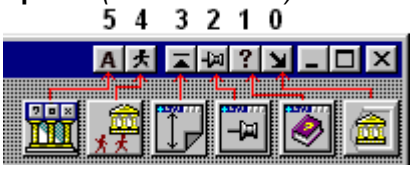
Initially, the button order is always the order in which the components were added to the form, but you can change this by changing the BtnOrder property. The value of the BtnOrder property is unique for each ancestor of the acCaptionButton on the form (like TabOrder property of ancestors of TWinControl). The first caption button added to the form has a BtnOrder value of 0, the second is 1, the third is 2, and so on. These values determine where a control is on the title bar.

If you attempt to give a button a BtnOrder value greater than the number of caption buttons on the form minus one (because numbering starts with 0), AppControls won't accept the new value, but will enter the value that assures the component will be the last in the button order.

The caption button with the BtnOrder value of 0 is the custom button which placed on right side of title bar, after regular system buttons.

To separate the buttons by additional empty space — use SeparatorWidth property. For example snapshot below shows that acTrayIcon component have BtnOrder = 0 (on first position from the left of system button) and SeparatorWidth = 2 pixels.

Snapshot *(buttons orders):*



See also

SeparatorWidth property and acCaptionButton component

9.44.2.3.2 Cursor

Applies to

acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller components as subproperty of CaptionButton.

Declaration

```
property Cursor: TCursor;
```

Description

The Cursor property controls the mouse cursor shape used when the mouse moves over CaptionButton in released state.

See also

CursorDown property and acCaptionButton component.

9.44.2.3.3 CursorDown

Applies to

acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller components as

© 1998-2006, UtilMind Solutions®

subproperty of CaptionButton.

Declaration

```
property CursorDown: TCursor;
```

Description

The CursorDown property controls the mouse cursor shape used when the mouse moves over CaptionButton in pressed state.

See also

Cursor property and acCaptionButton component.

9.44.2.3.4 Enabled

Applies to

acTrayIcon, acAppAutoRun, acFormHelp, acFormTopmost, acFormRoller components as subproperty of CaptionButton.

Declaration

```
property Enabled: Boolean;
```

Description

The Enabled property controls whether the button on the title bar responds to mouse and keyboard messages. If Enabled is True, button responds normally. If Enabled is False, button become disabled and user can not press that button.

See also

Visible property of CaptionButton and acCaptionButton component.

9.44.2.3.5 Hint

Applies to

acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller components as subproperty of CaptionButton.

Declaration

```
property Hint: String;
```

Description

The Hint property is the text string that can appear when mouse pointer moves over the caption button.

If the CaptionButton's ShowHint property is False, the Help Hint won't appear, but the other hints still will.

Default Hint string for acTrayIcon component is 'Minimize to System Tray'. You may change this string at run- or design-time if you're writting multilingual program.

See also

acCaptionButton component and ShowHint property.

9.44.2.3.6 SeparatorWidth

Applies to

acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller components as subproperty of CaptionButton.

© 1998-2006, UtilMind Solutions®

Declaration

```
property SeparatorWidth: Byte; // in screen pixels
```

Description

The SeparatorWidth property controls the space between current and previous by order caption button in screen pixels.

See also

BtnOrder property and acCaptionButton component.

9.44.2.3.7 ShowHint

Applies to

acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller components as subproperty of CaptionButton.

Declaration

```
property ShowHint: Boolean;
```

Description

The ShowHint property controls whether CaptionButton can show the Hint string when mouse pointer moves over button. If ShowHint is True, Hint will appears.

See also

acCaptionButton component and Hint property.

9.44.2.3.8 Visible

Applies to

acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller components as subproperty of CaptionButton.

Declaration

```
property Visible: Boolean;
```

Description

The Visible property determines whether the caption button currently visible on the title bar. Make Visible property True if you would like to show the button on title bar, or False if you would like to hide the button.

When you changing the Visible property, placements of others button will automacially recalculated and other buttons with higher BtnOrder will be shifted to the right side of the title bar, filling the empty space.

See also

Enabled property of CaptionButton and acCaptionButton component.

9.44.2.4 Hint

Applies to

acTrayIcon and acCaptionButton component.

Declaration

```
property Hint: String;
```

Description

© 1998-2006, UtilMind Solutions®

The Hint property is the text string which appears when the mouse pointer hovers over the system tray icon. If the ShowHint property is False, the hint will not appears.

Snapshot

See also

ShowHint, Visible, WhatsThisHint and Win2kInfoBalloon properties.

9.44.2.5 Icon

Applies to

acTrayIcon and acAppEvents components.

Declaration

```
property Icon: TIcon;
```

Description

The Icon property determines the icon that is displayed in system tray area when acTrayIcon component is visible.

If list of images already assigned to ImageList property, then images assigned to ImageList will be used for displaying the tray icon.

If no Icon and ImageList assigned, then for displaying the tray icon acTrayIcon component will use the application icon (Application.Icon property).

Note

This property CAN contain high-color icon. However, you must know that icons in the system tray with 256 and more colors can be visible only in Windows ME / XP and higher. All lower versions (Win95/98/NT4/2000) can display 16 colors only!

So if you decided to use high-color icon for WinME/XP, but still want to make it looks fine in lower Windows versions, you can specify 256 colors icon to Icon property and 16 colors icon to Icon16 property. Otherwise, if you have only 16-colors icon or just don't care about look in older Windows version, then specify just Icon property only, or use image from ImageList.

Alternatively you can use just icons from ImageList, determinate the Windows version, and show 16-color icons on Win95/98/NT4/2000 and 256-color icons on WinME/XP and higher. To determinate whether the 256 colors icon are allowed in the system tray, you can use Is256ColorTrayIconAllowed function from acUtils unit.

See also

Icon16, ImageList and ImageIndex properties.

9.44.2.6 Icon16

Applies to

acTrayIcon components.

Declaration

```
property Icon16: TIcon;
```

Description

© 1998-2006, UtilMind Solutions®

The Icon16 property determines the 16-colors icon that is displayed in system tray area, in case if the component has detected that Windows version is lower than WinME/XP and can not normally display high-color image from Icon property.

Note

This property should be specified only if you are using 256-color image in the Icon property (which can normally displayed only in WinXP/ME and higher versions), but still want to make your tray icon looks fine in lower Windows versions (Win95/98/NT4/2000). Otherwise, if you have only 16-colors icon or just don't care about look in older Windows version, then specify just Icon property only, or use images from ImageList.

Alternatively you can use just icons from ImageList, determinate the Windows version, and show 16-color icons on Win95/98/NT4/2000 and 256-color icons on WinME/XP and higher. To determinate whether the 256 colors icon are allowed in the system tray, you can use Is256ColorTrayIconAllowed function from acUtils unit.

See also

Icon, ImageList and ImageIndex properties.

9.44.2.7 ImageIndex

Applies to

acTrayIcon and acAppEvents components.

Declaration

```
property ImageIndex: Integer;
```

Description

The ImageIndex property specifies the index of image from ImageList property, to display specified image as tray icon. The ImageIndex property used only if TImageList component assigned to ImageList property.

If you're using the animated icons (see Animate property), the ImageIndex will increased every time after the expiration of interval specified in AnimateInterval property.

See also

Animate and ImageList properties.

9.44.2.8 ImageList

Applies to

acTrayIcon and acAppEvents components.

Declaration

```
property ImageList: TImageList;
```

Description

The ImageList property provides the list of icon images to display in the system tray (taskbar status area). To show specific icon from this image list you can use ImageIndex property.

If no images assigned to ImageList property, then image assigned to the Icon property will be used for displaying the tray icon. If Icon is not assigned too, acTrayIcon component will use the application icon (Application.Icon property).

See also

ImageIndex, Icon, Animate, AnimateInterval and Visible properties.

© 1998-2006, UtilMind Solutions®

9.44.2.9 IsMinimized

Applies to
acTrayIcon component.

Declaration
```
property IsMinimized: Boolean;
```

Description
The IsMinimized is the read-only property which allows to determinate whether the application is minimized to system tray.

See also
MinimizeToTray and RestoreFromTray methods;
OnMinimize and OnRestore events.

9.44.2.10 Options

Applies to
acTrayIcon component.

Declaration
```
type
  TacTrayIconOptions = set of (oHideIconOnRestore,
                               oMinimizeToTray,
                               oMinimizeToTrayOnDeactivate,
                               oOneClickRestore,
                               oDblClickRestore,
                               oPlaySysActionSounds,
                               oShowAtDesignTime,
                               oShowMenuOnLeftClick);

property Options: TacTrayIconOptions;
```

Description
The Options property controls the appearance and behavior of the tray icon. Use Options property to customize the appearance and functionality of the TrayIcon component. The possible values of Options are:

Value	Meaning
oHideIconOnRestore	Hides the icon from tray (makes Visible property False) when the form becomes restores from the tray;
oMinimizeToTray	Automatically minimize the window to tray when user clicks button on the form's title bar, or selects according menu item in the system menu;
oMinimizeToTray	Automatically minimize the window to tray when application deactivates (the input focus switches to another application). Alternatively you can write the OnDeactivate event handler ot TApplication object and Minimize the application to tray programmatically;
oOneClickRestore	Restores the window from tray icon on just one mouse click when True;
oDblClickRestore	Prevents the form from minimizing and restoring from the system tray when False;
oPlaySysActionSounds	Controls whether the component should play the default system

© 1998-2006, UtilMind Solutions®

	sounds (the WAV files defined by user in "Sounds" applet of Control Panel), associated with Minimize and Restore events (when the application minimizes or restores from system tray);
oShowAtDesignTime	Shows the tray icon at design time. Make it False if you would not see the design tray icon even if Visible property is True;
oShowMenuOnLeftClick	Shows the popup menu when user click left mouse button over tray icon.

See also
PopupMenu property;
OnDblClick event.

9.44.2.11 PopupMenu

Applies to
acTrayIcon and acCaptionButton components.

Declaration
```
property PopupMenu: TPopupMenu;
```

Description
The PopupMenu property identifies the pop-up menu associated with the tray icon.

Assign a value to PopupMenu to make a pop-up menu appear when the user selects the tray icon and clicks the right mouse button (or on left mouse button click, if `oShowMenuOnLeftClick` specified in Options property).

If the TPopupMenu's AutoPopup property is True, the pop-up menu appears automatically on right mouse click. If the menu's AutoPopup property is False, display the menu with a call to its Popup method from the control's OnClick event handler.

If the popup menu contains the default menu item, the action associated with item (it's OnClick event handler) will be executed when user double clicks the tray icon.

See also
PopupMenuRestoreItem and PopupMenuTrayMinimizeItem properties.

9.44.2.12 PopupMenuRestoreItem

Applies to
acTrayIcon component.

Declaration
```
property PopupMenuRestoreItem: TMenuItem;
```

Description
The PopupMenuRestoreItem points to the menu item in the PopupMenu. When user selects this menu item, application window will restored from system tray.

To associate a menu item with "Minimize To Tray" action - use PopupMenuTrayMinimizeItem property.

See also
PopupMenuTrayMinimizeItem, PopupMenu properties;
RestoreFromTray method.

© 1998-2006, UtilMind Solutions®

9.44.2.13PopupMenuTrayMinimizeItem

Applies to
acTrayIcon component.

Declaration
```
property PopupMenuTrayMinimizeItem: TMenuItem;
```

Description
The PopupMenuTrayMinimizeItem points to the menu item in the PopupMenu. When user select this menu item, application window will minimized to the system tray.

To associate a menu item with "Restore From Tray" action - use PopupMenuRestoreItem property.

See also
PopupMenuRestoreItem, PopupMenu properties;
MinimizeToTray method.

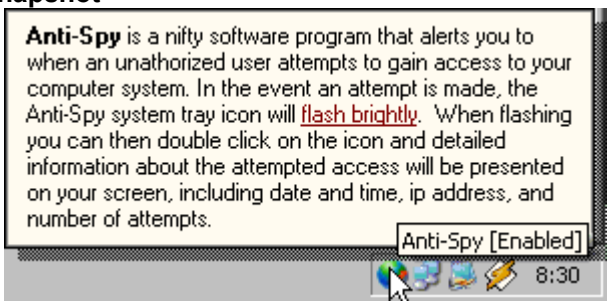
9.44.2.14PopupMenuWhatsThisItem

Applies to
acTrayIcon component.

Declaration
```
property PopupMenuWhatsThisItem: TMenuItem;
```

Description
The PopupMenuWhatsThisItem property used to point the menu item in the PopupMenu, to display the description of purpose of the application and its icon in the system tray.

The PopupMenuWhatsThisItem should point only to the item within PopupMenu, associated with tray icon. When user clicks this menu item, it will automatically show the popup window with the text specified in WhatsThisHint property. The popup window looks and feels like standard WinHelp popup window, and created using function of acFormHelp component.

Snapshot


Note
This feature are not available in stand-alone TrayIcon component and only comes with entire AppControls pack, since it uses function of acFormHelp component to display the popup window.

See also
PopupMenu and WhatsThisHint properties;
acFormHelp component.

© 1998-2006, UtilMind Solutions®

9.44.2.15ShowHint

Applies to
acTrayIcon and acCaptionButton component.

Declaration
```
property ShowHint: Boolean;
```

Description
The ShowHint property determines whether the TrayIcon displays a Help Hint when the mouse pointer rests momentarily on the tray icon.

See also
Hint property.

9.44.2.16StartMinimized

Applies to
acTrayIcon component.

Declaration
```
property StartMinimized: Boolean;
```

Description
The StartMinimized property controls whether an application should be minimized on startup. Set StartMinimized to True if you would like to minimize your program on startup, or leave it False otherwise.

Alternatively you can minimize application in OnShow event handler of your main form using following code (and acAppEvents component):
```
procedure TMainForm.FormShow(Sender: TObject);
begin
  acAppEvents1.ShowMainForm := False;
  acAppEvents1.ShowTaskIcon := False;
  acTrayIcon1.MinimizeToTray;
end;
```

Also you can automatically minimize your application at startup by specifying "/minimized" parameter at startup command-line option. The acTrayIcon component automatically recognize it and set StartMinimized property to True, if "/minimized" option are specified.

See also
MinimizeToTray method and acAppEvents component.

9.44.2.17SystemMenu

Applies to
acCaptionButton component and other components that uses TacCustomCaptionButton as ancestor class - acAppAutoRun, acTrayIcon, acFormHelp, acFormTopmost, acFormRoller.

Declaration
```
property SystemMenu: TacSystemMenu;
```

Description
The System Menu is the popup menu that appears when you click on the program icon on the upper-left corner of form's title bar. The SystemMenu property is the list of properties that manages

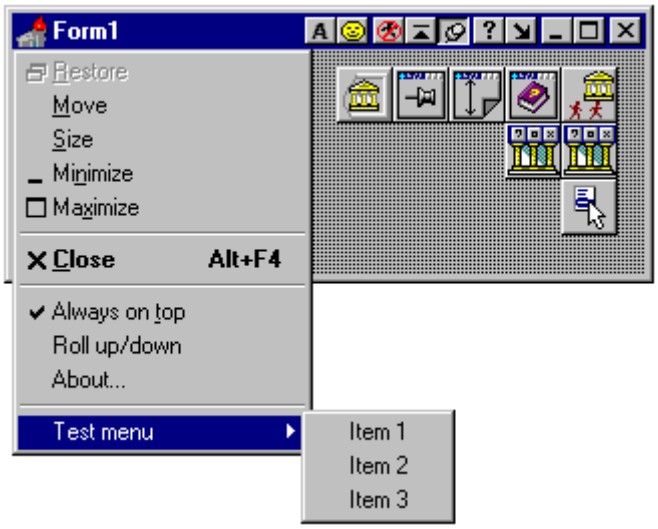
© 1998-2006, UtilMind Solutions®

the menu item associated with current button on the form's title bar.

Properties

ApplyToMenu	applies or removes the menu item associated with current button from the system menu;
Caption	text for the title of menu item;
Position	position of current menu item in the system menu;
Separators	specifies the separators for menu item, to separatate it from previous and / or next menu item of the system menu.

Snapshot



See also

acFormSystemMenu component.

9.44.2.17.1 ApplyToMenu

Applies to

acCaptionButton component and all other components which uses CaptionButton as base ancestor class - acAppAutoRun, acTrayIcon, acFormTopmost, acFormRoller. ApplyToMenu is subproperty of SystemMenu.

Declaration

```
property ApplyToMenu: Boolean;
```

Description

The ApplyToMenu property controls whether the CaptionButton currently have the menu item assiciated with this button. Set ApplyToMenu property to True to add according menu item (with text specified by Caption property, in position specified by Position property) to the form's system menu and False to remove.

See also

Caption property.

9.44.2.17.2 Caption

Applies to

acCaptionButton component and all other components which uses CaptionButton as base ancestor

© 1998-2006, UtilMind Solutions®

class - acAppAutoRun, acTrayIcon, acFormTopmost, acFormRoller. Caption is subproperty of SystemMenu.

Declaration

```
property Caption: String;
```

Description

The Caption property specifies the text for menu item in the form's system menu, associated with current caption button on the title bar. If Caption is not specified, text from Hint property will be taken as item title.

See also

Hint property of CaptionButton

9.44.2.17.3 Position

Applies to

acCaptionButton component and all other components which uses CaptionButton as base ancestor class - acAppAutoRun, acTrayIcon, acFormTopmost, acFormRoller. Position is subproperty of SystemMenu.

Declaration

```
property Position: Word;
```

Description

The Position property determines the position for menu item in the system menu associated with current caption button on the form's title bar. Change the Position value to move current menu item by system menu.

Maximum value for Position property is the current maximum number of items in the system menu (since orders for menu items starts with 0).

See also

Separators property.

9.44.2.17.4 Separators

Applies to

acCaptionButton component and all other components which uses CaptionButton as base ancestor class - acAppAutoRun, acTrayIcon, acFormTopmost, acFormRoller. Separators is subproperty of SystemMenu.

Declaration

```
type
  TacMenuSeparators = set of (seBefore, seAfter);

property Separators: TacMenuSeparators;
```

Description

The Separators property specifies the separators for current menu item to separate it from previous and / or next menu items of the system menu. To set separator before menu item — set `seBefore` to True. To set separator after menu item — set `seAfter` to True.

Specify Separators to separate current menu item in the system menu from another menu items.

See also

© 1998-2006, UtilMind Solutions®

Position property.

9.44.2.18Visible

Applies to
acTrayIcon and acCaptionButton components.

Declaration
```
property Visible: Boolean;
```

Description
The Visible property determines whether the icon associated with acTrayIcon component is currently visible in the system tray area. Set Visible property to True if you would like to show the tray icon, or False if you would like to hide the icon.

If you would like to make the tray icon always invisible at design-time, even if Visible property is True - disable `oShowAtDesignTime` option.

See also
Options property.

9.44.2.19WhatsThisHint

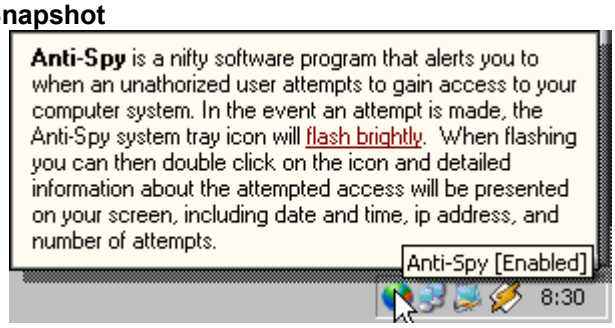
Applies to
acTrayIcon component.

Declaration
```
property WhatsThisHint: String;
```

Description
The WhatsThisHint property specifies the text for popup window which appears when user clicks the menu item associated with PopupMenuWhatsThisItem.

The popup window looks and feels like standard WinHelp popup window, and since it created using function of acFormHelp component it can contain the rich-formatting tags of FormHelp (i.e: [B], [I], [U], [*colorname*] etc). See the list of rich-formatting tags for more details.

Snapshot



Note
This feature are not available in stand-alone TrayIcon component and only comes with entire AppControls pack, since it uses function of acFormHelp component to display the popup window.

See also
PopupMenuWhatsThisItem, PopupMenu and Hint properties;
acFormHelp component and Rich-formatting tags.

© 1998-2006, UtilMind Solutions®

9.44.2.20Win2kInfoBalloon

Applies to
acTrayIcon component

Declaration
```
type
  TacWin2kInfoIcon = (iiNone, iiInfo, iiWarning, iiError);

  TacWin2kInfoBalloon = class(TPersistent)
  published
    property Enabled: Boolean; // display balloon when form minimized
    property Icon: TacWin2kInfoIcon; // the icon to display to the left
of the balloon tooltip text
    property NoSound: Boolean;
    property Text: String;   // specifies the information text
    property Title: String;  // the balloon title
    property Timeout: Word;  // The timeout value, in milliseconds, for
the display of the balloon-style tooltip
  end;
```

Description
The **Win2kInfoBalloon** structure contains the properties used to display some useful information after minimizing of the form to the system tray. This information will displayed in the cartoon-style "balloon" window which can be either single- or multi-line:

Remarks
The **Win2kInfoBalloon** supported in **Windows 2000 and later**. You can specify these properties in the Windows 95/98 or NT4, however, the balloon window can be displayed in the Windows 2000 systems only.

See also
Hint and WhatsThisHint properties;
acWin2kEffects component.

9.44.2.20.1 Enabled

Applies to
acTrayIcon component as subproperty of Win2kInfoBalloon structure.

Declaration
```
property Enabled: Boolean;
```

Description
The Enabled property determines whether you would like to display the information balloon when the window is minimized to the system tray or not. Set Enabled to True if you would like to display the information balloon when the icon appears in the system tray, or False otherwise.

If you would like to show the balloon tooltip when the form is already minimized to the system tray — use ShowInfoBalloon method.

© 1998-2006, UtilMind Solutions®

See also
ShowInfoBalloon method.

9.44.2.20.2 Icon

Applies to
acTrayIcon component as subproperty of Win2kInfoBalloon structure.

Declaration
```
type
  TacWin2kInfoIcon = (iiNone, iiInfo, iiWarning, iiError);

property Icon: TacWin2kInfoIcon;
```

Description
The Icon property specifies the icon image to display to the left of the balloon window.

Default value is `iiNone`, which means that no icon displayed. To show the "information" icon set the Icon property to `iiInfo`, to show the "warning" sign — use `iiWarning`, for "errors" use `iiError` value.

9.44.2.20.3 NoSound

Applies to
acTrayIcon component as subproperty of Win2kInfoBalloon structure.

Declaration
```
property NoSound: Boolean;
```

Description
The NoSound property specifies whether the system should utter sound when the balloon hint appears over tray icon. Set this property to True if you don't want a sound when the balloon appears.

Note
This property works only in Explorer v6.0 and higher (Windows XP+ or Windows 2000 with IE6).

9.44.2.20.4 Text

Applies to
acTrayIcon component as subproperty of Win2kInfoBalloon structure.

Declaration
```
property Text: String;
```

Description
The Text property specifies the information text to display in the cartoon-style balloon hint after minimizing of the form to the system tray. You can specify either single- or multi-line tooltip text.

Remark
The length of Text string is limited by 255 characters!

See also
Title property.

© 1998-2006, UtilMind Solutions®

9.44.2.20.5 Timeout

Applies to
acTrayIcon component as subproperty of Win2kInfoBalloon structure.

Declaration
```
property Timeout: Word; // in milliseconds
```

Description
The Timeout property determines the timeout value, in milliseconds, for the display of the balloon-style tooltip. If Timeout value outside of Window's allowable range, the timeout value will be "pushed" into the range of valid timeout periods. The minimum timeout value is 10 seconds, maximum is 30 seconds.

9.44.2.20.6 Title

Applies to
acTrayIcon component as subproperty of Win2kInfoBalloon structure.

Declaration
```
property Title: String;
```

Description
The Title property specifies the text to use as the title of balloon tooltip. This will appear in boldface above the regular text.

Remark
The length of Text string is limited by 63 characters!

See also
Text property.

9.44.3 Methods

9.44.3.1 MinimizeToTray

Applies to
acTrayIcon component.

Declaration
```
procedure MinimizeToTray;
```

Description
Call **MinimizeToTray** method to minimize your forms or whole application (if acTrayIcon component was dropped to the main form) to the system tray area. To restore minimized window from tray - call RestoreFromTray method.

However, acTrayIcon component can minimize forms to the system tray automatically when:

1. User clicks MinimizeToTray button on the form's title bar (if additional CaptionButton is Visible)
2. User selects "Minimize to Tray" menu item in the form's system menu (is ApplyToMenu property is True)
3. User clicks standard Minimize button and Options property contains enabled `oMinimizeToTray` flag.
4. User selects any menu item, assigned to PopupMenuTrayMinimizeItem property.

© 1998-2006, UtilMind Solutions®

See also
RestoreFromTray method;
IsMinimized, CaptionButton, SystemMenu, Options and PopupMenuTrayMinimizeItem properties;
OnMinimize and OnRestore events.

9.44.3.2 RestoreFromTray

Applies to
acTrayIcon component.

Declaration
```
procedure RestoreFromTray;
```

Description
Call **RestoreFromTray** method to restore your forms or whole application (if acTrayIcon component is on main form) to the system tray area. To minimize windows to tray - call MinimizeToTray method.

acTrayIcon component, however, can restore forms from the system tray automatically when:
1. User double clicks the tray icon and no popup menu assigned to PopupMenu property (If PopupMenu assigned, double click will invoke default menu action).
2. User clicks the tray icon once, if Options property contains enabled `oOneClickRestore` flag.
3. User selects any menu item assigned to PopupMenuRestoreItem property.

See also
MinimizeToTray method;
IsMinimized, Options, PopupMenu and PopupMenuRestoreItem properties;
OnMinimize and OnRestore events.

9.44.3.3 ShowInfoBalloon

Applies to
acTrayIcon component.

Declaration
```
procedure ShowInfoBalloon;
```

Description
The **ShowInfoBalloon** method shows the cartoon-style balloon tooltip with its stem pointing to the tray icon. The balloon will disappear when clicked, but you can also specify a time-out value.

To specify the text, title and icon for the balloon tooltip — use Win2kInfoBalloon structure.

If you would like to show the balloon tooltip when the form is minimized to the system tray, you don't need to call ShowInfoBalloon method. Just set Win2kInfoBalloon.Enabled property to True.

Snapshot

Remarks
The **Win2kInfoBalloon** supported in **Windows 2000 and later**. You can specify these properties in

© 1998-2006, UtilMind Solutions®

the Windows 95/98 or NT4, however, the balloon window can be displayed in the Windows 2000 systems only.

See also
ResetTrayIcon method;
Win2kInfoBalloon structure.

9.44.3.4 HideInfoBalloon

Applies to
acTrayIcon component.

Declaration
```
procedure HideInfoBalloon;
```

Description
The HideInfoBalloon method can be used to hide the information balloon over the tray icon before its timeout expired. This method just completely refreshes the tray icon in notification area (removes it from tray and imediately adds it again without displaying the balloon, even if Win2kInfoBalloon.Enabled property is True).

See also
ShowInfoBalloon method;
Win2kInfoBalloon structure.

9.44.4 Events

9.44.4.1 OnAnimate

Applies to
acTrayIcon component.

Declaration
```
property OnAnimate: TNotifyEvent;
```

Description
The OnAnimate event occurs when the animated tray icon changes the image increasing ImageIndex property.

See also
Animate, AnimateInterval, ImageList and ImageIndex properties.

9.44.4.2 OnCaptionDblClick

Applies to
acTrayIcon and acCaptionButton components.

Declaration
```
property OnCaptionDblClick: TNotifyEvent;
```

Description
The OnCaptionDblClick event notifies when user double clicks the caption of the form.

You can write this event handler to make some specific actions when user double clicks the title bar of the form with mouse, for example, minimize the form to the system tray using MinimizeToTray method.

© 1998-2006, UtilMind Solutions®

See also
MinimizeToTray method;
OnDblClick event.

9.44.4.3 OnClick

Applies to
acTrayIcon and acAppBar components.

Declaration
```
property OnClick: TNotifyEvent;
```

Description
The OnClick event occurs when user clicks the tray icon. Use the OnClick event handler to write code that responds when the user clicks the control.

See also
OnDblClick event.

9.44.4.4 OnBalloonHide

Applies to
acTrayIcon component.

Declaration
```
property OnBalloonHide: TNotifyEvent;
```

Description
The OnBalloonHide event occurs when the balloon tip disappears from screen.

Remark
For some reason the NIN_BALLOONHIDE message does not passes to WM_NOTIFYICON handler in Shell 5.x, so please use OnBalloonTimeOut and OnBalloonUserClick events instead.

See also
OnBalloonShow, OnBalloonHide, OnBalloonTimeOut and OnBalloonUserClick events;
Win2kInfoBalloon structure.

9.44.4.5 OnBalloonShow

Applies to
acTrayIcon component.

Declaration
```
property OnBalloonShow: TNotifyEvent;
```

Description
The OnBalloonShow event occurs when the balloon tip appears on screen over the tray icon.

See also
OnBalloonHide, OnBalloonTimeOut and OnBalloonUserClick events;
Win2kInfoBalloon structure.

9.44.4.6 OnBalloonTimeOut

Applies to
acTrayIcon component.

© 1998-2006, UtilMind Solutions®

Declaration
```
property OnBalloonTimeOut: TNotifyEvent;
```

Description
The OnBalloonTimeOut event occurs when the balloon tip disappears from screen due to TimeOut, OR because user closed the balloon by clicking "X" button on its caption.

See also
OnBalloonShow, OnBalloonHide and OnBalloonUserClick events;
Win2kInfoBalloon structure.

9.44.4.7 OnBalloonUserClick

Applies to
acTrayIcon component.

Declaration
```
property OnBalloonUserClick: TNotifyEvent;
```

Description
The OnBalloonUserClick event occurs when the balloon tip disappears from screen because user has clicked the balloon with mouse.

See also
OnBalloonTimeOut, OnBalloonShow and OnBalloonHide events;
Win2kInfoBalloon structure.

9.44.4.8 OnDblClick

Applies to
acTrayIcon and acAppBar components.

Declaration
```
property OnDblClick: TNotifyEvent;
```

Description
The OnDblClick even occurs when user double-clicks the primary mouse button over the tray icon. Use the OnDblClick event to write code that responds to the double-click event.

After firing of this event, the default double-click action will be executed. The default action is:
- the event handler of the *default menu item* in the PopupMenu, or…
- if the PopupMenu does not contains default menu item, but oDblClickRestore option is True, the form will be minimized, or restored from the system tray. To prevent your form from restoring on double click — set oDblClickRestore flag in Optons to False.

See also
OnClick event;
PopupMenu and Options properties.

9.44.4.9 OnHide

Applies to
acTrayIcon component.

Declaration
```
property OnHide: TNotifyEvent;
```

© 1998-2006, UtilMind Solutions®

Description
The OnHide event occurs when the tray icon hides from the system tray and Visible property gets False.

See also
OnShow event and Visible property.

9.44.4.10 OnMinimize

Applies to
acTrayIcon component.

Declaration
```
property OnMinimize: TNotifyEvent;
```

Description
The OnMinimize event occurs when the form minimizes to the system tray (after hiding).

See also
OnRestore event;
MinimizeToTray and RestoreFromTray methods;
IsMinimized property.

9.44.4.11 OnMouseDown

Applies to
acTrayIcon component.

Declaration
```
type
  TMouseEvent = procedure(Sender: TObject; Button: TMouseButton; Shift:
    TShiftState; X, Y: Integer) of object;

property OnMouseDown: TMouseEvent;
```

Description
The OnMouseDown event occurs when the user presses a mouse button with the mouse pointer over a tray icon. Use the OnMouseMove event handler to respond when the mouse pointer moves after the control has captured the mouse.

Use the Shift parameter of the OnMouseDown event handler, to determine to the state of the shift keys and mouse buttons. Shift keys are the Shift, Ctrl, and Alt keys or shift key-mouse button combinations. X and Y are pixel coordinates of the new location of the mouse pointer in the client area of the Sender.

See also
OnMouseUp and OnMouseMove events.

9.44.4.12 OnMouseMove

Applies to
acTrayIcon component.

Declaration
```
type
  TMouseMoveEvent = procedure(Sender: TObject; Shift: TShiftState; X, Y:
    Integer) of object;
```

© 1998-2006, UtilMind Solutions®

```
property OnMouseMove: TMouseMoveEvent;
```

Description

The OnMouseMove event occurs when the user moves the mouse pointer while the mouse pointer is over a tray icon. Use the OnMouseMove event handler to respond when the mouse pointer moves after the control has captured the mouse.

Use the Shift parameter of the OnMouseDown event handler, to determine to the state of the shift keys and mouse buttons. Shift keys are the Shift, Ctrl, and Alt keys or shift key-mouse button combinations. X and Y are pixel coordinates of the new location of the mouse pointer in the client area of the Sender.

See also

OnMouseDown and OnMouseUp events.

9.44.4.13OnMouseUp

Applies to

acTrayIcon component.

Declaration

```
type
  TMouseEvent = procedure(Sender: TObject; Button: TMouseButton; Shift:
    TShiftState; X, Y: Integer) of object;

property OnMouseUp: TMouseEvent;
```

Description

The OnMouseUp event occurs when the user releases a mouse button that was pressed with the mouse pointer over a tray icon. Use an OnMouseUp event handler to implement special processing when the user releases a mouse button.

The OnMouseUp event handler can respond to left, right, or center mouse button presses and shift key plus mouse-button combinations. Shift keys are the Shift, Ctrl, and Alt keys. X and Y are the pixel coordinates of the mouse pointer in the client area of the Sender.

See also

OnMouseDown and OnMouseMove events.

9.44.4.14OnPopupMenu

Applies to

acTrayIcon component.

Declaration

```
property OnPopupMenu: TNotifyEvent;
```

Description

The PopupMenu property identifies the pop-up menu associated with the tray icon. Assign a value to PopupMenu to make a pop-up menu appear when the user selects the control and clicks the right mouse button, or left mouse button and `oShowMenuOnLeftClick` specified in Options.

See also

PopupMenu and Options properties

© 1998-2006, UtilMind Solutions®

9.44.4.15OnRestore

Applies to
acTrayIcon component.

Declaration
```
property OnRestore: TNotifyEvent;
```

Description
The OnRestore event occurs when the form restores from the system tray. If `oHideWhenFormRestored` specified in Options, the tray icon will be hidden automatically.

See also
OnMinimize event;
MinimizeToTray and RestoreFromTray methods;
IsMinimized property.

9.44.4.16OnShow

Applies to
acTrayIcon component.

Declaration
```
property OnShow: TNotifyEvent;
```

Description
The OnShow event occurs when the tray icon appears on the system tray and Visible property gets True.

See also
OnHide event and Visible property.

9.45 acWavPlayer

9.45.1 TacWavPlayer

Overview
The acWavPlayer component can be used for playing default system sounds, associated with some certain events, or custom Wave-Audio (*.wav) files.

The type of sound controlled by SoundType property. It could be a custom (stCustom) or standard, system sound.

Custom sounds can be stored (uploaded) onto your form in WaveSound property editor at design-time and does not requires additional file in the supply package of your software. acWavPlayer component hold the files within a forms same way as it does the acFileStorage component.

Standard sounds is the sounds which associated with some system events (such as Windows Start or Program Minimize) and playing when any standard system event occurs.

How to use ?
To start playing the sound - call **Play** method. To terminate playing - call Stop method.

To upload a custom wave-audio file onto form - click on WaveSound property or click right mouse button on the component image and select "Upload WAV sound..." menu item.

To test the sound - double click on the component image.

© 1998-2006, UtilMind Solutions®

See also
Other components which stores (uploads) files onto Delphi/BCB form at design-time: acFileStorage, acWavPlayer.

9.45.2 Properties

9.45.2.1 Asynchronous

Applies to
acWavPlayer component.

Declaration
```
property Asynchronous: Boolean;
```

Description
The Asynchronous property controls whether sound will playing synchronously (if False) or asynchronously (if True, defauly), relatively to the main application thread. If Asynchronous is True, the sound will played asynchronously and the function returns immediately after beginning the sound, without suspending the main application thread. To terminate an asynchronously played sound, call Stop method.

See also
Looped and SoundType properties;
Play and Stop methods.

9.45.2.2 Looped

Applies to
acWavPlayer component.

Declaration
```
property Looped: Boolean;
```

Description
The Looped property controls whether sound sould be played once (if False) or should be played continuously (constantly looped). To terminate sound — use Stop method.

See also
Asynchronous and SoundType properties;
Play and Stop methods.

9.45.2.3 SoundType

Applies to
acWavPlayer component.

Declaration
```
procedure SoundType: TacSystemSoundType;
```

Description
The SoundType property specifies the type of a sound to play. When SoundType is stCustom, acWavPlayer will play custom sound specified in (uploaded to) WaveSound property (.WAV file uploaded onto form). When SoundType is any other value (*stAsterisk, stCloseProgram, stCriticalStop, stDefaultSound, stExclamation, stExitWindows, stMaximize, stMenuCommand, stMenuPopup, stMinimize, stNewMailNotification, stOpenProgram, stProgramError, stQuestion, stRestoreDown, stRestoreUp, stStartWindows*), acWavPlayer will play sound associated with enumerated system events.

© 1998-2006, UtilMind Solutions®

Note
After selecting of any system sound, content of WaveSound property will be cleared.
If you would like to play system sound event, associated with some certain action, you can alternativaly use PlaySystemSound and StopSystemSound routines.

See also
WaveSound property and TacWavPlayerSndType type;
Asynchronous and Looped properties;
Play and Stop methods.

9.45.2.4 WaveSound

Applies to
acWavPlayer component.

Declaration
```
property WaveSound: TStoredFile;
```

Description
The WaveSound property is the storage for custom wave-audio file. You may upload sound file at design-time, double clicking on WaveSound property in Object Inspector.

To test uploaded sound - double click on component image.

See also
acFileStorage component.

9.45.3 Methods

9.45.3.1 Play

Applies to
acWavPlayer component.

Declaration
```
procedure Play;
```

Description
The Play method plays a waveform sound specified either by uploaded onto form in WaveSound property or by system sound specified in SoundType property.

If Asynchronous property is True, sound will plays asynchronously and and the function returns immediately after beginning the sound.

Usually Play method initiates playing of the sound only once. However, If Looped property is True, sound will playing continuously. To terminate playing - call Stop method.

See also
Stop method.

9.45.3.2 Stop

Applies to
acWavPlayer component.

Declaration

© 1998-2006, UtilMind Solutions®

```
procedure Stop;
```

Description
Call Stop method to terminate playing the sound.

See also
Play method.

9.45.4 Events

9.45.4.1 OnAfterPlay

Applies to
acWavPlayer component.

Declaration
```
procedure OnAfterPlay: TNotifyEvent;
```

Description
The OnAfterPlay event occurs after playing the sound.

See also
OnBeforePlay event

9.45.4.2 OnBeforePlay

Applies to
acWavPlayer component.

Declaration
```
procedure OnBeforePlay: TNotifyEvent;
```

Description
The OnBeforePlay event occurs before playing the sound.

See also
OnAfterPlay event

9.46 acWin2kEffects

9.46.1 TacWin2kEffects

Overview
The acWin2kEffects component allows to produce some special effects of the Windows 98 / 2000. You can show roll, slide and alpha-blending animation effects on showing and hiding windows, make your forms translucent, flash the caption bars and taskbar icons.

Rolling / sliding animation and flashing effect works on Windows 98 and later. Blending animation and translucent forms supported in Windows 2000 and later. You can specify any of these effects on Windows 95 / Windows NT4 systems, but no effects will be shown.

Animation effects ?
You can produce special effects when showing (AnimateOnShow structure) or hiding windows (AnimateOnHide structure). There is three types of animation: roll, slide and alpha-blending. See possible animation Flags for more details.

© 1998-2006, UtilMind Solutions®

Translucent forms ?
The partially translucent, alpha-blended forms supported in Windows 2000. If you would like to make the form translucent — set Translucent property to True and specify the rate of alpha-blending in the TranslucentRate property.

Flashing windows ?
The acWin2kEffects component can flash the window caption and the application icon on task bar. Typically, you flash a window to inform the user that the window requires attention but does not currently have the keyboard focus. When a window flashes, it appears to change from inactive to active status. An inactive caption bar changes to an active caption bar; an active caption bar changes to an inactive caption bar. To flash the form — use Flash method. Number of times to flash the window specified in the FlashCount property.

See also
acAnimationEffect component.

9.46.2 Properties

9.46.2.1 AnimateOnHide

Applies to
acWin2kEffects and acAnimationEffect components.

Declaration
```
type
  TacWin2kAnimation = class(TPersistent)
  published
    property Enabled: Boolean;
    property Flags: TacWin2kAnimationFlags; // kind of animation
    property Time: Word; // duration of animation
  end;

property AnimateOnHide: TacWin2kAnimation;
```

Description
The AnimateOnHide structure controls the behaviors of the Win2k-based animation effect which will be shown when the form is about to disappear from the screen.

See also
AnimateOnShow property.

9.46.2.1.1 Enabled

Applies to
acWin2kEffects as subproperty of AnimateOnShow and AnimateOnHide structures.

Declaration
```
property Enabled: Boolean;
```

Description
The Enabled property controls whether you would like to show the animation effect on hiding the form. Kind of the animation effect can be specifyed in the Flags property.

See also
AnimateOnShow and AnimateOnHide structures.

© 1998-2006, UtilMind Solutions®

9.46.2.1.2 Flags

Applies to
acWin2kEffects as subproperty of AnimateOnShow and AnimateOnHide structures.

Declaration
```
type
  TacWin2kAnimationFlags = set of (afHorPositive, afHorNegative,
afVerPositive, afVerNegative, afCenter, afSlide, afBlend);

property Flags: TacWin2kAnimationFlags;
```

Description
Specifies the type of animation. This parameter can be one of the following values:

Value	Description
afHorPositive	Animates the window from left to right. This flag can be used with roll or slide animation. It is ignored when used with afCenter or afBlend flags.
afHorNegative	Animates the window from right to left. This flag can be used with roll or slide animation. It is ignored when used with afCenter or afBlend flags.
afVerPositive	Animates the window from top to bottom. This flag can be used with roll or slide animation. It is ignored when used with afCenter or afBlend flags.
afVerNegative	Animates the window from bottom to top. This flag can be used with roll or slide animation. It is ignored when used with afCenter or afBlend flags.
afCenter	Makes the window appear to collapse inward when form is hiding or expand outward if form is showing.
afSlide	Uses slide animation. By default, roll animation is used. This flag is ignored when used with afCenter.
afBlend	Uses a fade effect. This flag works in Win2000 or later *(does not works in Win95/98/NT4!)* and can be used only if you animating a top-level window.

Short tutorial
To show the rolling effect from left to right set the Flags property to [afHorPositive]:
```
acWin2kEffects1.AnimateOnHide.Flags := [afHorPositive];
```

To show sliding effect instead of rolling, just add the afSlide flag. For example, to show sliding effect from upper-right corner to bottom-left corner, set the Flags property to [afHorNegative, afVerPositive, afSlide]:
```
acWin2kEffects1.AnimateOnHide.Flags := [afHorNegative, afVerPositive,
  afSlide];
```

To show the alpha-blending effect, set the Flags property to [afBlend]:
```
acWin2kEffects1.AnimateOnHide.Flags := [afBlend];
```
Alpha-blending works on Windows 2000 and later only. If you will try to use blending effect on Windows 98, nothing will happends. When you're using fade effect, you don't need to specify any others flags.

See also
AnimateOnShow and AnimateOnHide structures.

9.46.2.1.3 Time

Applies to
acWin2kEffects as subproperty of AnimateOnShow and AnimateOnHide structures.

© 1998-2006, UtilMind Solutions®

Declaration

```
property Time: Word; // Duration of animation. Default 200
```

Description

The Time property specifies how long it takes to play the animation, in milliseconds. Typically, an animation takes 200 milliseconds to play.

See also

AnimateOnShow and AnimateOnHide structures.

9.46.2.2 AnimateOnShow

Applies to

acWin2kEffects and acAnimationEffect components.

Declaration

```
type
  TacWin2kAnimation = class(TPersistent)
  published
    property Enabled: Boolean;
    property Flags: TacWin2kAnimationFlags; // kind of animation
    property Time: Word; // duration of animation
  end;

property AnimateOnShow: TacWin2kAnimation;
```

Description

The AnimateOnShow structure controls the behaviors of the Win2k-based animation effect which will be shown when the form is about to appear on the screen.

See also

AnimateOnHide property.

9.46.2.2.1 Enabled

Applies to

acWin2kEffects as subproperty of AnimateOnShow and AnimateOnHide structures.

Declaration

```
property Enabled: Boolean;
```

Description

The Enabled property controls whether you would like to show the animation effect on showing the form. Kind of the animation effect can be specifyed in the Flags property.

See also

AnimateOnShow and AnimateOnHide structures.

9.46.2.2.2 Flags

Applies to

acWin2kEffects as subproperty of AnimateOnShow and AnimateOnHide structures.

Declaration

```
type
  TacWin2kAnimationFlags = set of (afHorPositive, afHorNegative,
afVerPositive, afVerNegative, afCenter, afSlide, afBlend);
```

© 1998-2006, UtilMind Solutions®

```
property Flags: TacWin2kAnimationFlags;
```

Description

Specifies the type of animation. This parameter can be one of the following values:

Value	Description
afHorPositive	Animates the window from left to right. This flag can be used with roll or slide animation. It is ignored when used with afCenter or afBlend flags.
afHorNegative	Animates the window from right to left. This flag can be used with roll or slide animation. It is ignored when used with afCenter or afBlend flags.
afVerPositive	Animates the window from top to bottom. This flag can be used with roll or slide animation. It is ignored when used with afCenter or afBlend flags.
afVerNegative	Animates the window from bottom to top. This flag can be used with roll or slide animation. It is ignored when used with afCenter or afBlend flags.
afCenter	Makes the window appear to collapse inward when form is hiding or expand outward if form is showing.
afSlide	Uses slide animation. By default, roll animation is used. This flag is ignored when used with afCenter.
afBlend	Uses a fade effect. This flag works in Win2000 or later *(does not works in Win95/98/NT4!)* and can be used only if you animating a top-level window.

Short tutorial

To show the rolling effect from left to right set the Flags property to [afHorPositive]:

```
acWin2kEffects1.AnimateOnShow.Flags := [afHorPositive];
```

To show sliding effect instead of rolling, just add the afSlide flag. For example, to show sliding effect from upper-right corner to bottom-left corner, set the Flags property to [afHorNegative, afVerPositive, afSlide]:

```
acWin2kEffects1.AnimateOnShow.Flags := [afHorNegative, afVerPositive,
  afSlide];
```

To show the alpha-blending effect, set the Flags property to [afBlend]:

```
acWin2kEffects1.AnimateOnShow.Flags := [afBlend];
```

Alpha-blending works on Windows 2000 and later only. If you will try to use blending effect on Windows 98, nothing will happends. When you're using fade effect, you don't need to specify any others flags.

See also

AnimateOnShow and AnimateOnHide structures.

9.46.2.2.3 Time

Applies to

acWin2kEffects as subproperty of AnimateOnShow and AnimateOnHide structures.

Declaration

```
property Time: Word; // Duration of animation. Default 200
```

Description

The Time property specifies how long it takes to play the animation, in milliseconds. Typically, an animation takes 200 milliseconds to play.

See also

AnimateOnShow and AnimateOnHide structures.

© 1998-2006, UtilMind Solutions®

9.46.2.3 FlashCount

Applies to
acWin2kEffects component.

Declaration
```
property FlashCount: Word; // Default 4
```

Description
The FlashCount property specifies the number of times to flash the window. To flash the window and/or task icon — use Flash method.

Requirements
Windows NT/2000: Requires Windows 2000 or later.
Windows 95/98: Requires Windows 98 or later.

If you call this function on Win95/NT4 — nothing will happends.

See also
FlashTaskIcon property and Flash method.

9.46.2.4 FlashTaskIcon

Applies to
acWin2kEffects component.

Declaration
```
property FlashTaskIcon: Boolean;
```

Description
The FlashTaskIcon property controls whether you would like to flash the application icon on the taskbar at the same time with the caption. Set FlashTaskIcon to True to flash the task icon or False to flash only window caption.

The number of times to flash the window can be specified in the FlashCount property.

Requirements
Windows NT/2000: Requires Windows 2000 or later.
Windows 95/98: Requires Windows 98 or later.

If you call this function on Win95/NT4 — nothing will happends.

See also
FlashCount property and Flash method.

9.46.2.5 Translucent

Applies to
acWin2kEffects component.

Declaration
```
property Translucent: Boolean;
```

Description
The Translucent property controls whether the form is translucent, that is, alpha-blended. Set Translucent to True to make the form partally translucent, by rate specified in the TranslucentRate

© 1998-2006, UtilMind Solutions®

property.

The translucent forms supported by Windows 2000 and later. You can specify this property in lower Windows versions but nothing will happends.

See also
TranslucentRate property.

9.46.2.6 TranslucentRate

Applies to
acWin2kEffects component.

Declaration
```
type
  TacWin2kTranslucentRate = 0..100;

property TranslucentRate: TacWin2kTranslucentRate; // 0%..100%
```

Description
The TranslucentRate controls the transparency rate, in percents, of the alpha-blended form, when the Translucent property is True.

The translucent forms supported by Windows 2000 and later. You can specify this property in lower Windows versions but nothing will happends.

See also
Translucent property.

9.46.3 Methods

9.46.3.1 Flash

Applies to
acWin2kEffects component.

Declaration
```
procedure Flash;
```

Description
The Flash method flashes the caption of current form + the taskbar icon if the FlashTaskIcon property is True. Number of times to flash the window can be specifyed in the FlashCount property.

The Flash method flashes the window but does not change the active state of the window.

Typically, you flash a window to inform the user that the window requires attention but does not currently have the keyboard focus. When a window flashes, it appears to change from inactive to active status. An inactive caption bar changes to an active caption bar; an active caption bar changes to an inactive caption bar.

Requirements
Windows NT/2000: Requires Windows 2000 or later.
Windows 95/98: Requires Windows 98 or later.

If you call this function on Win95/NT4 — nothing will happends.

© 1998-2006, UtilMind Solutions®

See also
FlashBackground method;
FlashCount, FlashTaskIcon properties.

9.46.3.2 FlashBackground

Applies to
acWin2kEffects component.

Declaration
```
procedure FlashBackground;
```

Description
The FlashBackground method flashes the caption and window icon of current form *continuously*, until the window comes to the foreground.

Typically, you flash a window to inform the user that the window requires attention but does not currently have the keyboard focus. When a window flashes, it appears to change from inactive to active status. An inactive caption bar changes to an active caption bar; an active caption bar changes to an inactive caption bar.

Requirements
Windows NT/2000: Requires Windows 2000 or later.
Windows 95/98: Requires Windows 98 or later.

See also
Flash method.

9.46.4 Events

9.46.4.1 OnShowEffect

Applies to
acWin2kEffects component.

Declaration
```
property OnShowEffect: TNotifyEvent;
```

Description
The OnShowEffect event occurs when the form is about to appear on screen and the component preparing to produce the animation effect specified in AnimateOnShow structure.

See also
OnHideEffect event.

9.46.4.2 OnHideEffect

Applies to
acWin2kEffects component.

Declaration
```
property OnHideEffect: TNotifyEvent;
```

Description
The OnHideEffect event occurs when the form is about to be closed or hidden and the component is about to produce the animation effect specified in AnimateOnHide structure.

© 1998-2006, UtilMind Solutions®

See also
OnHideEffect event.

10 Components (Part II)

10.1 acApartPopupMenu

10.1.1 TacApartPopupMenu

Overview
The acApartPopupMenu is the PopupMenu component which properly works in several instances of application running as ActiveX control in *Apartment* thread model.

If you ever had pain adapting the standard PopupMenus (with single PopupList object per application) when written DLL forms for, let's say, Internet Explorer toolbars, then this popup menu is what you need.

Note

This component only available in Delphi 5 / C++ Builder 5 and higher.

10.1.2 Methods

10.1.2.1 PopupFromCursorPos

Applies to
acApartPopupMenu component.

Declaration
```
procedure PopupFromCursorPos;
```

Description
The PopupFromCursorPos method displays the pop-up menu on screen at current mouse position.

10.1.3 Events

10.1.3.1 OnCloseUp

Applies to
acApartPopupMenu component.

Declaration
```
property OnCloseUp: TNotifyEvent;
```

Description
The OnCloseUp event occurs when the menu disappears from screen.

Write the OnCloseUp event handler to take specific actions after the pop-up menu disappears from screen.

10.2 acBitBtn

10.2.1 TacBitBtn

Overview
The acBitBtn component is descendant of TacButton (not standard TBitBtn), which can include a

© 1998-2006, UtilMind Solutions®

bitmap image, or picture from ImageLists on its button face, automatically create glyphs in disabled state and eat much less number of system resources. It also supports multiline captions (see WordWrap property), custom background colors (see Color) and can be aligned within its container (see Align). The caption of button can include rich-style tags (like in acRichLabel or acFormHelp's hints).

Snapshots

Additional / Improved properties

Align	controls placement of button within its container or parent control;
Caption	specifies the text of the button. To separate single line of text on several lines, even if single line fits the button width, use "|" characters;
Color	specifies the background color for button;
WordWrap	specifies whether the button text wraps when it is too long for the width of the button. When WordWrap is True, caption can be multiline.

New events

OnMouseEnter	occurs when mouse pointer hover button;
OnMouseLeave	occurs when mouse pointer leave button.

See also

acButton and acButtonsPanel components.

10.2.2 Properties

10.2.2.1 Align

Applies to

acButton, acBitBtn, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acIPEdit, acMRUComboBox, acLabeledComboBox, acTreeComboBox and acButtonsPanel components *(in generally to ALL successors of TControl)*.

Declaration

```
type
  TAlign = (alNone, alTop, alBottom, alLeft, alRight, alClient);

property Align: TAlign;
```

Declaration

The Align property determines how the control aligns within its container (or parent control).

Use Align to align a control to the top, bottom, left, or right of a form or panel and have it remain there even if the size of the form, panel, or component that contains the control changes.

For example, to use a panel component with various controls on it as a tool palette, change the panel's Align value to alLeft. The value of alLeft for the Align property of the panel guarantees that the tool palette remains on the left side of the form and always equals the client height of the form.

The default value of Align is alNone, which means a control remains where it is positioned on a form or panel. Specify another value to align the control. Use one of these possible values:

Value	Meaning
alNone	The component remains where it was placed on the form or panel. This is the default value.
alTop	The component moves to the top of the form and resizes to fill the width of the form.

© 1998-2006, UtilMind Solutions®

The height of the component is not affected.

`alBottom` The component moves to the bottom of the form and resizes to fill the width of the form. The height of the component is not affected.

`alLeft` The component moves to the left side of the form and resizes to fill the height of the form. The width of the component is not affected.

`alRight` The component moves to the right side of the form and resizes to fill the height of the form. The width of the component is not affected.

`alClient` The component resizes to fill the client area of a form. If a component already occupies part of the client area, the component resizes to fit within the remaining client area.

Note

If Align is set to alClient, the control fills the entire client area so that it is impossible to select the parent form by clicking on it. In this case, select the parent by selecting the control on the form and pressing Esc, or by using the Object Inspector.

Any number of child components within a single parent can have the same Align value, in which case they stack up along the edge of the parent form or panel. The child controls stack up in z-order. To adjust the order in which the controls stack up, drag the controls into their desired positions.

10.2.2.2 Caption

Applies to

acBitBtn, acButton, acCheckGroupBox and acRichLabel components *(to all visible controls, in general)*

Declaration

```
property Caption: String;
```

Description

The Caption property specifies the text for the button.

To separate the lines in the multiline caption set WordWrap to True, and use linebreaks (#13#10) as separators.

To underline a character in a Caption that labels a component, include an ampersand (&) before the character. This type of character is called an accelerator character. The user can then select the component by pressing Alt while typing the underlined character. To display an ampersand character in the caption, use two ampersands (&&).

If the button's glyph does not fits the buton width and shifts behind left edge, use Margin property to specify spacing between the glyph and left edge.

See also

WordWrap property.

10.2.2.3 Color

Applies to

acBitBtn component.

Declaration

```
property Color: TColor;
```

© 1998-2006, UtilMind Solutions®

Description
The Color property specifies the background color of the button. Use Color property to read or change the background color of the control.

If a button's ParentColor property is True, then changing the Color property of the button's parent automatically changes the Color property of the button. When the value of the Color property is changed, the button's ParentColor property is automatically set to False.

[AK] I guess that guys from Borland simply forgot about the Color for standard TBitBtn…

See also
ParentColor property.

10.2.2.4 ParentColor

Applies to
acBitBtn component.

Declaration
```
property ParentColor: Boolean;
```

Description
The ParentColor property for the acBitBtn component determines where a button looks for its color information.

To have a control use the same color as its parent control, set ParentColor to True. For example, if the ParentColor of a button on a panel is True, the button has the same color as the panel. If ParentColor is False, the control uses its own Color property.

Set ParentColor to True for all controls in order to ensure that all the controls on a form have a uniform appearance. For example, if ParentColor is True for all controls in a form, changing the background color of the form to gray causes all the controls on the form to also have a gray background.

To specify a different color for a particular control, assign the desired color to that control's Color property, and ParentColor becomes False automatically.

See also
Color property.

10.2.2.5 WordWrap

Applies to
acBitBtn, acButton, acLabel, acURLLabel, acRichLabel, acCheckBox and acRadioButton components.

Declaration
```
property WordWrap: Boolean;
```

Description
The WordWrap property specifies whether the button text wraps when it is too long for the width of the button. When WordWrap is True, caption can be multiline, and single line when WordWrap is False.

See also
Caption property.

© 1998-2006, UtilMind Solutions®

10.2.3 Events

10.2.3.1 OnMouseEnter

Applies to
acLabel, acURLLabel, acRichLabel, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox, acTreeComboBox, acHeader, acHeaderPanel, acTreeView and acListView components.

Declaration
```
property OnMouseEnter: TNotifyEvent;
```

Description
The OnMouseEnter occurs when mouse pointer hovers the button.

See also
OnMouseLeave event.

10.2.3.2 OnMouseLeave

Applies to
acLabel, acURLLabel, acRichLabel, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox, acTreeComboBox, acHeader, acHeaderPanel, acTreeView and acListView components.

Declaration
```
property OnMouseLeave: TNotifyEvent;
```

Description
The OnMouseLeave occurs when mouse pointer leaves the button.

See also
OnMouseEnter event.

10.3 acButton

10.3.1 TacButton

Overview
The acButton component is improved TButton, which supports multiline caption (see WordWrap property) and can be aligned within its container (see Align).

Since this is just enchancement of standard TButton, you can find out about its basic features in the Delphi/BCB documentation (but you probably already know how to use Buttons. If not — select necessary property in Object Inspector and press F1). This documentation descibes only additional features and improvements.

Snapshot

Additional / improved properties

Align	controls placement of button within its container or parent control;
Caption	specifies the text of the button. To separate single line of text on several lines, even if single line fits the button width, use "|" characters;

© 1998-2006, UtilMind Solutions®

WordWrap specifies whether the button text wraps when it is too long for the width of the button. When WordWrap is True, caption can be multiline.

New events

OnMouseEnter occurs when mouse pointer hover button;
OnMouseLeave occurs when mouse pointer leave button.

See also

acBitBtn and acButtonsPanel components.

10.3.2 Properties

10.3.2.1 Align

Applies to

acButton, acBitBtn, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acIPEdit, acMRUComboBox, acLabeledComboBox, acTreeComboBox components *(in generally to ALL successors of TControl)*.

Declaration

```
type
  TAlign = (alNone, alTop, alBottom, alLeft, alRight, alClient);

property Align: TAlign;
```

Declaration

The Align property determines how the control aligns within its container (or parent control).

Use Align to align a control to the top, bottom, left, or right of a form or panel and have it remain there even if the size of the form, panel, or component that contains the control changes.

For example, to use a panel component with various controls on it as a tool palette, change the panel's Align value to alLeft. The value of alLeft for the Align property of the panel guarantees that the tool palette remains on the left side of the form and always equals the client height of the form.

The default value of Align is alNone, which means a control remains where it is positioned on a form or panel. Specify another value to align the control. Use one of these possible values:

Value	Meaning
alNone	The component remains where it was placed on the form or panel. This is the default value.
alTop	The component moves to the top of the form and resizes to fill the width of the form. The height of the component is not affected.
alBottom	The component moves to the bottom of the form and resizes to fill the width of the form. The height of the component is not affected.
alLeft	The component moves to the left side of the form and resizes to fill the height of the form. The width of the component is not affected.
alRight	The component moves to the right side of the form and resizes to fill the height of the form. The width of the component is not affected.
alClient	The component resizes to fill the client area of a form. If a component already occupies part of the client area, the component resizes to fit within the remaining client area.

Note

© 1998-2006, UtilMind Solutions®

If Align is set to alClient, the control fills the entire client area so that it is impossible to select the parent form by clicking on it. In this case, select the parent by selecting the control on the form and pressing Esc, or by using the Object Inspector.

Any number of child components within a single parent can have the same Align value, in which case they stack up along the edge of the parent form or panel. The child controls stack up in z-order. To adjust the order in which the controls stack up, drag the controls into their desired positions.

10.3.2.2 Caption

Applies to
acButton, acBitBtn, acCheckGroupBox and acRichLabel components *(to all visible controls, in general)*

Declaration
```
property Caption: String;
```

Description
The Caption property specifies the text for the button.

To separate the lines in the multiline caption set WordWrap to True, and use linebreaks (#13#10) as separators.

To underline a character in a Caption that labels a component, include an ampersand (&) before the character. This type of character is called an accelerator character. The user can then select the component by pressing Alt while typing the underlined character. To display an ampersand character in the caption, use two ampersands (&&).

See also
WordWrap property.

10.3.2.3 WordWrap

Applies to
acButton, acBitBtn, acLabel, acURLLabel, acRichLabel, acCheckBox and acRadioButton components.

Declaration
```
property WordWrap: Boolean;
```

Description
The WordWrap property specifies whether the button text wraps when it is too long for the width of the button. When WordWrap is True, caption can be multiline, and single line when WordWrap is False.

See also
Caption property.

10.3.3 Events

10.3.3.1 OnMouseEnter

Applies to
acLabel, acURLLabel, acRichLabel, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox, acTreeComboBox, acHeader, acHeaderPanel, acTreeView and acListView components.

© 1998-2006, UtilMind Solutions®

Declaration

```
property OnMouseEnter: TNotifyEvent;
```

Description

The OnMouseEnter occurs when mouse pointer hovers the button.

See also

OnMouseLeave event.

10.3.3.2 OnMouseLeave

Applies to

acLabel, acURLLabel, acRichLabel, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox, acTreeComboBox, acHeader, acHeaderPanel, acTreeView and acListView components.

Declaration

```
property OnMouseLeave: TNotifyEvent;
```

Description

The OnMouseLeave occurs when mouse pointer leaves the button.

See also

OnMouseEnter event.

10.4 acButtonsPanel

10.4.1 TacButtonsPanel

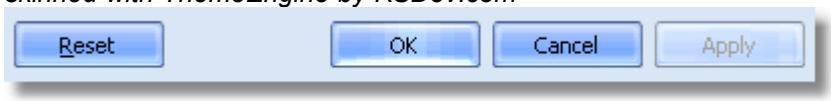
Overview

The acButtonsPanel is an easy and elegant way to put standard buttons (like OK, Cancel, Apply, Help or some custom buttons) to your dialog forms. Supports vertical and horizontal buttons (TacBitBtn's), can show the button glyphs, the divider between buttons and the dialog contents and even size grip at the bottom-right corner.

Snapshots

standard buttons

skinned with ThemeEngine by KSDev.com



How to use?

First, drop the ButtonsPanel onto your dialog form and choose the placement of buttons (how the buttons should be aligned on the form, vertically or horizontally, see Align property).

By default there is 3 visible buttons: OK, Cancel and Apply. If you don't need the Apply button, you can make it invisible, just set the BtnApply.Visible property to False. To take some specific actions when user clicks OK or Apply buttons (for example, save the program settings changed in the dialog box), write **OnApplySettings** event handler.

If you need some custom button (for example, "Browse" or "Search"), set BtnCustom property to

© 1998-2006, UtilMind Solutions®

True, specify its text to BtnCustom.Caption property, and write OnBtnCustomClick event handler to take some specific actions when user clicks this button. If you want to display the custom button *before* other buttons (at the left or top of OK) — set CustomButtonPosition property to True.

To control the width and height of buttons — use ButtonWidth and ButtonHeight properties. To increase or decrease the spacing between buttons — set ButtonSpacing property.

Also the acButtonsPanel have some miscellaneous properties like ShowDivider (can display the thin divider line between buttons and the content of the dialog box), and ShowSizeGrip (displays the resize grip at the bottom-right corner of panel, useful for resizeable dialog boxes).

See also
acBitBtn component.

10.4.2 Properties

10.4.2.1 Align

Applies to
acButtonsPanel component.

Declaration
```
type
  TacButtonsPanelAlign = (alHorizontal, alVertical);

property Align: TacButtonsPanelAlign;
```

Declaration
The Align property determines how the control aligns within its container (or parent control).

There are to possible values for Align property of acButtonsPanel component: alHorizontal and alVertical.

When the Align = alHorizontal, the panel with buttons are displayed at the bottom of dialog box. When Align is alVertical, the panel displayed at the right side and all its buttons placed vertically, each next under previous.

If you wish to arrange the buttons inside the panel by left/bottom side, or align them by center of panel — use Alignment property instead.

To change the size of buttons — modify the ButtonWidth and ButtonHeight properties. To control the spacing between buttons — use ButtonSpacing property.

See also
Alignment, ButtonWidth, ButtonHeight, ButtonSpacing, MarginHorz and MarginVert properties.

10.4.2.2 Alignment

Applies to
acButtonsPanel component.

Declaration
```
type
  TAlignment = (taLeftJustify, taRightJustify, taCenter);

property Alignment: TAlignment;
```

© 1998-2006, UtilMind Solutions®

Declaration
The Alignment property controls placement of buttons inside of the panel.

You can align the buttons by left or right side (top or bottom side for vertical panel, when Align is alVertical), or put them to center of panel.

To put the buttons to center of panel — set Alignment to taCenter.

If you need to specify the placement of entire panel (vertical or hizontal) within a form — use Align property instead.

See also
Align, ButtonWidth, ButtonHeight, ButtonSpacing, MarginHorz and MarginVert properties.

10.4.2.3 ApplyEnabled

Applies to
acButtonsPanel component.

Declaration
```
property ApplyEnabled: Boolean;
```

Declaration
The ApplyEnabled property controls whether the Apply button is enabled and whether user can press it.

This is just alternative way of modifying of BtnApply.Enabled property.

See also
BtnApply property;
OnApplyEnabledChanged event.

10.4.2.4 BtnApply

Applies to
acButtonsPanel component.

Declaration
```
type // this structure introduced in acButtons unit
  TacPanelBitBtn = class(TPersistent)
  public
    property Button: TacBitBtn read GetButton;
  published
    property BiDiMode: TBiDiMode;
    property Caption: TCaption;
    property Color: TColor;
    property Cursor: TCursor;
    property Enabled: Boolean;
    property Hint: String;
    property Font: TFont;
    property ParentBiDiMode: Boolean;
    property ParentColor: Boolean;
    property ParentFont: Boolean;
    property ParentShowHint: Boolean;
    property PopupMenu: TPopupMenu;
    property ShowHint: Boolean;
```

© 1998-2006, UtilMind Solutions®

```
    property Visible: Boolean;

    property Glyph: TBitmap;
    property GlyphLayout: TacBitBtnGlyphLayout default glGlyphLeft;
    property ImageIndex: TacImageIndex default -1;
    property Kind: TacBitBtnKind default bkCustom;
    property ModalResult: TModalResult default mrNone;
    property ShowFocus: Boolean default True;
    property TextStyle: TacTextStyle;
  end;

property BtnOK: TacPanelBitBtn;
```

Declaration

The BtnOK property used to specify the properties of appearance and behaviour of "Apply" button which appears on ButtonsPanel.

To specify custom text for button — use *Caption* property. To make the button visible or invisible — modify *Visible* property. To disallow user to press the button — set *Enabled* property to False. Also you can specify the custom icon which appears on the button using *Glyph* property, or sepecify the icon index to *ImageIndex* property (if you decided to use an icon from some ImageList specified in Images property).

To be notified when user clicked Apply button — write OnApplyClick event handler.

See also

ApplyEnabled, BtnOk, BtnCancel, BtnApply, BtnHelp and BtnCustom properties; OnAnyBtnClick, OnApplySettings, OnBtnOkClick, OnBtnCancelClick, OnBtnApplyClick, OnBtnHelpClick and OnBtnCustomClick events.

10.4.2.5 BtnCancel

Applies to

acButtonsPanel component.

Declaration

```
type // this structure introduced in acButtons unit
  TacPanelBitBtn = class(TPersistent)
  public
    property Button: TacBitBtn read GetButton;
  published
    property BiDiMode: TBiDiMode;
    property Caption: TCaption;
    property Color: TColor;
    property Cursor: TCursor;
    property Enabled: Boolean;
    property Hint: String;
    property Font: TFont;
    property ParentBiDiMode: Boolean;
    property ParentColor: Boolean;
    property ParentFont: Boolean;
    property ParentShowHint: Boolean;
    property PopupMenu: TPopupMenu;
    property ShowHint: Boolean;
    property Visible: Boolean;

    property Glyph: TBitmap;
```

© 1998-2006, UtilMind Solutions®

```
    property GlyphLayout: TacBitBtnGlyphLayout default glGlyphLeft;
    property ImageIndex: TacImageIndex default -1;
    property Kind: TacBitBtnKind default bkCustom;
    property ModalResult: TModalResult default mrNone;
    property ShowFocus: Boolean default True;
    property TextStyle: TacTextStyle;
  end;

property BtnOK: TacPanelBitBtn;
```

Declaration

The BtnOK property used to specify the properties of appearance and behaviour of "Cancel" button which appears on ButtonsPanel.

To specify custom text for button — use *Caption* property. To make the button visible or invisible — modify *Visible* property. To disallow user to press the button — set *Enabled* property to False. Also you can specify the custom icon which appears on the button using *Glyph* property, or sepecify the icon index to *ImageIndex* property (if you decided to use an icon from some ImageList specified in Images property).

To be notified when user clicked Help button — write OnBtnCancelClick event handler.

See also

BtnOk, BtnCancel, BtnApply, BtnHelp and BtnCustom properties;
OnAnyBtnClick, OnApplySettings, OnBtnOkClick, OnBtnCancelClick, OnBtnApplyClick, OnBtnHelpClick and OnBtnCustomClick events.

10.4.2.6 BtnCustom

Applies to

acButtonsPanel component.

Declaration

```
type // this structure introduced in acButtons unit
  TacPanelBitBtn = class(TPersistent)
  public
    property Button: TacBitBtn read GetButton;
  published
    property BiDiMode: TBiDiMode;
    property Caption: TCaption;
    property Color: TColor;
    property Cursor: TCursor;
    property Enabled: Boolean;
    property Hint: String;
    property Font: TFont;
    property ParentBiDiMode: Boolean;
    property ParentColor: Boolean;
    property ParentFont: Boolean;
    property ParentShowHint: Boolean;
    property PopupMenu: TPopupMenu;
    property ShowHint: Boolean;
    property Visible: Boolean;

    property Glyph: TBitmap;
    property GlyphLayout: TacBitBtnGlyphLayout default glGlyphLeft;
    property ImageIndex: TacImageIndex default -1;
    property Kind: TacBitBtnKind default bkCustom;
```

© 1998-2006, UtilMind Solutions®

```
    property ModalResult: TModalResult default mrNone;
    property ShowFocus: Boolean default True;
    property TextStyle: TacTextStyle;
  end;

property BtnOK: TacPanelBitBtn;
```

Declaration

The BtnOK property used to specify the properties of appearance and behaviour of *custom* button which appears on ButtonsPanel.

To specify custom text for button — use *Caption* property. To make the button visible or invisible — modify *Visible* property. To disallow user to press the button — set *Enabled* property to False. Also you can specify the custom icon which appears on the button using *Glyph* property, or sepecify the icon index to *ImageIndex* property (if you decided to use an icon from some ImageList specified in Images property).

To be notified when user clicked Help button — write OnBtnCustomClick event handler.

If you would like to show the custom button BEFORE the OK button — set CustomButtonPosition property to True.

See also

BtnOk, BtnCancel, BtnApply, BtnHelp, BtnCustom and CustomButtonPosition properties; OnAnyBtnClick, OnApplySettings, OnBtnOkClick, OnBtnCancelClick, OnBtnApplyClick, OnBtnHelpClick and OnBtnCustomClick events.

10.4.2.7 BtnHelp

Applies to

acButtonsPanel component.

Declaration

```
type // this structure introduced in acButtons unit
  TacPanelBitBtn = class(TPersistent)
  public
    property Button: TacBitBtn read GetButton;
  published
    property BiDiMode: TBiDiMode;
    property Caption: TCaption;
    property Color: TColor;
    property Cursor: TCursor;
    property Enabled: Boolean;
    property Hint: String;
    property Font: TFont;
    property ParentBiDiMode: Boolean;
    property ParentColor: Boolean;
    property ParentFont: Boolean;
    property ParentShowHint: Boolean;
    property PopupMenu: TPopupMenu;
    property ShowHint: Boolean;
    property Visible: Boolean;

    property Glyph: TBitmap;
    property GlyphLayout: TacBitBtnGlyphLayout default glGlyphLeft;
    property ImageIndex: TacImageIndex default -1;
```

© 1998-2006, UtilMind Solutions®

```
    property Kind: TacBitBtnKind default bkCustom;
    property ModalResult: TModalResult default mrNone;
    property ShowFocus: Boolean default True;
    property TextStyle: TacTextStyle;
  end;

property BtnOK: TacPanelBitBtn;
```

Declaration

The BtnOK property used to specify the properties of appearance and behaviour of "OK" button which appears on ButtonsPanel.

To specify custom text for button — use *Caption* property. To make the button visible or invisible — modify *Visible* property. To disallow user to press the button — set *Enabled* property to False. Also you can specify the custom icon which appears on the button using *Glyph* property, or sepecify the icon index to *ImageIndex* property (if you decided to use an icon from some ImageList specified in Images property).

To be notified when user clicked Help button — write OnBtnHelpClick event handler.

See also

BtnOk, BtnCancel, BtnApply, BtnHelp and BtnCustom properties;
OnAnyBtnClick, OnApplySettings, OnBtnOkClick, OnBtnCancelClick, OnBtnApplyClick, OnBtnHelpClick and OnBtnCustomClick events.

10.4.2.8 BtnOK

Applies to

acButtonsPanel component.

Declaration

```
type // this structure introduced in acButtons unit
  TacPanelBitBtn = class(TPersistent)
  public
    property Button: TacBitBtn read GetButton;
  published
    property BiDiMode: TBiDiMode;
    property Caption: TCaption;
    property Color: TColor;
    property Cursor: TCursor;
    property Enabled: Boolean;
    property Hint: String;
    property Font: TFont;
    property ParentBiDiMode: Boolean;
    property ParentColor: Boolean;
    property ParentFont: Boolean;
    property ParentShowHint: Boolean;
    property PopupMenu: TPopupMenu;
    property ShowHint: Boolean;
    property Visible: Boolean;

    property Glyph: TBitmap;
    property GlyphLayout: TacBitBtnGlyphLayout default glGlyphLeft;
    property ImageIndex: TacImageIndex default -1;
    property Kind: TacBitBtnKind default bkCustom;
    property ModalResult: TModalResult default mrNone;
    property ShowFocus: Boolean default True;
```

© 1998-2006, UtilMind Solutions®

```
    property TextStyle: TacTextStyle;
  end;

property BtnOK: TacPanelBitBtn;
```

Declaration
The BtnOK property used to specify the properties of appearance and behaviour of "OK" button which appears on ButtonsPanel.

To specify custom text for button — use *Caption* property. To make the button visible or invisible — modify *Visible* property. To disallow user to press the button — set *Enabled* property to False. Also you can specify the custom icon which appears on the button using *Glyph* property, or sepecify the icon index to *ImageIndex* property (if you decided to use an icon from some ImageList specified in Images property).

To be notified when user clicked Help button — write OnBtnOkClick event handler.

See also
BtnOk, BtnCancel, BtnApply, BtnHelp and BtnCustom properties;
OnAnyBtnClick, OnApplySettings, OnBtnOkClick, OnBtnCancelClick, OnBtnApplyClick, OnBtnHelpClick and OnBtnCustomClick events.

10.4.2.9 ButtonHeight

Applies to
acButtonsPanel component.

Declaration
```
property ButtonHeight: Boolean;
```

Declaration
The ButtonHeight property controls the height of ALL buttons on the panel.

See also
ButtonWidth, MarginHorz, MarginVert and ButtonSpacing properties.

10.4.2.10 ButtonSpacing

Applies to
acButtonsPanel component.

Declaration
```
property ButtonSpacing: Boolean;
```

Declaration
The ButtonSpacing property controls the size (in pixels) of the space between buttons on the panel.

See also
ButtonWidth, ButtonHeight, MarginHorz and MarginVert properties.

10.4.2.11 ButtonWidth

Applies to
acButtonsPanel component.

Declaration
```
property ButtonWidth: Boolean;
```

© 1998-2006, UtilMind Solutions®

Declaration
The ButtonWidth property controls the width of ALL buttons on the panel.

See also
ButtonHeight, MarginHorz, MarginVert and ButtonSpacing properties.

10.4.2.12CustomButtonPosition

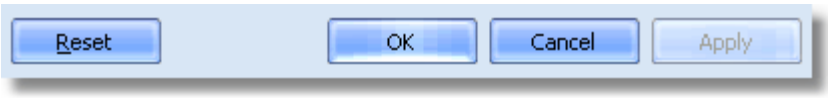
Applies to
acButtonsPanel component.

Declaration
```
type
  TacCustomButtonPosition = (cbpAfterLast, cbpBeforeFirst,
cbpOppositeEdge);

property CustomButtonPosition: TacCustomButtonPosition;
```

Declaration
The CustomButtonPosition property controls the placement of the Custom button (button which performs some custom action) on the buttons panel.

There is three possible positions for custom button:

`cbpAfterLast`	the custom button must be the last in the set of buttons (i.e: right after Cancel or Help);
`cbpBeforeFirst`	the custom button must be *before* OK button;
`cbpOppositeEdge`	the custom button must be placed to an opposite to other buttons edge (opposite to edge specified in Alignment property, see screenshot below).



If the Align is alHorizontal then custom button will appears from the left of OK button, in case if the Align is alVertical, the custom button will be displayed at the top side of the panel, before the buttons.

See also
Align, Alignment and BtnCustom properties.

10.4.2.13Images

Applies to
acButtonsPanel component.

Declaration
```
property Images: TImageList;
```

Description
The Images property determines which image list (TImageList) is associated with the panel. An image list contains a bitmaps that can be displayed as the glyphs at the left of buttons.

After specifying the Images, you should also specify the *ImageIndex* in the buttons, to display right glyphs.

See also

© 1998-2006, UtilMind Solutions®

BtnOk, BtnCancel, BtnApply, BtnHelp and BtnCustom properties (and their *ImageIndex* sub-property).

10.4.2.14MarginHorz

Applies to
acButtonsPanel component.

Declaration
```
property MarginHorz: Integer;
```

Declaration
The MarginHorz property specifies the spacing between the right side of the panel and the buttons.

See also
MarginVert, ButtonWidth, ButtonHeight and ButtonSpacing properties.

10.4.2.15MarginVert

Applies to
acButtonsPanel component.

Declaration
```
property MarginVert: Integer;
```

Declaration
The MarginVert property specifies the spacing between the top side of the panel and the buttons.

See also
MarginHorz, ButtonWidth, ButtonHeight and ButtonSpacing properties.

10.4.2.16ShowAccelChar

Applies to
acButtonsPanel component.

Declaration
```
property ShowAccelChar: Boolean;
```

Description
The ShowAccelChar property controls whether the buttons should display ampersand character in caption.

Set ShowAccelChar to True to allow the button to display an underlined accelerator key value. When ShowAccelChar is True, any character preceded by an ampersand (&) appears underlined. To display an ampersand when ShowAccelChar is True, use two ampersands (&&) to stand for the single ampersand that is displayed.

Set ShowAccelChar to False to display the caption with all ampersands appearing as ampersands.

See also
ShowDivider and ShowSizeGrip properties.

10.4.2.17ShowDivider

Applies to
acButtonsPanel component.

© 1998-2006, UtilMind Solutions®

Declaration

```
property ShowSizeGrip: Boolean;
```

Description

The ShowSizeGrip property controls whether the panel should display the thin divider line between the buttons and other content of the form.

Set ShowSizeGrip to True to display the divider line, or leave it False if you don't want it.

See also

ShowAccelChar and ShowSizeGrip properties.

10.4.2.18 ShowSizeGrip

Applies to

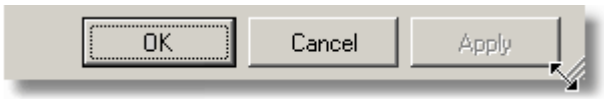
acButtonsPanel component.

Declaration

```
property ShowSizeGrip: Boolean;
```

Description

The ShowSizeGrip property specifies whether the panel should display a resize grip at the bottom-right corner of the panel.



Set ShowSizeGrip to True if you want to make your dialog box resizeable by dragging the bottom-right corner with mouse.

If the most right button covers the resize grip, increase the MarginHorz value, so the buttons will be slightly shifted to the left side of panel.

See also

ShowAccelChar and ShowDivider properties.

10.4.3 Events

10.4.3.1 OnAnyBtnClick

Applies to

acButtonsPanel component.

Declaration

```
property OnAnyBtnClick: TNotifyEvent;
```

Description

The OnAnyBtnClick event occurs when user clicks ANY button on the panel.

See also

OnAnyBtnClick, OnBtnOKClick, OnBtnCancelClick, OnBtnHelpClick, OnBtnApplyClick, OnBtnCustomClick and OnApplySettings events.

© 1998-2006, UtilMind Solutions®

10.4.3.2 OnApplyEnabledChanged

Applies to
acButtonsPanel component.

Declaration
```
property OnApplyEnabledChanged: TNotifyEvent;
```

Description
The OnApplyEnabledChanged event occurs when the Apply button becomes enabled or disabled.

To change the enabled state of Apply button — use ApplyEnabled property.

See also
ApplyEnabled, BtnOk, BtnCancel, BtnApply, BtnHelp and BtnCustom properties;
OnAnyBtnClick, OnBtnOKClick, OnBtnCancelClick, OnBtnHelpClick, OnBtnApplyClick, OnBtnCustomClick and OnApplySettings events.

10.4.3.3 OnApplySettings

Applies to
acButtonsPanel component.

Declaration
```
type
  TacApplySettingsEvent = procedure(Sender: TObject; var Apply: Boolean)
of object;

property OnApplySettings: TacApplySettingsEvent;
```

Description
The OnApplySettings event occurs when user clicks the Apply *or* OK buttons, so the settings of the dialog box should be saved/applied.

Write this event hander to save the changed settings of your dialog box and/or previously validate the dialog controls for correct input. If something incorrect — you can set *Apply* parameter to False, so the Apply button will not become disabled and dialog box will not be closed.

See also
BtnOk, BtnCancel, BtnApply, BtnHelp and BtnCustom properties;
OnAnyBtnClick, OnBtnOKClick, OnBtnCancelClick, OnBtnHelpClick, OnBtnApplyClick, OnBtnCustomClick, OnApplySettings and OnApplyEnabledChanged events.

10.4.3.4 OnBtnApplyClick

Applies to
acButtonsPanel component.

Declaration
```
property OnAnyBtnClick: TNotifyEvent;
```

Description
The OnAnyBtnClick event occurs when user clicks ANY button on the panel.

See also
BtnOk, BtnCancel, BtnApply, BtnHelp and BtnCustom properties;
OnAnyBtnClick, OnBtnOKClick, OnBtnCancelClick, OnBtnHelpClick, OnBtnApplyClick,

© 1998-2006, UtilMind Solutions®

OnBtnCustomClick and OnApplySettings events.

10.4.3.5 OnBtnCancelClick

pplies to
acButtonsPanel component.

Declaration
```
property OnBtnCancelClick: TNotifyEvent;
```

Description
The OnBtnCancelClick event occurs when user clicks the cancel button on the panel.

See also
BtnOk, BtnCancel, BtnApply, BtnHelp and BtnCustom properties;
OnAnyBtnClick, OnBtnOKClick, OnBtnCancelClick, OnBtnHelpClick, OnBtnApplyClick, OnBtnCustomClick and OnApplySettings events.

10.4.3.6 OnBtnCustomClick

Applies to
acButtonsPanel component.

Declaration
```
property OnBtnCustomClick: TNotifyEvent;
```

Description
The OnBtnCustomClick event occurs when user clicks the Custom button on the panel.

See also
BtnOk, BtnCancel, BtnApply, BtnHelp and BtnCustom properties;
OnAnyBtnClick, OnBtnOKClick, OnBtnCancelClick, OnBtnHelpClick, OnBtnApplyClick, OnBtnCustomClick and OnApplySettings events.

10.4.3.7 OnBtnHelpClick

Applies to
acButtonsPanel component.

Declaration
```
property OnBtnHelpClick: TNotifyEvent;
```

Description
The OnBtnHelpClick event occurs when user clicks the Help button on the panel.

See also
BtnOk, BtnCancel, BtnApply, BtnHelp and BtnCustom properties;
OnAnyBtnClick, OnBtnOKClick, OnBtnCancelClick, OnBtnHelpClick, OnBtnApplyClick, OnBtnCustomClick and OnApplySettings events.

10.4.3.8 OnBtnOKClick

Applies to
acButtonsPanel component.

Declaration
```
property OnBtnOKClick: TNotifyEvent;
```

© 1998-2006, UtilMind Solutions®

Description
The OnBtnOKClick event occurs when user clicks the OK button on the panel.

See also
BtnOk, BtnCancel, BtnApply, BtnHelp and BtnCustom properties;
OnAnyBtnClick, OnBtnOKClick, OnBtnCancelClick, OnBtnHelpClick, OnBtnApplyClick, OnBtnCustomClick and OnApplySettings events.

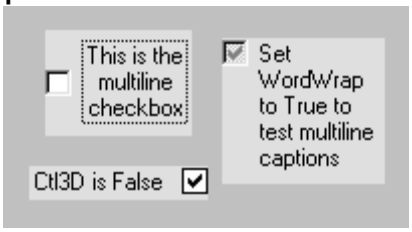
10.5 acCheckBox

10.5.1 TacCheckBox

Overview
The acCheckBox component is the improved TCheckBox, which supports multiline captions and able to have non-3D (flat) border for the checkbox. Contains all needed properties for flexible adjustment of multiline text within client area of control and aligment withing its container.

Since this is just enchancement of standard TCheckBox, you can find out about its basic features in the Delphi/BCB documentation (but you probably already know how to use CheckBoxes. If not — select necessary property in Object Inspector and press F1). This documentation descibes only additional features and improvements.

Snapshot



To specify multiline caption — set WordWrap property to True. To make the flat checkbox — use Ctl3D property.

Additional and improved properties

Align	controls placement of button within its container;
AlignmentHorz	specifies horizontal alignment of text within client area of control;
AlignmentVert	specifies vertical alignment of text within client area of control;
BoxAlignment	controls placement of the checkbox relatively to the caption;
Ctl3D	determines whether a checkbox has a three-dimensional (3-D) or two-dimensional look;
ParentCtl3D	determines whether a checkbox has same 3-D look as its container;
WordWrap	specifies whether the text on checkbox wraps when it is too long for the width of the control. When WordWrap is True, caption can be multiline.

Events

OnMouseEnter	occurs when mouse pointer hovers the checkbox;
OnMouseLeave	occurs when mouse pointer leaves the checkbox.

See also
acRadioButton and acCheckGroupBox components.

© 1998-2006, UtilMind Solutions®

10.5.2 Properties

10.5.2.1 Align

Applies to

acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acIPEdit and acMRUComboBox components *(in generally to ALL successors of TControl)*.

Declaration

```
type
  TAlign = (alNone, alTop, alBottom, alLeft, alRight, alClient);

property Align: TAlign;
```

Declaration

The Align property determines how the control aligns within its container (or parent control).

Use Align to align a control to the top, bottom, left, or right of a form or panel and have it remain there even if the size of the form, panel, or component that contains the control changes.

For example, to use a panel component with various controls on it as a tool palette, change the panel's Align value to alLeft. The value of alLeft for the Align property of the panel guarantees that the tool palette remains on the left side of the form and always equals the client height of the form.

The default value of Align is alNone, which means a control remains where it is positioned on a form or panel. Specify another value to align the control. Use one of these possible values:

Value	Meaning
alNone	The component remains where it was placed on the form or panel. This is the default value.
alTop	The component moves to the top of the form and resizes to fill the width of the form. The height of the component is not affected.
alBottom	The component moves to the bottom of the form and resizes to fill the width of the form. The height of the component is not affected.
alLeft	The component moves to the left side of the form and resizes to fill the height of the form. The width of the component is not affected.
alRight	The component moves to the right side of the form and resizes to fill the height of the form. The width of the component is not affected.
alClient	The component resizes to fill the client area of a form. If a component already occupies part of the client area, the component resizes to fit within the remaining client area.

Note

If Align is set to alClient, the control fills the entire client area so that it is impossible to select the parent form by clicking on it. In this case, select the parent by selecting the control on the form and pressing Esc, or by using the Object Inspector.

Any number of child components within a single parent can have the same Align value, in which case they stack up along the edge of the parent form or panel. The child controls stack up in z-order. To adjust the order in which the controls stack up, drag the controls into their desired positions.

© 1998-2006, UtilMind Solutions®

10.5.2.2 AlignmentHorz

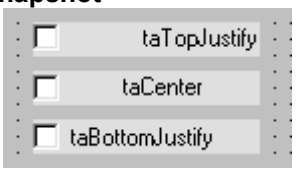
Applies to
acCheckBox and acRadioButton components.

Declaration
```
type
  TAlignment = (taLeftJustify, taRightJustify, taCenter);

property AlignmentHorz: TAlignment;
```

Description
The AlignmentHorz property controls horizontal placement of text string that appears as a label for checkbox control.

By default, text is lined up along the left edge of control (near the check box, if BoxAlignment is `taLeftJustify` as well). To make text lined up along the right edge — set AlignmentHorz to `taRightJustify`. For centering the text horziontally within a client area of control — set AlignmentHorz to `taCenter`.

The effect of the AlignmentHorz property is more obvious if the WordWrap property is True and the caption includes more than one line of text.

Snapshot


See also
AlignmentVert and BoxAlignment properties.

10.5.2.3 AlignmentVert

Applies to
acCheckBox and acRadioButton components.

Declaration
```
type
  TVerticalAlignment = (vaTopJustify, vaBottomJustify, vaCenter);

property AlignmentVert: TVerticalAlignment;
```

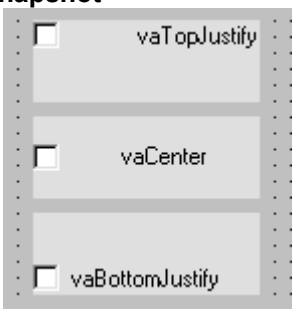
Description
The AlignmentVert property controls vertical placement of text string that appears as a label for checkbox control.

By default, text is lined up along the top edge of control (`AlignmentVert = vaTopJustify`). To align text along the bottom edge — set AlignmentVert to `vaRightJustify`. For centering the text vertically within a client area — set AlignmentVert to `vaCenter`.

The effect of the AlignmentVert property is more obvious if the WordWrap property is True and the caption includes more than one line of text.

© 1998-2006, UtilMind Solutions®

Snapshot



See also

AlignmentHorz property.

10.5.2.4 BoxAlignment

Applies to

acCheckBox, acRadioButton and acCheckGroupBox components.

Declaration

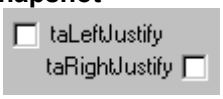
```
type
  TAlignment = (taLeftJustify, taRightJustify, taCenter);
  TLeftRight = taLeftJustify..taRightJustify;

property BoxAlignment: TLeftRight;
```

Description

The BoxAlignment property specifies placement of the check box, relatively to the text on the checkbox control.

By default, checkbox is left-aligned. When the BoxAlignment is `taRightJustify`, the checkbox will be displayed to the right of text.

Snapshot



See also

AlignmentHorz and AlignmentVert properties.

10.5.2.5 Ctl3D

Applies to

acCheckBox, acRadioButton and acCheckGroupBox components *(in generally, to ALL successors of TControl)*.

Declaration

```
property Ctl3D: Boolean;
```

Description

The Ctl3D property determines whether a control has a three-dimensional (3-D) or two-dimensional look.

Use Ctl3D to control the appearance of the control. If Ctl3D is True, the control has a 3-D

© 1998-2006, UtilMind Solutions®

appearance. If Ctl3D is False, the control appears flat. The default value of Ctl3D is True.

If a control's ParentCtl3D property is True, then changes in the Ctl3D property of the control's parent automatically changes the Ctl3D property of the control. When assigning a value directly to a control's Ctl3D property, the control's ParentCtl3D property is automatically set to False.

See also
ParentCtl3D property.

10.5.2.6 ParentCtl3D

Applies to
acCheckBox and acRadioButton components *(in generally, to ALL successors of TControl)*.

Declaration
```
property ParentCtl3D: Boolean;
```

Description
The ParentCtl3D property determines where a radiobutton looks to determine if it should appear three dimensional.

Use ParentCtl3D to ensure that all the components on a form have a uniform appearance. For example, to make all components on a form appear three dimensional, set the form's Ctl3D property to True and each control's ParentCtl3D property to True. Not only will all components have a three-dimensional appearance, but to change to a uniform two-dimensional appearance, you only have to change the Ctl3D property of the form and all the components will become two dimensional.

If ParentCtl3D is True, the component uses the value of its parent component's Ctl3D property. If ParentCtl3D is False, the control uses the value of its own Ctl3D property. The default value is True.

To specify either a 2D or 3D appearance for a particular component, set Ctl3D to True (for 3D) or False (for 2D). When setting the Ctl3D property of a control, ParentCtl3D automatically gets set to False.

See also
Ctl3D property.

10.5.2.7 WordWrap

Applies to
acCheckBox, acRadioButton, acLabel, acURLLabel, acRichLabel, acButton and acBitBtn components.

Declaration
```
property WordWrap: Boolean;
```

Description
The WordWrap property controls whether the text on checkbox wraps when it is too long for the width of the control. When WordWrap is True, caption can be multiline, and single line only when WordWrap is False.

10.5.3 Events

10.5.3.1 OnMouseEnter

Applies to
acLabel, acURLLabel, acRichLabel, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit,

© 1998-2006, UtilMind Solutions®

acNumberEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox, acTreeComboBox, acHeader, acHeaderPanel, acTreeView and acListView components.

Declaration

```
property OnMouseEnter: TNotifyEvent;
```

Description

The OnMouseEnter occurs when mouse pointer hovers the checkbox control.

See also

OnMouseLeave event.

10.5.3.2 OnMouseLeave

Applies to

acLabel, acURLLabel, acRichLabel, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox, acTreeComboBox, acHeader, acHeaderPanel, acTreeView and acListView components.

Declaration

```
property OnMouseLeave: TNotifyEvent;
```

Description

The OnMouseLeave occurs when mouse pointer leaves the checkbox control.

See also

OnMouseEnter event.

10.6 acCheckGroupBox

10.6.1 TacCheckGroupBox

Overview

The acCheckGroupBox component is an easy way to implement to your forms a group box with a checkbox instead of usual caption. When the checkbox is checked, all controls within a group box is enabled. When user uncheck the checkbox, all controls withing a group became automatically disabled.

Snapshots

Additional and improved properties

BoxAlignment	Controls placement of checkbox relatively to the caption. By default, checkbox is left-aligned. When the BoxAlignment is `taRightJustify`, the checkbox will be displayed to the right of text;
Caption	Specifies a text string on the caption of checkbox;
Checked	Indicates whether a checkbox is selected;
Ctl3D	Determines whether a groupbox and checkbox above has a three-dimensional (3-D) or two-dimensional look;
State	Determines the various states a check box (checked, unchecked or grayed).

© 1998-2006, UtilMind Solutions®

New events

OnCheckBoxClick occurs when user clicks the checkbox above the group box, or when the Checked property is changed.

See also

acCheckBox component.

10.6.2 Properties

10.6.2.1 BoxAlignment

Applies to

acCheckGroupBox, acCheckBox and acRadioButton components.

Declaration

```
type
  TAlignment = (taLeftJustify, taRightJustify, taCenter);
  TLeftRight = taLeftJustify..taRightJustify;

property BoxAlignment: TLeftRight;
```

Description

The BoxAlignment property specifies placement of the check box, relatively to the caption of control.

By default, checkbox is left-aligned. When the BoxAlignment is `taRightJustify`, the checkbox will be displayed to the right of text.

Snapshot

10.6.2.2 Caption

Applies to

acCheckGroupBox, acButton, acBitBtn and acRichLabel components *(to all visible controls, in general)*

Declaration

```
property Caption: String;
```

Description

The Caption property specifies a text string that appears as a label for checkbox above the group box..

To underline a character in a Caption that labels a checkbox, include an ampersand (&) before the character. This type of character is called an accelerator character. The user can then select the component by pressing Alt while typing the underlined character. To display an ampersand character in the caption, use two ampersands (&&).

See also

Checked and BoxAlignment properties.

© 1998-2006, UtilMind Solutions®

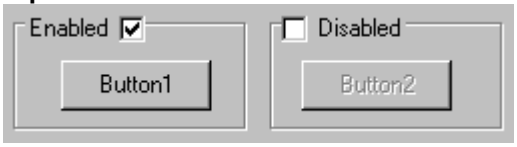
10.6.2.3 Checked

Applies to
acCheckGroupBox component.

Declaration
```
property Checked: Boolean;
```

Description
The Checked property indicates whether the check box on the title of group box of is selected.

When Checked is True, the controls in a group box is enabled. When user *unchecks* the group box (Checked is False), ALL controls within this group box became automatically disabled.

Snapshot


See also
State property and OnCheckBoxClick event.

10.6.2.4 Ctl3D

Applies to
acRadioButton, acCheckBox and acCheckGroupBox components *(in generally, to ALL successors of TControl)*.

Declaration
```
property Ctl3D: Boolean;
```

Description
The Ctl3D property determines whether a control has a three-dimensional (3-D) or two-dimensional look.

Use Ctl3D to control the appearance of the control. If Ctl3D is True, the control has a 3-D appearance. If Ctl3D is False, the control appears flat. The default value of Ctl3D is True.

If a control's ParentCtl3D property is True, then changes in the Ctl3D property of the control's parent automatically changes the Ctl3D property of the control. When assigning a value directly to a control's Ctl3D property, the control's ParentCtl3D property is automatically set to False.

10.6.2.5 State

Applies to
acCheckGroupBox component.

Declaration
```
type
  TCheckBoxState = (cbUnchecked, cbChecked, cbGrayed);

property State: TCheckBoxState;
```

Description
The State property determines the various states a check box control (on top of group box) can

© 1998-2006, UtilMind Solutions®

have. These are the possible values:

Value	Meaning
cbUnchecked	The check box has no check mark indicating the user hasn't selected the option.
cbChecked	The check box has a check mark in it indicating the user has selected the option.
cbGrayed	The check box is gray indicating a third state that is neither checked nor unchecked. Your application determines the meaning of a grayed check box.

See also
Checked property and OnCheckBoxClick event.

10.6.3 Events

10.6.3.1 OnCheckBoxClick

Applies to
acCheckGroupBox component.

Declaration
```
property OnCheckBoxClick: Boolean;
```

Description
The OnCheckBoxClick occurs when the user clicks the checkbox, or when the Checked property is changed. Use the OnCheckBoxClick event handler to write code that responds to the click event.

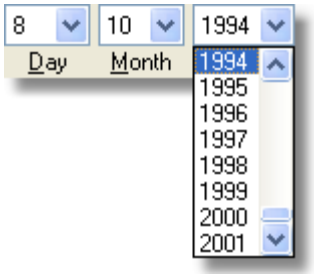
See also
Checked property.

10.7 acComboDatePicker

10.7.1 TacComboDatePicker

Overview
The acComboDatePicker control is the panel with group of 3 combo-boxes used to quickly select the date (day, month and year). You can specify the range of years which can be selected, and selectable values for the Day box will be specified automatically, accordingly to number of days per month and leap year. The control contains the number of properties which specifies its look, i.e: date format, text for labels "attached" to the selection boxes, width of boxes, fonts, etc.

Screenshot



How to use?
Just drop the control on your form and specify the range of years which can be selected to the MinYear and MaxYear properties (MaxYear can be 0, in case if the highest selectable year should be current year).

To determinate selected date — read the Date property, or the Year, Month and Day properties

© 1998-2006, UtilMind Solutions®

separately.
In case if either Year, Month or Day value are not specified — the Date property return **0** (corresponds to 30.12.1899).

If your application should be notified when the date in the combo-box changes — write the OnChange event handler.

Other properties can be used to change the look of the control. For example, you can specify prefered date format, hide the labels "attached" to each combo-box, or change their placement relatively to boxes, and specify the spacing between boxes and labels.

To clear the selection boxes — set Date property to 0, or call the Clear method.

10.7.2 Properties

10.7.2.1 Date

Applies to
acComboDatePicker component.

Declaration
```
property Date: TDateTime;
```

Description
The Date property determines the date selected in the control, in case if the Day, Month and Year specified.

If either Day, Month or Year values are not specified — the Date returns 0 (corresponds to 30.12.1899).

If you would like to clear the combo-boxes — set Date to 0, or call Clear method.

See also
Day, Month and Year properties;
Clear method;
OnChange event.

10.7.2.2 DateFormat

Applies to
acComboDatePicker component.

Declaration
```
type
  TacComboDatePickerFormat = (dpfAuto, dpfDMY, dpfMDY, dpfYMD);

property DateFormat: TacComboDatePickerFormat;
```

Description
The DateFormat property specifies the order of the selection combo-boxes.

Value	Meaning
dpfAuto	the order will be chosen automatically, accordingly to user's regional settings, specified in the Control Panel.
dpfDMY	European format: the day box is first, month — second (from left to right), year —

© 1998-2006, UtilMind Solutions®

third.

dpfMDY	US format: the month box is first, day — second, year — third.
dpfYMD	Chinese format: the year box is first, month — second, day — third.

You can also localize the text in "attached" labels, specifying it to StrDay, StrMonth and StrYear properties of Labels structure.

See also
Labels, Day, Month, Year and Date properties.

10.7.2.3 Day

Applies to
acComboDatePicker component.

Declaration
```
property Day: Word;
```

Description
The Day property determinates the value of the Day box.

Exception can be raised in case if you specify incorrect value.

See also
Day, Month, Year and Date properties;
OnChange event.

10.7.2.4 Labels

Applies to
acComboDatePicker component.

Declaration
```
type
  TacComboDatePickerLabelsPosition = (dlpBelow, dlpAbove);
  TacComboDatePickerLabels = class
  published
    property StrDay: String;
    property StrMonth: String;
    property StrYear: String;
    property Spacing: Integer default 1;
    property Position: TacComboDatePickerLabelsPosition default
dlpBelow;
    property Visible: Boolean default True;
  end;

property Labels: TacComboDatePickerLabels;
```

Description
The Labels structure specifies the properties for the labels, "attached" to the selection combo-boxes.

Visible property controls whether the labels are visible or not, Position — specifies the placement of attached labels (below or above), Spacing — specifies the space in pixels between the combo-box and the labels.

© 1998-2006, UtilMind Solutions®

StrDay, StrMonth and StrYear properties specifies the text for lables. You can change these values if you need to translate the text to another language.

See also
SpacingBetweenCombos, WidthDayMonthBoxes and WidthYearBox properties.

10.7.2.4.1 Position

Applies to
acComboDatePicker component as subproperty of Labels structure.

Declaration
```
type
  TacComboDatePickerLabelsPosition = (dlpBelow, dlpAbove);

property Position: TacComboDatePickerLabelsPosition;
```

Description
The Position property controls the placement of the labels "attached" to day/month/year combo-boxes.

The labels can be below (default), or above the combo-boxes.

If you don't want to display the labels — set Visible to False.

See also
Visible and Spacing properties.

10.7.2.4.2 Spacing

Applies to
acComboDatePicker component as subproperty of Labels structure.

Declaration
```
property Spacing: Integer; // 1 by default
```

Description
The Spacing property controls the vertical spacing, in pixels, between the combo-boxes and their "attached" labels.

If you don't want to display the labels — set Visible to False.

See also
Visible and Position properties.

10.7.2.4.3 StrDay

Applies to
acComboDatePicker component as subproperty of Labels structure.

Declaration
```
property StrDay: String;
```

Description
The StrDay property specifies the text for the label "attached" to the combo-box which selects the Day. You can change this value if you need to translate the text to another language.

© 1998-2006, UtilMind Solutions®

See also
StrMonth and StrYear properties.

10.7.2.4.4 StrMonth

Applies to
acComboDatePicker component as subproperty of Labels structure.

Declaration
```
property StrMonth: String;
```

Description
The StrMonth property specifies the text for the label "attached" to the combo-box which selects the Month. You can change this value if you need to translate the text to another language.

See also
StrDay and StrYear properties.

10.7.2.4.5 StrYear

Applies to
acComboDatePicker component as subproperty of Labels structure.

Declaration
```
property StrYear: String;
```

Description
The StrYear property specifies the text for the label "attached" to the combo-box which selects the Year. You can change this value if you need to translate the text to another language.

See also
StrDay and StrMonth properties.

10.7.2.4.6 Visible

Applies to
acComboDatePicker component as subproperty of Labels structure.

Declaration
```
property Visible: Boolean; // True by default
```

Description
The Visible property controls whether the labels "attached" to selection combo-boxes are visible or not.

Set Visible to False if you don't want to display the labels.

See also
Position and Spacing properties.

10.7.2.5 MaxYear

Applies to
acComboDatePicker component.

Declaration
```
property MaxYear: Word;
```

© 1998-2006, UtilMind Solutions®

Description
The MaxYear property specifies the highest value which can be selected in the Year combo-box.

The MaxYear value can be 0, in case if the maximum selected year should be current year.

See also
Day, Month, Year and Date properties.

10.7.2.6 MinYear

Applies to
acComboDatePicker component.

Declaration
```
property MinYear: Word;
```

Description
The MinYear property specifies the lowest value which can be selected in the Year combo-box.

See also
Day, Month, Year and Date properties.

10.7.2.7 Month

Applies to
acComboDatePicker component.

Declaration
```
property Month: Word;
```

Description
The Month property determinates the value of the Month box.

Exception can be raised in case if you specify incorrect value.

See also
Day, Month, Year and Date properties;
OnChange event.

10.7.2.8 SpacingBetweenCombos

Applies to
acComboDatePicker component.

Declaration
```
property SpacingBetweenCombos: Byte;
```

Description
The SpacingBetweenCombos property specifies the horizontal spacing (in pixels) between the selection combo-boxes.

See also
Labels, WidthDayMonthBoxes and WidthYearBox properties.

© 1998-2006, UtilMind Solutions®

10.7.2.9 WidthDayMonthBoxes

Applies to
acComboDatePicker component.

Declaration
```
property WidthDayMonthBoxes: Byte;
```

Description
The WidthDayMonthBoxes property specifies the width for the Day and Month combo-boxes.

See also
WidthYearBox, SpacingBetweenCombos and Labels properties.

10.7.2.10 WidthYearBox

Applies to
acComboDatePicker component.

Declaration
```
property WidthYearBox: Byte;
```

Description
The WidthDayYearBox property specifies the width for the Year combo-box.

See also
WidthDayMonthBoxes, SpacingBetweenCombos and Labels properties.

10.7.2.11 Year

Applies to
acComboDatePicker component.

Declaration
```
property Year: Word; // 0..9999
```

Description
The Year property determinates the value of the Year box.

You can not specify the value less specified in MinYear property or greater than specified in MaxYear property. The component will automatically adjust specified year to highest or lowest possible value.

See also
Day, Month, Year, Date, MinYear and MaxYear properties;
OnChange event.

10.7.3 Methods

10.7.3.1 Clear

Applies to
acComboDatePicker component.

Declaration
```
procedure Clear;
```

© 1998-2006, UtilMind Solutions®

Description
The Clear method assigns 0 to the Date property, thus clears the selection combo-boxes.

See also
Day, Month, Year and Date properties;
OnChange event.

10.7.4 Events

10.7.4.1 OnChange

Applies to
acComboDatePicker component.

Declaration
```
property OnChange: TNotifyEvent;
```

Description
The OnChange event occurs when the Date (or Day, Month and Year) properties changes.

The OnChange occurs both, when the application changes the Date programmatically, or when user changes the date selecting the value in combo-boxes.

See also
Day, Month, Year and Date properties.

10.8 acControlKeyState

10.8.1 TacControlKeyState

Overview
The acControlKeyState can determinate states of -Caps, -Num, -Scroll, -Control, -Shift keys and change their conditions by blinking the LEDs on the keypad.

acControlKeyState will also watch for changes of the keyboard states and fire some events, indicating that state of some control keys has changes.

Photoshoot

```
NumLock := True;
CapsLock := False;
ScrollLock := True;
```

© 1998-2006, UtilMind Solutions®

10.8.2 Properties

10.8.2.1 CapsLock

Applies to
acControlKeyState component.

Declaration
```
property CapsLock: Boolean;
```

Description
Controls the state of *CapsLock* key on the keyboard. When this property is True, keypad LED indicator is light up.

10.8.2.2 Control

Applies to
acControlKeyState component.

Declaration
```
property Control: Boolean;
```

Description
Controls the state of *Control* key on the keyboard.

10.8.2.3 Insert

Applies to
acControlKeyState component.

Declaration
```
property Insert: Boolean;
```

Description
Controls the state of *Insert* key on the keyboard.

10.8.2.4 NumLock

Applies to
acControlKeyState component.

Declaration
```
property NumLock: Boolean;
```

Description
Controls the state of *Num Lock* key on the keyboard. When this property is True, keypad LED indicator is light up.

10.8.2.5 ScrollLock

Applies to
acControlKeyState component.

Declaration
```
property ScrollLock: Boolean;
```

Description
Controls the state of *Scroll Lock* key on the keyboard. When this property is True, keypad LED

© 1998-2006, UtilMind Solutions®

indicator is light up.

10.8.2.6 Shift

Applies to
acControlKeyState component.

Declaration
```
property Shift: Boolean;
```

Description
Controls the state of *Shift* key on the keyboard.

10.8.3 Events

10.8.3.1 OnCapsLock

Applies to
acControlKeyState component.

Declaration
```
property OnCapsLock: TacControlKeyStateEvent;
```

Description
The OnCapsLock event occurs whenever state of *CapsLock* key has changed.

10.8.3.2 OnControl

Applies to
acControlKeyState component.

Declaration
```
property OnControl: TacControlKeyStateEvent;
```

Description
The OnCapsLock event occurs whenever state of *Control* key has changed.

10.8.3.3 OnInsert

Applies to
acControlKeyState component.

Declaration
```
property OnInsert: TacControlKeyStateEvent;
```

Description
The OnCapsLock event occurs whenever state of *Insert* key has changed.

10.8.3.4 OnNumLock

Applies to
acControlKeyState component.

Declaration
```
property OnNumLock: TacControlKeyStateEvent;
```

Description
The OnCapsLock event occurs whenever state of *Num Lock* key has changed.

© 1998-2006, UtilMind Solutions®

10.8.3.5 OnScrollLock

Applies to
acControlKeyState component.

Declaration
```
property OnScrollLock: TacControlKeyStateEvent;
```

Description
The OnCapsLock event occurs whenever state of *Scroll Lock* key has changed.

10.8.3.6 OnShift

Applies to
acControlKeyState component.

Declaration
```
property OnShift: TacControlKeyStateEvent;
```

Description
The OnShift event occurs whenever state of *Shift* key has changed.

10.9 acCurrencyEdit

10.9.1 TacCurrencyEdit

Overview
The TacCurrencyEdit is the edit box the edit box used to easily edit currency values with 2 decimal digits (both positive and negative). Returns typed value as Value property.

10.9.2 Properties

10.9.2.1 Value

Enter topic text here.

10.9.3 Events

10.9.3.1 OnChange

Enter topic text here.

10.10 acDateTimeCalc

10.10.1 TacDateTimeCalc

Overview
The acDateTimeCalc component calculates number of years, months, days, hours, munutes, seconds and milliseconds between two dates specified by BeginTime and EndTime properties.

How to use?
Just specify the BeginTime and EndTime properties and read number of years, months, days, hours, minutes, seconds and milliseconds from the according public properties.

Example
```
procedure TForm1.DateTimePicker1Change(Sender: TObject);
begin
  DTCalc.BeginTime := DateTimePicker1.Date + DateTimePicker2.Time;
```

© 1998-2006, UtilMind Solutions®

```
  DTCalc.EndTime := DateTimePicker3.Date + DateTimePicker4.Time;

  LYears.Caption := 'Years:' + IntToStr(DTCalc.Years);
  LMonths.Caption := 'Months:' + IntToStr(DTCalc.Months);
  LDays.Caption := 'Days:' + IntToStr(DTCalc.Days);
  LHours.Caption := 'Hours:' + IntToStr(DTCalc.Hours);
  LMinutes.Caption := 'Minutes:' + IntToStr(DTCalc.Minutes);
  LSeconds.Caption := 'Seconds:' + IntToStr(DTCalc.Seconds);
  LMSeconds.Caption := 'MSeconds:' + IntToStr(DTCalc.MSeconds);
end;
```

10.10.2 Properties

10.10.2.1BeginTime

Applies to
acDateTimeCalc component.

Declaration
```
property BeginTime: TDateTime;
```

Description
The BeginTime property controls the beginning date of period specified by BeginTime and EndTime properties.

After specifying the BeginTime or EndTime properies the acDateTimeCalc component automatically calculates number of years, months, days, hours, seconds and milliseconds between these two dates.

See also
EndTime property.

10.10.2.2EndTime

Applies to
acDateTimeCalc component.

Declaration
```
property EndTime: TDateTime;
```

Description
The EndTime property controls the beginning date of period specified by BeginTime and EndTime properties.

After specifying the BeginTime or EndTime properies the acDateTimeCalc component automatically calculates number of years, months, days, hours, seconds and milliseconds between these two dates.

See also
BeginTime property.

10.10.2.3Years

Applies to
acDateTimeCalc component.

© 1998-2006, UtilMind Solutions®

Declaration

```
property Years: LongInt; // public and read-only
```

Description

The Years property determines the number of years between two dates specified by BeginTime and EndTime properties.

See also

Months, Days, Hours, Minutes, Seconds, MSeconds properties.

10.10.2.4Months

Applies to

acDateTimeCalc component.

Declaration

```
property Months: LongInt; // public and read-only
```

Description

The Months property determines the number of months between two dates specified by BeginTime and EndTime properties.

See also

Years, Days, Hours, Minutes, Seconds, MSeconds properties.

10.10.2.5Days

Applies to

acDateTimeCalc component.

Declaration

```
property Days: LongInt; // public and read-only
```

Description

The Days property determines the number of days between two dates specified by BeginTime and EndTime properties.

See also

Years, Months, Hours, Minutes, Seconds, MSeconds properties.

10.10.2.6Hours

Applies to

acDateTimeCalc component.

Declaration

```
property Hours: LongInt; // public and read-only
```

Description

The Hours property determines the number of hours between two dates specified by BeginTime and EndTime properties.

See also

Years, Months, Days, Minutes, Seconds, MSeconds properties.

© 1998-2006, UtilMind Solutions®

10.10.2.7Minutes

Applies to
acDateTimeCalc component.

Declaration
```
property Minutes: LongInt; // public and read-only
```

Description
The Minutes property determines the number of minutes between two dates specified by BeginTime and EndTime properties.

See also
Years, Months, Days, Hours, Seconds, MSeconds properties.

10.10.2.8Seconds

Applies to
acDateTimeCalc component.

Declaration
```
property Seconds: LongInt; // public and read-only
```

Description
The Seconds property determines the number of seconds between two dates specified by BeginTime and EndTime properties.

See also
Years, Months, Days, Hours, Minutes, MSeconds properties.

10.10.2.9MSeconds

Applies to
acDateTimeCalc component.

Declaration
```
property MSeconds: LongInt; // public and read-only
```

Description
The MSeconds property determines the number of milliseconds between two dates specified by BeginTime and EndTime properties.

See also
Years, Months, Days, Hours, Minutes, Seconds properties.

10.11 acDBListView

10.11.1 TacDBListView

Overview
The acDBListView is the database-aware version of acListView component. Were added just one property — DataSource. When you select the active data source which points to some TQuery component, the acDBListView will be filled with fields / records, accordingly to the SQL query. However the acDBListView can be used just for displaying database records, not for editing.

How to use?

© 1998-2006, UtilMind Solutions®

Just select the TQuery or any its third-party replacement in the DataSource property and the ListView will be filled with values accordingly to the SQL statement specified in the TQuery. Then just add required number of columns using property editor of Column property.

For example, you have some table. Let's suppose, "users", which contains some personal information about your customers. There is two fields with usernames and passwords ("username" and "password") and you wish to show these fields using acDBListView.

Then just specify following SQL statement to the TQuery:

```
SELECT username,password FROM users
```

and set Active property of TQuery to True.

Or another advanced example: you wish to change order of columns, add some new field (i.e: email address, "email" field) + only select users which has subscribed ("subscribe" field) to some mailing list.
Specify following SQL statement to TQuery:

```
SELECT email,password,username FROM users WHERE subscribe="Y"
```

Then activate TQuery and don't foget to set the columns accordingly to selected fields.

See also
acListView component.

10.11.2 Properties

10.11.2.1DataSource

Applies to
acDBListView component.

Declaration

```
property DataSource: TDataSource; // see 'Data Access' tab
```

Description
The DataSource property determines where the component obtains the data to display. Specify the data source component that identifies the dataset the data is found in.

10.12 acDesktopWallpaper

10.12.1 TacDesktopWallpaper

Overview
The acDesktopWallpaper component will implement to your application all needed functionality to make operations with the Desktop Wallpapers neat and easy.

How to use?
To deteminate or specify a path to current desktop wallpaper — read or change Wallpaper property. To specify the wallpaper's appearance — use Stretch and Tile properties.

To change the desktop wallpaper — just point the Wallpaper property to another .BMP file or call SetDesktopWallpaper method. To make wallpaper tiled on desktop- set Tile property to True (this is great feature for small wallpaper images). To make wallpaper stretched on entire desktop — set Stretch property to True and Tile to False.

If you wish to specify the wallpaper appearance automatically — set AutoFormat property to True and all pictures with size less than a quarter of screen will be "tiled", big pictures will be "stretched". Note: When AutoFormat = True, values of Stretch and Tile property will be ignored.

© 1998-2006, UtilMind Solutions®

To clear wallpaper from desktop — call RemoveWallpaper method.

Demo application
The acDesktopWallpaper component are successfully used in our Wallpaper Manager. You may freely download and test this software (or just take a look on it's screenshots), go to http://www.appcontrols.com/software/wallpaperman.html

10.12.2 Properties

10.12.2.1AutoFormat

Applies to
acDesktopWallpaper component.

Declaration
```
property AutoFormat: Boolean;
```

Description
The AutoFormat property allows to automatically specify the best appearance for picture used as wallpaper.

Set AutoFormat to True, and acDesktopWallpaper will automatically choose the best appearance when you specify the picture using Wallpaper property. All pictures with size less than a quarter of screen will be "tiled", big images will be automatically "stretched".
Note: when AutoFormat = True, values of Tile and Stretch properties will be ignored.

Set AutoFormat to False if you wish to specify appearance programatically using Stretch and Tile properties.

See also
Wallpaper, Stretch and Tile properties.

10.12.2.2Stretch

Applies to
acDesktopWallpaper component.

Declaration
```
property Stretch: Boolean;
```

Description
The Stretch property controls the appearance of wallpapers on Desktop. If Stretch property is True, wallpaper will stretched on whole screen.

See also
Tile and AutoFormat properties.

10.12.2.3Tile

Applies to
acDesktopWallpaper component.

Declaration
```
property Tile: Boolean;
```

Description
The Tile property controls the appearance of wallpapers on Desktop. If Tile property is True,

© 1998-2006, UtilMind Solutions®

wallpaper will "tiled" on whole screen.

See also
Stretch and AutoFormat properties.

10.12.2.4Wallpaper

Applies to
acDesktopWallpaper component.

Declaration
```
property Wallpaper: String; // filename, points to .BMP file
```

Description
The Wallpaper property is the file name of the bitmap image. To change the desktop wallpaper - point this property to another .BMP file.

Alternatively you can set new wallpaper and its appearance calling SetDesktopWallpaper method.

See also
Stretch, Tile and AutoFormat properties;
SetDesktopWallpaper method.

10.12.3 Methods

10.12.3.1SetDesktopWallpaper

Applies to
acDesktopWallpaper component.

Declaration
```
procedure SetDesktopWallpaper(WallpaperFile: String;
                              Stretch, Tile: Boolean);
```

Description
Use SetDesktopWallpaper method to imedaitely specify the picture file which will be used as wallpaper (*WallpaperFile* parameter) and its appearance on desktop using *Stretch* and *Tile* parameters.

Alternatively you can set the desktop wallpaper pointing the Wallpaper property to another .BMP file and specifying appearance in Stretch and Tile properties.

See also
Wallpaper, Stretch, Tile and AutoFormat properties;
RemoveWallpaper method.

10.12.3.2RemoveWallpaper

Applies to
acDesktopWallpaper component.

Declaration
```
procedure RemoveWallpaper;
```

Description
Call RemoveWallpaper to clear the wallpaper from desktop.

© 1998-2006, UtilMind Solutions®

See also
SetDesktopWallpaper method.

10.13 acDragPanel

10.13.1 TacDragPanel

Overview
The acDragPanel component is the panel which lets user to move and dock the owner form dragging the panel that covers this form; or simply move the panel within its container. Works either with normal or MDI forms. The DragPanel is great addition to acAppBar.

How to use?
Just drop the acDragPanel on your form and specify the DragObject. Then run the project and try to move the form dragging the panel.

Additional / improved properties

DragObject	specifies which object should be moved when user dragging the panel with mouse. User can drag the owner form by panel, or the panel itself within its container;
DragCursor	specifies the cursor image when user dragging the form by panel.

10.13.2 Properties

10.13.2.1 DragCursor

Applies to
acDragPanel component.

Declaration
```
property DragCursor: TCursor;
```

Description
The DragCursor property specifies the image for mouse cursor when user dragging the form by panel.

See also
DragObject property.

10.13.2.2 DragObject

Applies to
acDragPanel component.

Declaration
```
type
  TacDragPanelDragObject = (doForm, doPanel, doNone);

property DragObject: TacDragPanelDragObject;
```

Description
The DragObject property specifies which object should be moved when user dragging the panel with mouse. There are possible values for DragObject:

Value	Meaning
doForm	the owner *form* will be moved, when user dragging the panel with mouse;
doPanel	the *panel* will be moved itself within its container;

© 1998-2006, UtilMind Solutions®

doNone *nothing happends*. The panel behave like usual TPanel component.

See also
DragCursor property.

10.14 acEdit

10.14.1 TacEdit

Overview
The acEdit component is the enchanced TEdit control. The acEdit can align the text by center or right edge control, able to have custom button with bitmap image, can automatically validate entered characters and disallow to type some unwanted letters. Also it contains the AttachedLabel structure which lets to operate with the label associated with the edit control and specify its behavior and position near the control.

Additionally, the acEdit can automatically change the background color on disabled state (see ColorDisabled property) and supports custom cursor images for non-client area of edit control (see CursorBorder property).

Additional and improved properties

Align	controls placement of edit control within its container or parent control;
Alignment	controls horizontal placement of text within the edit control;
AttachedLabel	controls the behavior of the label attached to the control and its position;
AutoSelect	whether the acEdit should automatically select the line when user moves to the edit control pressing the *Tab* key;
Button	settings for custom button at the right edge of edit control;
ColorDisabled	background color of edit control at disabled state (when Enabled = False);
CursorBorder	cursor image when mouse pointer passes the border of edit control;
ValidateChars	whether the acEdit component should validate entered characters and disallow symbols which not described in ValidChars property.
ValidChars	specifies characters allowed for input when ValidateChars is True.

New events

OnButtonClick	occurs when user clicks the button at the right side of control;
OnLabelClick	occurs when user clicks the label attached to the control;
OnLabelDblClick	occurs when user double clicks the label attached to the control;
OnMouseEnter	occurs when mouse pointer passes over edit control;
OnMouseLeave	occurs when mouse pointer leaves the edit control.

See also
acNumberEdit and acIPEdit components.

10.14.2 Properties

10.14.2.1Align

Applies to
acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acIPEdit and

© 1998-2006, UtilMind Solutions®

acMRUComboBox components *(in generally to ALL successors of TControl)*.

Declaration

```
type
  TAlign = (alNone, alTop, alBottom, alLeft, alRight, alClient);

property Align: TAlign;
```

Declaration

The Align property determines how the control aligns within its container (or parent control).

Use Align to align a control to the top, bottom, left, or right of a form or panel and have it remain there even if the size of the form, panel, or component that contains the control changes.

For example, to use a panel component with various controls on it as a tool palette, change the panel's Align value to alLeft. The value of alLeft for the Align property of the panel guarantees that the tool palette remains on the left side of the form and always equals the client height of the form.

The default value of Align is alNone, which means a control remains where it is positioned on a form or panel. Specify another value to align the control. Use one of these possible values:

Value	Meaning
alNone	The component remains where it was placed on the form or panel. This is the default value.
alTop	The component moves to the top of the form and resizes to fill the width of the form. The height of the component is not affected.
alBottom	The component moves to the bottom of the form and resizes to fill the width of the form. The height of the component is not affected.
alLeft	The component moves to the left side of the form and resizes to fill the height of the form. The width of the component is not affected.
alRight	The component moves to the right side of the form and resizes to fill the height of the form. The width of the component is not affected.
alClient	The component resizes to fill the client area of a form. If a component already occupies part of the client area, the component resizes to fit within the remaining client area.

Note

If Align is set to alClient, the control fills the entire client area so that it is impossible to select the parent form by clicking on it. In this case, select the parent by selecting the control on the form and pressing Esc, or by using the Object Inspector.

Any number of child components within a single parent can have the same Align value, in which case they stack up along the edge of the parent form or panel. The child controls stack up in z-order. To adjust the order in which the controls stack up, drag the controls into their desired positions.

10.14.2.2Alignment

Applies to

acEdit, acNumberEdit, acIPEdit and acTreeComboBox components.

Declaration

```
type
  TAlignment = (taLeftJustify, taRightJustify, taCenter);
```

© 1998-2006, UtilMind Solutions®

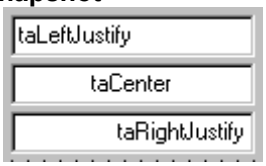
```
property Alignment: TAlignment;
```

Description

The Alignment property controls horizontal placement of the text within the edit control. Set Alignment to specify how the text of the edit control is aligned within the client area (ClientRect property) of this control. Alignment can be one of the following values:

Value	Meaning
taLeftJustify	The text appears lined up along the left edge of the edit control;
taRightJustify	The text appears lined up along the right edge of the edit control;
taCenter	The text is horizontally centered in the edit control.

Snapshot



10.14.2.3AttachedLabel

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and acTreeComboBox components.

Unit

acAttachedLabel

Declaration

```
type
  TacAttachedLabelPosition = (lpAbove, lpBelow, lpLeft, lpRight);
  TacAttachedLabelSpacingKind = (skNearPoint, skFarPoint);
  TacAttachedLabel = class
    property Caption: TCaption;
    property Color: TColor default clBtnFace;
    property Cursor: TCursor default crDefault;
    property Enabled: Boolean;
    property FocusOnClick: Boolean default True;
    property Font: TFont;
    property Hint: String;
    property ParentColor: Boolean default True;
    property ParentFont: Boolean default True;
    property ParentShowHint: Boolean default True;
    property PopupMenu: TPopupMenu;
    property Position: TacAttachedLabelPosition default lpLeft;
    property ShowAccelChar: Boolean default True;
    property ShowHint: Boolean default False;
    property Spacing: Integer default 4;
    property SpacingKind: TacAttachedLabelSpacingKind default
skNearPoint;
    property Transparent: Boolean default False;
    property Visible: Boolean;
    property WordWrap: Boolean default False;
  end;
```

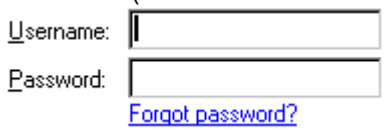
Description

© 1998-2006, UtilMind Solutions®

The AttachedLabel structure lets you to operate with the label attached to the some side of control (at the left or right side, above or below). You can specify the behaviour of label, its position near the control, and accelerator key for the control (character in caption, after ampersand (&)).

The label always associated with the control. When user press the accelerator key (Alt + Character, specified after ampersand (&)), the input focus will be imediately switched to the labeled control. The input focus also will be moved to the control if user clicks the label with mouse.

Screenshot (the labels at the left side is attached to the edit control):



See also

OnLabelClick and OnLabelDblClick events.

10.14.2.3.1 Caption

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property Caption: TCaption;
```

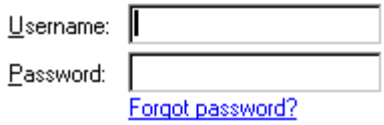
Description

The Caption property specified a text string which appers on label attached to control (attached label).

To underline a character in a Caption that labels a component, include an ampersand (&) before the character. This type of character is called an accelerator character. The user can then select the component by pressing Alt while typing the underlined character. To display an ampersand character in the caption, use two ampersands (&&).

The label always associated with the control. When user press the accelerator key (Alt + Character, specified after ampersand (&)), the input focus will be imediately switched to the labeled control.

Screenshot (the labels at the left side is attached to the edit control):



See also

ShowAccelChar, Color, Font, Transparent, Visible and WordWrap properties.

10.14.2.3.2 Color

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration

© 1998-2006, UtilMind Solutions®

```
property Color: TColor;
```

Description

The Color property specifies the background color for label attached to control. Use Color to read or change the background color of the control.

If a control's ParentColor property is true, then changing the Color property of the control's parent automatically changes the Color property of the control. When the value of the Color property is changed, the control's ParentColor property is automatically set to False.

If you don't want to show background — set Transparent property to True.

See also

ParentColor and Transparent properties.

10.14.2.3.3 Cursor

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property Cursor: TCursor;
```

Description

The Cursor property specifies the image used to represent the mouse pointer when it passes into the region covered by this label attached to control.

Change the value of Cursor to provide feedback to the user when the mouse pointer enters the control. The value of Cursor is the index of the cursor in the list of cursors maintained by the global variable, Screen. In addition to the built-in cursors provided by TScreen, applications can add custom cursors to the list.

See also

Additional Cursors

10.14.2.3.4 Enabled

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property Enabled: Boolean;
```

Description

The Enabled property controls whether the control responds to mouse, keyboard, and timer events.

Use Enabled to change the availability of the control to the user. To disable a control, set Enabled to false. Disabled controls appear dimmed. If Enabled is false, the control ignores mouse, keyboard, and timer events.

To re-enable a control, set Enabled to true. The control is no longer dimmed, and the user can use the control.

See also

© 1998-2006, UtilMind Solutions®

Caption and Color properties.

10.14.2.3.5 FocusOnClick

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property FocusOnClick: Boolean; // True by default
```

Description
The FocusOnClick property controls whether the control, associated with the attached label, should accept focus when user clicks the label with mouse.

Set FocusOnClick to True if you want to switch focus to the control when user clicks the label, or set it to False otherwise.

Note
Additionally you can use OnLabelClick and OnLabelDblClick events of the control.

10.14.2.3.6 Font

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property Font: TFont;
```

Description
The Font property controls the attributes of text written on the attached label of control.

To change to a new font, specify a new TFont object. To modify a font, change the value of the Charset, Color, Height, Name, Pitch, Size, or Style of the TFont object.

See also
ParentFont property.

10.14.2.3.7 Hint

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property Hint: String;
```

Description
The Hint property contains the text string that can appear when the user moves the mouse over the label attached control.

If you are using the FormHelp component to show the context-sensitive help taken from secondary part of hint AND the Hint for attached label is not specified, the FormHelp will use the hint of main control.

© 1998-2006, UtilMind Solutions®

See also
ShowHint and ParentShowHint properties;
acFormHelp and acFormHints components.

10.14.2.3.8 ParentColor

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property ParentColor: Boolean;
```

Description
The ParentColor determines where an attached label looks for its color information.

To have a control use the same color as its parent control, set ParentColor to true If ParentColor is false the control uses its own Color property.

Set ParentColor to true for all controls in order to ensure that all the controls on a form have a uniform appearance. For example, if ParentColor is true for all controls in a form, changing the background color of the form to gray causes all the controls on the form to also have a gray background.

When the value of a label's Color property changes, ParentColor becomes false automatically.

See also
Color and Transparent properties.

10.14.2.3.9 ParentFont

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property ParentFont: Boolean;
```

Description
The ParentFont property determines where a control looks for its font information.

To have a control use the same font as its parent control, set ParentFont to true If ParentFont is false the label uses its own Font property.

Set ParentFont to true for all controls in order to ensure that all the controls on a form have a uniform appearance. For example, if ParentFont is true for all controls in a form, changing the form's Font property to 12-point Courier causes all controls on the form to use that font.

When the value of a control's Font property changes, ParentFont becomes false automatically.

When ParentFont is true for a form, the form uses the default font.

See also
Font property.

© 1998-2006, UtilMind Solutions®

10.14.2.3.10 ParentShowHint

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property ParentShowHint: Boolean;
```

Description
The ParentShowHint property determines where an attached label looks to find out if its Help Hint should be shown.

Use ParentShowHint to ensure that all the controls on a form either uniformly show their Help Hints or uniformly do not show them.

If ParentShowHint is true, the label uses the ShowHint property value of its parent. If ParentShowHint is false, the label uses the value of its own ShowHint property.

To provide Help Hints for only selected controls on a form, set the ShowHint property for those controls that should have Help Hints to true, and ParentShowHint becomes false automatically.

Note
Enable or disable all Help Hints for the entire application using the ShowHint property of the application object.

See also
ShowHint and Hint properties;
acFormHelp and acFormHints components.

10.14.2.3.11 PopupMenu

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property PopupMenu: TPopupMenu;
```

Description
The PopupMenu property Identifies the pop-up menu associated with the label attached to control.

Assign a value to PopupMenu to make a pop-up menu appear when the user selects the label and clicks the right mouse button. If the TPopupMenu's AutoPopup property is True, the pop-up menu appears automatically. If the menu's AutoPopup property is False, display the menu with a call to its Popup method from the control's OnContextPopup event handler.

10.14.2.3.12 Position

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
type
  TacAttachedLabelPosition = (lpAbove, lpBelow, lpLeft, lpRight);
```

© 1998-2006, UtilMind Solutions®

```
property Position: TacAttachedLabelPosition;
```

Description

The Position property of the attached label specifies the layout of the label near the control.

You can make the label visible above, below, at the left or the right side of control.

See also

Spacing and SpacingKind properties.

10.14.2.3.13 ShowAccelChar

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property ShowAccelChar: Boolean;
```

Description

The ShowAccelChar property determines how an ampersand in the label text is displayed.

Set ShowAccelChar to True to allow the label to display an underlined accelerator key value. When ShowAccelChar is True, any character preceded by an ampersand (&) appears underlined. To display an ampersand when ShowAccelChar is True, use two ampersands (&&) to stand for the single ampersand that is displayed.

Set ShowAccelChar to False to display the label text with all ampersands appearing as ampersands.

Example

This example uses two labels on a form. The first label has a caption with an accelerator character in it. The second label also includes an ampersand, but it does not appear as an accelerator character.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Label1.ShowAccelChar := True;
  Label1.Caption := 'An &Underlined character appears here';
  Label2.ShowAccelChar := False;
  Label2.Caption := 'An ampersand (&) appears here';
end;
```

See also

Caption property.

10.14.2.3.14 ShowHint

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property ShowHint: Boolean;
```

Description

© 1998-2006, UtilMind Solutions®

The ShowHint property determines whether the label attached to control displays a Help Hint when the mouse pointer rests momentarily on the label.

The Help Hint is the value of the Hint property, which is displayed in a box just beneath the control. Use ShowHint to determine whether a Help Hint appears for the control.

To enable Help Hint for a particular control, the application ShowHint property must be true and either the control's own ShowHint property must be true, or the control's ParentShowHint property must be true and its parent's ShowHint property must be true.

For example, imagine a check box within a group box. If the ShowHint property of the group box is true and the ParentShowHint property of the check box is true, but the ShowHint property of the check box is false, the check box still displays its Help Hint.

Changing the ShowHint value automatically sets the ParentShowHint property to false.

See also
Hint and ParentShowHint properties.

10.14.2.3.15 Spacing

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

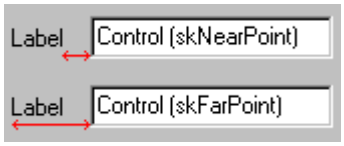
Declaration
```
property Spacing: Integer;
```

Description
The Spacing property determines the space (in pixels) between the control and attached label.

There is two spacing kinds which have an influence on distance between the control and label (see SpacingKind property).

When SpacingKind is `skNearPoint`, the Spacing property signifies the distance between control and nearest point of attached label.

When SpacingKind is `skFarPoint`, the Spacing property signigies the distance between control and far point of label.

Screenshot


See also
SpacingKind and Position properties.

10.14.2.3.16 SpacingKind

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration

© 1998-2006, UtilMind Solutions®

```
type
  TacAttachedLabelSpacingKind = (skNearPoint, skFarPoint);

property SpacingKind: TacAttachedLabelSpacingKind;
```

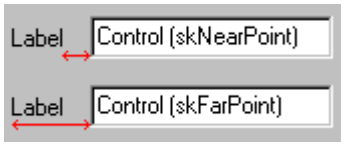
Description

The SpacingKind property specifies how to measure the spacing between the control and attached label.

There is two possible values for SpacingKind property:

Value	Meaning
skNearPoint	the Spacing property signifies the distance between control and nearest point of attached label;
skFarPoint	the Spacing property signigies the distance between control and far point of label.

Screenshot



See also

Spacing and Position properties.

10.14.2.3.17 Transparent

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property Transparent: Boolean;
```

Description

The Transparent property specifies whether controls that sit below the label on a form can be seen through the label.

Set Transparent to True to prevent the label from obscuring other controls on the form. For example, if the label is used to add text to a graphic, set Transparent to True so that the label does not stand out as a separate object.

Writing text so that the background is transparent is slower than writing text when Transparent is False. If the label is not obscuring a complicated image, performance can be improved by setting the background color of the label to match the object beneath it and setting Transparent to False.

See also

Color property.

10.14.2.3.18 Visible

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration

© 1998-2006, UtilMind Solutions®

```
property Visible: Boolean;
```

Description
The Visible property determines whether the label attached to the control appears on screen.

Use the Visible property to control the visibility of the control at runtime. If Visible is true, the control appears. If Visible is false, the control is not visible.

10.14.2.3.19 WordWrap

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property WordWrap: Boolean;
```

Description
The WordWrap property specifies whether the label text wraps when it is too long for the width of the label.

Set WordWrap to True to allow the label to display multiple line of text. When WordWrap is True, text that is too wide for the label control wraps at the right margin and continues in additional lines.

Set WordWrap to False to limit the label to a single line. When WordWrap is False, text that is too wide for the label appears truncated.

See also
Caption property.

10.14.2.4AutoSelect

Applies to
acEdit, acNumberEdit, acIPEdit and acTreeComboBox components.

Declaration
```
property AutoSelect: Boolean; // True by default
```

Description
The AutoSelect property controls whether the text in the edit control should be automatically selected when user moves to the control pressing *Tab* key, activating input focus of the edit control.

10.14.2.5Button

Applies to
acEdit, acNumberEdit and acTreeComboBox components.

Declaration
```
type
  TacEditButton = class(TPersistent)
  published
    property Cursor: TCursor;
    property Flat: Boolean; // in Delphi 3 and later !!!
    property Glyph: TBitmap;
    property NumGlyphs: TNumGlyphs; // 1..4
    property Hint: String;
    property Kind: TacEditButtonKind;
```

© 1998-2006, UtilMind Solutions®

```
    property Width: Word;
    property Visible: Boolean; // False by default
  end;
```

Description

The acEdit component (and all its successors, like acNumberEdit or acTreeComboBox) is able to have a custom button at the right edge of edit control. The Button structure intended for specify settings for this button. To make button visible on edit control — set Button.Visible to True.

You can specify the glyph image and number of images, hint, cursor and width for button.

The button which at the right of edit control is just usual TSpeedButton component. You can even gain direct access to all button's properties using hidden **EditButton** property.
For example, to specify caption for this button you may use following code:

```
acEdit1.EditButton.Caption := 'Test';
```

See also

OnButtonClick event.

10.14.2.5.1 Cursor

Applies to

acEdit, acNumberEdit and acTreeComboBox components as subproperty of Button structure.

Declaration

```
property Cursor: TCursor;
```

Description

The Cursor property specifies the image for mouse pointer when it passes over the custom button at the right side of edit control.

See also

CursorBorder property.

10.14.2.5.2 Flat

Applies to

acEdit, acNumberEdit and acTreeComboBox components as subproperty of Button structure.

Declaration

```
property Flat: Boolean;
```

Description

The Flat property determines whether the button has a 3D border that provides a raised or lowered look.

Set Flat to True to remove the raised border when the button is unselected and the lowered border when the button is clicked or selected. When Flat is True, use separate bitmaps for the different button states to provide visual feedback to the user about the button state.

Remarks

This property available in Delphi 3 and later.

10.14.2.5.3 Glyph

Applies to

acEdit, acNumberEdit and acTreeComboBox components as subproperty of Button structure.

© 1998-2006, UtilMind Solutions®

Declaration

```
property Glyph: TBitmap;
```

Description

The Glyph property specifies the bitmap that appears on the custom button at the right side of edit control.

Set Glyph to a bitmap object that contains the image that should appear on the face of the button. Bring up the Open dialog box from the Object Inspector to choose a bitmap file (with a .BMP extension), or specify a bitmap file at runtime.

Glyph can provide up to four images within a single bitmap. All images must be the same size and next to each other in a horizontal row. Button displays one of these images depending on the state of the button.

Image position	Buttons state	Description
First	Up	This image appears when the button is unselected. If no other images exist in the bitmap, this image is used for all states.
Second	Disabled	This image usually appears dimmed to indicate that the button can't be selected.
Third	Clicked	This image appears when the button is clicked. The Up image reappears when the user releases the mouse button.
Fourth	Down	This image appears when the button stays down indicating that it remains selected.

If only one image is present, Button attempts to represent the other states by altering the image slightly for each state, although the Down state is always the same as the Up state.

If the bitmap contains multiple images, specify the number of images in the bitmap with the NumGlyphs property.

Note

The lower left hand pixel of the bitmap is reserved for the "transparent" color. Any pixel in the bitmap which matches that lower left hand pixel will be transparent.

See also

NumGlyphs and Kind properties.

10.14.2.5.4 NumGlyphs

Applies to

acEdit, acNumberEdit and acTreeComboBox components as subproperty of Button structure.

Declaration

```
property NumGlyphs: TNumGlyphs; // 1..4
```

Description

The NumGlyphs is the number of images specified in the Glyph property.

Set NumGlyphs to the number of images provided by the bitmap assigned to the Glyph property. All images must be the same size and next to each other in a row. The Glyph property can provide up to four images.

See also

© 1998-2006, UtilMind Solutions®

Glyph property.

10.14.2.5.5 Hint

Applies to
acEdit, acNumberEdit and acTreeComboBox components as subproperty of Button structure.

Declaration
```
property Hint: String;
```

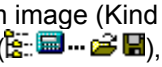
Description
The Hint property contains the text string that can appear when the user moves the mouse pointer over the button at right side of edit control.

10.14.2.5.6 Kind

Applies to
acEdit, acNumberEdit and acTreeComboBox components as subproperty of Button structure.

Declaration
```
type
  TacEditButtonKind = (bkCustom, bkEllipsis, bkDropDown, bkFileOpen,
bkFileSave, bkBrowse, bkCalc);

property Kind: TacEditButtonKind;
```

Description
The Kind property determines the kind of bitmap button.

You can use custom image (Kind = bkCustom) and specify your sign to Glyph property, or use one of following images (), just switching the Kind property.

See also
Glyph and NumGlyphs properties.

10.14.2.5.7 Width

Applies to
acEdit, acNumberEdit and acTreeComboBox components as subproperty of Button structure.

Declaration
```
property Width: Integer;
```

Description
The Width property controls the horizontal size of the button in pixels.

The button's height controlled by height of acEdit control.

10.14.2.5.8 Visible

Applies to
acEdit, acNumberEdit and acTreeComboBox components as subproperty of Button structure.

Declaration
```
property Visible: Boolean;
```

Description

© 1998-2006, UtilMind Solutions®

The Visible property determines whether the button appears at the right side of acEdit control.

Use the Visible property to control the visibility of the button at runtime. If Visible is True, the button appears. If Visible is False, the button is not visible.

10.14.2.6Color

Applies to
acEdit, acNumberEdit, acIPEdit and acTreeComboBox components (in general, to all successors of TControl class).

Declaration
```
property Color: TColor;
```

Description
The Color property for acEdit component is the background color of the edit control. Use Color to read or change the background color of the control.

If a control's ParentColor property is True, then changing the Color property of the control's parent automatically changes the Color property of the control. When the value of the Color property is changed, the control's ParentColor property is automatically set to False.

See also
ColorDisabled property.

10.14.2.7ColorDisabled

Applies to
acEdit, acNumberEdit, acIPEdit and acTreeComboBox components.

Declaration
```
property ColorDisabled: TColor; // clBtnFace by default
```

Description
The ColorDisabled property for acEdit component is the background color of the edit control at disabled state (when Enabled is False). Use the ColorDisabled property to specify color for disabled edit control.

See also
Color property.

10.14.2.8CursorBorder

Applies to
acEdit, acNumberEdit and acTreeComboBox components.

Declaration
```
property CursorBorder: TCursor;
```

Description
The CursorBorder property specifies the image used to represent the mouse pointer when it passes over the non-client area of edit control.

See also
Cursor property of the button;
acAppCursors component and Additional cursors in the IDE.

© 1998-2006, UtilMind Solutions®

10.14.2.9 ValidateChars

Applies to
acEdit component.

Declaration
```
property ValidateChars: Boolean;
```

Description
The VaildateChars property controls whether you would like to validate the typed characters. If you would like to validate them — set the ValidateChars to True and specify characters allowed for input in the ValidChars property.

When the ValidateChars is True, the acEdit control will allow to input only characters specified in ValidChars property. All others, not specified symbols will be forbidden.

See also
ValidChars property.

10.14.2.10 ValidChars

Applies to
acEdit component.

Declaration
```
property ValidChars: String;
```

Description
The ValidChars property specifies the set of characters, allowed for input into the acEdit control when the ValidateChars is True.

When the ValidateChars is True, the acEdit control will allow to input only characters specified in ValidChars property. All others, not specified symbols will be forbidden.

For example, you would like to allow only lower-case latin characters without numbers. Set ValidChars to 'abcdefghijklmnopqrstuvwxyz' and ValidateChars to True.

See also
ValidateChars property.

10.14.3 Methods

10.14.3.1 SetSelection

Applies to
acEdit, acNumberEdit, acMRUComboBox, acLabeledComboBox and acTreeComboBox components.

Declaration
```
procedure SetSelection(SelStart: Integer = 0; SelEnd: Integer = -1);
```

Description
The SetSelection method used to select a range of characters in an edit control.

SelStart parameter specifies the starting character position of the selection.

© 1998-2006, UtilMind Solutions®

SelEnd parameter specifies the ending character position of the selection.

Remarks
If the *SelStart* parameter is 0 and the *SelEnd* parameter is -1, all the text in the edit control is selected. If *SelStart* is -1, any current selection is removed. The caret is placed at the end of the selection indicated by the greater of the two values *SelEnd* and *SelStart*.

10.14.4 Events

10.14.4.1OnButtonClick

Applies to
acEdit, acNumberEdit and acTreeComboBox components.

Declaration
```
property OnButtonClick: TNotifyEvent;
```

Description
The OnButtonClick event occurs when user clicks the custom button at the right side of edit control.

To make the button appear on the edit control — set Button.Visible to True, or to False if you'd like to hide the button.

See also
Button structure.

10.14.4.2OnButtonLeftMouseDown

Applies to
acEdit, acNumberEdit and acTreeComboBox components.

Declaration
```
property OnButtonLeftMouseDown: TNotifyEvent;
```

Description
The OnButtonLeftMouseDown event occurs when user presses the left mouse button over the button at the right side of edit control.

To make the button appear on edit control — set Button.Visible to True.

See also
Button structure.

10.14.4.3OnLabelClick

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and acTreeComboBox components.

Declaration
```
property OnLabelClick: TNotifyEvent;
```

Description
The OnLabelClick event occurs when user clicks the label attached to the control with a mouse.

To make the label appear near the edit control — set AttachedLabel.Visible to True, or to False if

© 1998-2006, UtilMind Solutions®

you'd like to hide the attached label.

See also
AttachedLabel structure.

10.14.4.4OnLabelDblClick

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and acTreeComboBox components.

Declaration
```
property OnLabelDblClick: TNotifyEvent;
```

Description
The OnLabelDblClick event occurs when user double clicks the label attached to the control with a mouse.

To make the label appear near the edit control — set AttachedLabel.Visible to True, or to False if you'd like to hide the attached label.

See also
AttachedLabel structure.

10.14.4.5OnMouseEnter

Applies to
acLabel, acURLLabel, acRichLabel, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox, acTreeComboBox, acHeader, acHeaderPanel, acTreeView and acListView components.

Declaration
```
property OnMouseEnter: TNotifyEvent;
```

Description
The OnMouseEnter occurs when mouse pointer hovers the edit control.

See also
OnMouseLeave event.

10.14.4.6OnMouseLeave

Applies to
acLabel, acURLLabel, acRichLabel, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox, acTreeComboBox, acHeader, acHeaderPanel, acTreeView and acListView components.

Declaration
```
property OnMouseLeave: TNotifyEvent;
```

Description
The OnMouseLeave occurs when mouse pointer leaves the edit control.

See also
OnMouseEnter event.

© 1998-2006, UtilMind Solutions®

10.15 acHeader

10.15.1 TacHeader

Overview

The acHeader is the control that has the functionality of Microsoft Outlook Header. Using this control you will be able to create highly customizable Outlook-style headers (rich-formatted text and icon at the left or right side of header, two customizeable buttons etc) and integrate them in your applications with other controls from the pack.

Snapshots

See also

acHeaderPanel component.

10.15.2 Properties

10.15.2.1AutoSize

Applies to

acHeader component.

Declaration

```
property AutoSize: Boolean; // True by default
```

Description

The AutoSize property specifies whether the header sizes itself automatically to accommodate the caption text shown on the header.

When the AutoSize is True, the text on the header (both, main caption and specified in RightCaption) will be always fits the height of the header.

See also

Font property.

10.15.2.2Background

Unit

acGraphics

Applies to

acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components.

Declaration

```
type
  TacBackground = class(TPersistent)
  published
    property Active: Boolean default False;
```

© 1998-2006, UtilMind Solutions®

```
    property GradientColor1: TColor default clBlue;
    property GradientColor2: TColor default clBlack;
    property GradientDirection: TacGradientDirection default
gdHorizontal;
    property Kind: TacBackgroundKind default bkGradient;
    property Picture: TPicture;
    property PictureStyle: TacBackgroundPictureStyle default bsTile;
  end;
```

Description

The Background structure used to specify custom background and its behaviors for the panel.

If you're tired of single-tone color panels and want to show the picture as "wallpaper", or smooth gradient effect under the controls of panel — simply activate it by specifying True to Active property and choose the Kind of background (specify which background you want: picture or gradient effect).

If you want to show the picture (wallpaper) — set Kind property to `bkPicture`, specify the image in Picture property and choose the style of wallpaper using PictureStyle property.

If you want to show the gradient effect instead of usual background — set Kind property to `bkGradient`, choose the style of effect in GradientDirection property (it can be vertical, horizontal or central), then specify the colors in GradientColor1 and GradientColor2 properties.

Alternatively, if you want to show your very own background / effect, you can use OnPaint event. Just write the OnPaint event handler and draw on the canvas of panel before all controls located on this panel will be displayed.

See also

OnPaint event.
acFormBackground component.

10.15.2.2.1 Active

Applies to

acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components as sub-property of Background structure.

Declaration

```
property Active: Boolean;
```

Description

The Active property of Background structure specifies whether the control should have custom background (gradient or picture).

Set Active to True to show the custom background (with smooth gradient effect or wallpaper specified in Picture property). Otherwise, set it to False to show standard single-tone background with color specified in Color property of control.

To specify the kind of background (gradient or picture) — use Kind property.

Alternatively, for better customization, you can use OnPaint event.

See also

Kind property;
OnPaint event.

© 1998-2006, UtilMind Solutions®

10.15.2.2.2 GradientColor1

Applies to
acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components as sub-property of Background structure.

Declaration
```
property GradientColor1: TColor;
```

Description
Specified the first (left if GradientDirection is horizontal or top if GradientDirection is vertical) gradient color of your form.

See also
GradientColor2, GradientDirection and Kind properties.

10.15.2.2.3 GradientColor2

Applies to
acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components as sub-property of Background structure.

Declaration
```
property GradientColor2: TColor;
```

Description
Specified the second (right if GradientDirection is horizontal or bottom if GradientDirection is vertical) gradient color of your form.

See also
GradientColor1, GradientDirection and Kind properties.

10.15.2.2.4 GradientDirection

Applies to
acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components as sub-property of Background structure.

Declaration
```
type
  TacGradientDirection = (gdVertical, gdHorizontal, gdCentral);

property GradientDirection: TacGradientDirection;
```

Description
Specifies the direction of the gradient effect. Direction can be vertical, horizontal or central. To specify the colors for gradient effect — use GradientColor1 and GradientColor2 properties.

Snapshots
```
GradientDirection = gdVertical
```

```
GradientDirection = gdHorizontal
```

© 1998-2006, UtilMind Solutions®

```
GradientDirection = gdCentral
```

See also
GradientColor1, GradientColor2 and Kind properties.

10.15.2.2.5 Kind

Applies to
acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components as sub-property of Background structure.

Declaration

```
type
  TacFormBackgroundKind = (bkPicture, bkGradient);

property Kind: TacFormBackgroundKind;
```

Description
The Kind property specifies how to paint the form background: using the picture (bkPicture), or drawing the gradient effect (bkGradient).

If you would like to paint background using the **picture**:
1. Set Kind property to `bkPicture`;
2. Specify the bitmap image in Picture property;
3. Select the style of drawing in the PictureStyle property.

If you wish to show the **gradient effect**:
1. Set Kind property to `bkGradient`;
2. Specify the direction of the effect in GradientDirection property;
3. Choose two colors to draw the effect in GradientColor1 and GradientColor2 properties.

See also
Active, GradientDirection and PictureStyle properties;
OnPaint event.

10.15.2.2.6 Picture

Applies to
acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components as sub-property of Background structure.

Declaration

```
property Picture: TPicture;
```

Description
The Picture property specifies the image that appears as form background. The Picture can contains an icon, betafile, bitmap graphic, or user-defined graphic to be displayed as the background (if Kind = bkPicture).

© 1998-2006, UtilMind Solutions®

Setting Picture at design time brings up the Picture Editor, which can be used to specify the file that contains the image.

To specify the style of drawing — use PictureStyle property.

See also
PictureStyle and Kind properties.

10.15.2.2.7 PictureStyle

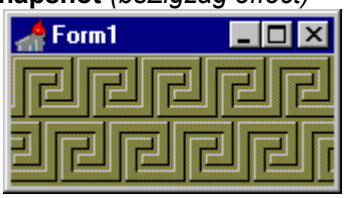
Applies to
acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components as sub-property of Background structure.

Declaration
```
property PictureStyle: TacBackgroundPictureStyle;
```

Description
The PictureStyle property specifies the effect used to draw the Picture. There are possible values:

Value	Meaning
bsTile	tiles the bitmap image on form's background;
bsStretch	stretches the image so that it exactly fits the bounds of the form. When the form resized, the image will be resized as well;
bsZigzag	almost the same as tile effect, but every next row of tile is shifted relatively to previous row, like bricks in the wall;
bsCenter	draws the image in the center of form.

Snapshot *(bsZigzag effect)*



See also
Picture and Kind properties.

10.15.2.3CaptionMenu

Applies to
acHeader and acHeaderPanel components.

Declaration
```
property CaptionMenu: TPopupMenu;
```

Description
The CaptionMenu property used to specify the popup menu for the header caption.

When user clicks the left mouse button over the caption on header panel, it will popup the menu (if specified) at the bottom-left position of caption, and trigger the OnCaptionClick event.

Snapshot

© 1998-2006, UtilMind Solutions®

See also
ImageMenu property;
OnCaptionClick and OnImageClick events;
CaptionMenu property of TacHeaderPanel component.

10.15.2.4Image

Applies to
acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components.

Declaration
```
property Image: TPicture;
```

Description
The Image property specifies the icon which appears at the left or right side of header control. This can be any picture in any format installed to your IDE (ICO, BMP, JPG or even GIF).

The placement of this icon controlled by ImageAlignment property.

If you don't want to specify stand-alone image, you can just specify the ImageIndex for the image from any ImageList.

Snapshot *(icon near text)*

See also
ImageAlignment, ImageIndex and ImageList properties.

10.15.2.5ImageAlignment

Applies to
acHeader, acHeaderPanel, acPicturePanel and acBitBtn components.

Declaration
```
type
  TacImageAlignment = (iaLeft, iaRight);

property ImageAlignment: TacImageAlignment;
```

Description
The ImageAlignment property controls the placement of the icon (specified by Image or ImageIndex properties) on the header control.

There are two possible values:

Value	Meaning
iaLeft	icon displayed at the left side of header control, shifting the caption to the right;
iaRight	icon displayed at the right side of control.

See also

© 1998-2006, UtilMind Solutions®

Image, ImageIndex and ImageList properties.

10.15.2.6ImageIndex

Applies to
acHeader, acHeaderPanel, acPicturePanel and acBitBtn components.

Declaration
```
property ImageIndex: Integer;
```

Description
The ImageIndex property specifies the index of image, which should be displayed as icon on the header control, from the image list, specified by ImageList property.

See also
Image, ImageAlignment and ImageList properties.

10.15.2.7ImageList

Applies to
acHeader, acHeaderPanel, acPicturePanel and acBitBtn components.

Declaration
```
property ImageList: TImageList;
```

Description
The ImageList property specifies the link with ImageList component, which contains the icon which should appear on the panel. The index of the icon in the ImageList should be specified in ImageIndex property.

See also
Image, ImageAlignment and ImageIndex properties.

10.15.2.8ImageMenu

Applies to
acHeader component.

Declaration
```
property ImageMenu: TPopupMenu;
```

Description
The ImageMenu property specifies the popup menu which should be displayed when user press the left mouse button over the Image on the header control.

See also
OnImageClick event;
CaptionMenu property.

10.15.2.9ImageTransparent

Applies to
acHeader, acHeaderPanel, acPicturePanel and acBitBtn components.

Declaration
```
property ImageTransparent: Boolean;
```

© 1998-2006, UtilMind Solutions®

Description

Note: available since Delphi 3 and higher.

See also
Image, ImageAlignment and ImageIndex properties.

RightButton
10.15.2.10

Applies to
acHeader and acHeaderPanel components.

Declaration
```
type
  TacHeaderRightButton = class
  published
    property AutoRoll: Boolean default True;
    property Cursor: TCursor default crDefault;
    property Enabled: Boolean default True;
    property Glyph: TBitmap;
    property Hint: String;
    property Kind: TacHeaderButtonKind default bkClose;
    property ShowEdges: Boolean default True;
    property ShowHint: Boolean default True;
    property UseCaptionColor: Boolean default True;
    property Visible: Boolean;
  end;
```

Description

See also
RightButton2 structure.

10.15.2.10.1 AutoRoll

Applies to
acHeader and acHeaderPanel components as subproperty of RightButton and RightButton2 structures.

Declaration
```
property AutoRoll: Boolean; // True by default
```

Description

See also

10.15.2.10.2 Cursor

Applies to
acHeader and acHeaderPanel components as subproperty of RightButton and RightButton2 structures.

Declaration
```
property Cursor: TCursor;
```

© 1998-2006, UtilMind Solutions®

Description

See also

10.15.2.10.3 Enabled

Applies to
acHeader and acHeaderPanel components as subproperty of RightButton and RightButton2 structures.

Declaration
```
property Enabled: Boolean; // True by default
```

Description

See also

10.15.2.10.4 Glyph

Applies to
acHeader and acHeaderPanel components as subproperty of RightButton and RightButton2 structures.

Declaration
```
property Glyph: TBitmap;
```

Description

See also

10.15.2.10.5 Hint

Applies to
acHeader and acHeaderPanel components as subproperty of RightButton and RightButton2 structures.

Declaration
```
property Hint: String;
```

Description

See also

10.15.2.10.6 Kind

Applies to
acHeader and acHeaderPanel components as subproperty of RightButton and RightButton2 structures.

Declaration
```
type
  TacHeaderButtonKind = (bkClose, bkPinUp, bkPinDown, bkRollUp,
bkRollDown, bkCustom);
```

© 1998-2006, UtilMind Solutions®

```
property Kind: TacHeaderButtonKind;
```

Description

See also

10.15.2.10.7 ShowEdges

Applies to
acHeader and acHeaderPanel components as subproperty of RightButton and RightButton2 structures.

Declaration
```
property ShowEdges: Boolean; // True by default
```

Description

See also

10.15.2.10.8 ShowHint

Applies to
acHeader and acHeaderPanel components as subproperty of RightButton and RightButton2 structures.

Declaration
```
property ShowHint: Boolean; // True by default
```

Description

See also

10.15.2.10.9 UseCaptionColor

Applies to
acHeader and acHeaderPanel components as subproperty of RightButton and RightButton2 structures.

Declaration
```
property UseCaptionColor: Boolean; // True by default
```

Description

See also

10.15.2.10.10 Visible

Applies to
acHeader and acHeaderPanel components as subproperty of RightButton and RightButton2 structures.

Declaration
```
property Visible: Boolean; // False by default
```

© 1998-2006, UtilMind Solutions®

Description

See also

RightButton2

10.15.2.11

Enter topic text here.

10.15.2.11.1 AutoRoll

Enter topic text here.

10.15.2.11.2 Cursor

Enter topic text here.

10.15.2.11.3 Enabled

Enter topic text here.

10.15.2.11.4 Glyph

Enter topic text here.

10.15.2.11.5 Hint

Enter topic text here.

10.15.2.11.6 Kind

Enter topic text here.

10.15.2.11.7 UseCaptionColor

Enter topic text here.

10.15.2.11.8 Visible

Enter topic text here.

RightCaption

10.15.2.12

Applies to
acHeader component.

Declaration
```
property RightCaption: String;
```

Description

See also
RightCaptionFont, RightButton and RightButton2 properties.

RightCaptionFont

10.15.2.13

Applies to
acHeader component.

© 1998-2006, UtilMind Solutions®

Declaration

```
property RightCaptionFont: TFont;
```

Description

See also

RightCaption, RightButton and RightButton2 properties.

TextStyle

10.15.2.14

Applies to

acRichLabel, acFormHelp, acInfoBox, acHeader, acHeaderPanel and acPicturePanel components.

Declaration

```
type
  TacTextStyle = class(TPersistent)
  published
    property PlainText: Boolean default False;
    property TagOpen: Char default '[';
    property TagClose: Char default ']';
  end;
```

Description

The TextStyle structure used to specify the style of text displayed by this component.

The PlainText property specifies whether you would like to use right-formatted text or just flat, plain text. TagOpen and TagClose properties used to specify prefered signs for brackets which identifies begin and end of tags used for rich formatting. By default every tag beginning with '**[**' sign and ending with '**]**' sign.

For example, when PlainText is False, TagOpen is '[' and TagClose is ']', following text:

```
[i]Hello[]  [red][b]World[def]!
```

will looks like:

Hello **World**!

Otherwise, if PlainText is True, the tags will not be used and users will see just flat text, including all tags.

See also

acRichLabel component and Rich-formatting tags overview.

10.15.2.14.1 PlainText

Applies to

acRichLabel, acInfoBox, acFormHelp, acHeader, acHeaderPanel and acPicturePanel components as subproperty of TextStyle structure.

Declaration

```
property PlainText: Boolean;
```

Description

The PlainText property controls whether the text should be displayed as rich text (if PlainText is False) or as plain, non-formatted text (if PlainText is True). When PlainText is True, the text will be displayed as usual flat text and nor tags will be used.

© 1998-2006, UtilMind Solutions®

See also
TagOpen and TagClose properties;
Using rich-formatting tags topic.

10.15.2.14.2 TagClose

Applies to
acRichLabel, acInfoBox, acFormHelp, acHeader, acHeaderPanel and acPicturePanel components as subproperty of TextStyle structure.

Declaration
```
property TagClose: Char;
```

Description
The TagClose property specifies the sign which marks the ending of formatting area of rich-text. The rich formatting tags of AppControls pack are similar to brackets "<>" of tags in HTML format. Default TagClose value is "**]**" sign.

Examples of standard tags
`[b]` - makes text after this tag **bold**.
`[i]` - makes text after this tag *italic*
`[biu]` - makes text after this tag ***bold, italic and underlined***.
`[]` - clears text formatting.
`[red]` - makes text after this tag red.
`[lime]` - makes text after this tag light green (lime).
`[def]` or `[default]` - returns the text formatting to default state as specified in Font property.

Note
If PlainText property is True, nor rich text will be displayed. All formating will be shown as usual flat text.

See also
TagOpen and PlainText properties and example of using rich-formatting tags.

10.15.2.14.3 TagOpen

Applies to
acRichLabel, acInfoBox, acFormHelp, acHeader, acHeaderPanel and acPicturePanel components as subproperty of TextStyle structure.

Declaration
```
property TagOpen: Char;
```

Description
The TagOpen property specifies the sign which marks the beginning of rich-text formatting area. The rich formatting tags of AppControls pack are similar to brackets "<>" of tags in HTML format. Default TagOpen value is "**[**" sign.

Examples of standard tags
`[b]` - makes text after this tag **bold**.
`[i]` - makes text after this tag *italic*
`[biu]` - makes text after this tag ***bold, italic and underlined***.
`[]` - clears text formatting.
`[red]` - makes text after this tag red.
`[lime]` - makes text after this tag light green (lime).

© 1998-2006, UtilMind Solutions®

`[def]` or `[default]` - returns the text formatting to default state as specified in Font property.

See also
TagClose and PlainText properties and example of using rich-formatting tags.

10.15.3 Events

10.15.3.1OnCaptionClick

Applies to
acHeader component.

Declaration
```
property OnCaptionClick: TNotifyEvent;
```

Description
The OnCaptionClick event occurs when user clicks the caption text on the header control.

See also
CaptionMenu property;
OnImageClick event.

10.15.3.2OnImageClick

Applies to
acHeader component.

Declaration
```
property OnImageClick: TNotifyEvent;
```

Description
The OnImageClick event occurs when user press the left mouse button over the Image on the header control.

See also
Image and ImageMenu properties;
OnCaptionClick event.

10.15.3.3OnMouseEnter

Applies to
acLabel, acURLLabel, acRichLabel, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acMRUComboBox, acHeader, acHeaderPanel, acTreeView, acListView components.

Declaration
```
property OnMouseEnter: TNotifyEvent;
```

Description
The OnMouseEnter occurs when mouse pointer hovers the header control.

See also
OnMouseLeave event.

© 1998-2006, UtilMind Solutions®

10.15.3.4OnMouseLeave

Applies to
acLabel, acURLLabel, acRichLabel, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acMRUComboBox, acHeader, acHeaderPanel, acTreeView, acListView components.

Declaration
```
property OnMouseLeave: TNotifyEvent;
```

Description
The OnMouseLeave occurs when mouse pointer leaves the header control.

See also
OnMouseEnter event.

10.15.3.5OnPaint

Applies to
acHeader, acHeaderPanel, acSplitterPanel and acPicturePanel components.

Declaration
```
property OnPaint: TNotifyEvent;
```

Description
The OnPaint event occurs when Windows requires the panel to paint, such as when the form or panel receives focus or becomes visible when it wasn't previously.

Your application can use this event to draw on the canvas of the header control. For example, to draw custom background.

See also
Background structure.

10.15.3.6OnRightButtonClick

Applies to
acHeader and acHeaderPanel components.

Declaration
```
property OnRightButtonClick: TNotifyEvent;
```

Description
The OnRightButtonClick event occurs when user clicks the button (which behavior should be specified in RightButton structure), at the right side of header control.

This button can change its state

Example

See also
RightButton and RightButton2 structures;
OnRightButton2Click event.

© 1998-2006, UtilMind Solutions®

10.15.3.7OnRightButton2Click

Applies to
acHeader and acHeaderPanel components.

Declaration
```
property OnRightButton2Click: TNotifyEvent;
```

Description
The OnRightButton2Click event occurs when user clicks the button (which behavior should be specified in RightButton2 structure), at the right side of header control.

See also
RightButton and RightButton2 structures;
OnRightButton2Click event.

10.16 acHeaderPanel

10.16.1 TacHeaderPanel

Overview

Features

How to use

See also
acHeader, acSplitterPanel and acPicturePanel components.

10.16.2 Properties

10.16.2.1Background

Unit
acGraphics

Applies to
acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components.

Declaration
```
type
  TacBackground = class(TPersistent)
  published
    property Active: Boolean default False;
    property GradientColor1: TColor default clBlue;
    property GradientColor2: TColor default clBlack;
    property GradientDirection: TacGradientDirection default
gdHorizontal;
    property Kind: TacBackgroundKind default bkGradient;
    property Picture: TPicture;
    property PictureStyle: TacBackgroundPictureStyle default bsTile;
  end;
```

Description
The Background structure used to specify custom background and its behaviors for the panel.

© 1998-2006, UtilMind Solutions®

If you're tired of single-tone color panels and want to show the picture as "wallpaper", or smooth gradient effect under the controls of panel — simply activate it by specifying True to Active property and choose the Kind of background (specify which background you want: picture or gradient effect).

If you want to show the picture (wallpaper) — set Kind property to `bkPicture`, specify the image in Picture property and choose the style of wallpaper using PictureStyle property.

If you want to show the gradient effect instead of usual background — set Kind property to `bkGradient`, choose the style of effect in GradientDirection property (it can be vertical, horizontal or central), then specify the colors in GradientColor1 and GradientColor2 properties.

Alternatively, if you want to show your very own background / effect, you can use OnPaint event. Just write the OnPaint event handler and draw on the canvas of panel before all controls located on this panel will be displayed.

See also
OnPaint event.
acFormBackground component.

10.16.2.1.1 Active

Applies to
acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components as sub-property of Background structure.

Declaration
```
property Active: Boolean;
```

Description
The Active property of Background structure specifies whether the control should have custom background (gradient or picture).

Set Active to True to show the custom background (with smooth gradient effect or wallpaper specified in Picture property). Otherwise, set it to False to show standard single-tone background with color specified in Color property of control.

To specify the kind of background (gradient or picture) — use Kind property.

Alternatively, for better customization, you can use OnPaint event.

See also
Kind property;
OnPaint event.

10.16.2.1.2 GradientColor1

Applies to
acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components as sub-property of Background structure.

Declaration
```
property GradientColor1: TColor;
```

Description
Specified the first (left if GradientDirection is horizontal or top if GradientDirection is vertical) gradient color of your form.

© 1998-2006, UtilMind Solutions®

See also
GradientColor2, GradientDirection and Kind properties.

10.16.2.1.3 GradientColor2

Applies to
acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components as sub-property of Background structure.

Declaration
```
property GradientColor2: TColor;
```

Description
Specified the second (right if GradientDirection is horizontal or bottom if GradientDirection is vertical) gradient color of your form.

See also
GradientColor1, GradientDirection and Kind properties.

10.16.2.1.4 GradientDirection

Applies to
acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components as sub-property of Background structure.

Declaration
```
type
  TacGradientDirection = (gdVertical, gdHorizontal, gdCentral);

property GradientDirection: TacGradientDirection;
```

Description
Specifies the direction of the gradient effect. Direction can be vertical, horizontal or central. To specify the colors for gradient effect — use GradientColor1 and GradientColor2 properties.

Snapshots
GradientDirection = gdVertical

GradientDirection = gdHorizontal

GradientDirection = gdCentral

See also
GradientColor1, GradientColor2 and Kind properties.

© 1998-2006, UtilMind Solutions®

10.16.2.1.5 Kind

Applies to
acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components as sub-property of Background structure.

Declaration
```
type
  TacFormBackgroundKind = (bkPicture, bkGradient);

property Kind: TacFormBackgroundKind;
```

Description
The Kind property specifies how to paint the form background: using the picture (bkPicture), or drawing the gradient effect (bkGradient).

If you would like to paint background using the **picture**:
1. Set Kind property to `bkPicture`;
2. Specify the bitmap image in Picture property;
3. Select the style of drawing in the PictureStyle property.

If you wish to show the **gradient effect**:
1. Set Kind property to `bkGradient`;
2. Specify the direction of the effect in GradientDirection property;
3. Choose two colors to draw the effect in GradientColor1 and GradientColor2 properties.

See also
Active, GradientDirection and PictureStyle properties;
OnPaint event.

10.16.2.1.6 Picture

Applies to
acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components as sub-property of Background structure.

Declaration
```
property Picture: TPicture;
```

Description
The Picture property specifies the image that appears as form background. The Picture can contains an icon, betafile, bitmap graphic, or user-defined graphic to be displayed as the background (if Kind = bkPicture).

Setting Picture at design time brings up the Picture Editor, which can be used to specify the file that contains the image.

To specify the style of drawing — use PictureStyle property.

See also
PictureStyle and Kind properties.

10.16.2.1.7 PictureStyle

Applies to
acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components as sub-property of

© 1998-2006, UtilMind Solutions®

Background structure.

Declaration

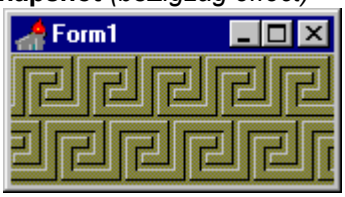
```
property PictureStyle: TacBackgroundPictureStyle;
```

Description

The PictureStyle property specifies the effect used to draw the Picture. There are possible values:

Value	Meaning
bsTile	tiles the bitmap image on form's background;
bsStretch	stretches the image so that it exactly fits the bounds of the form. When the form resized, the image will be resized as well;
bsZigzag	almost the same as tile effect, but every next row of tile is shifted relatively to previous row, like bricks in the wall;
bsCenter	draws the image in the center of form.

Snapshot *(bsZigzag effect)*



See also

Picture and Kind properties.

10.16.2.2BackgroundHeader

Unit

acGraphics

Applies to

acHeaderPanel component.

Declaration

```
type
  TacBackground = class(TPersistent)
  published
    property Active: Boolean default False;
    property GradientColor1: TColor default clBlue;
    property GradientColor2: TColor default clBlack;
    property GradientDirection: TacGradientDirection default
gdHorizontal;
    property Kind: TacBackgroundKind default bkGradient;
    property Picture: TPicture;
    property PictureStyle: TacBackgroundPictureStyle default bsTile;
  end;
```

Description

The BackgroundHeader structure used to specify custom background and its behaviors for the panel.

If you're tired of single-tone color panels and want to show the picture as "wallpaper", or smooth gradient effect under the controls of panel — simply activate it by specifying True to Active property

© 1998-2006, UtilMind Solutions®

and choose the Kind of background (specify which background you want: picture or gradient effect).

If you want to show the picture (wallpaper) — set Kind property to `bkPicture`, specify the image in Picture property and choose the style of wallpaper using PictureStyle property.

If you want to show the gradient effect instead of usual background — set Kind property to `bkGradient`, choose the style of effect in GradientDirection property (it can be vertical, horizontal or central), then specify the colors in GradientColor1 and GradientColor2 properties.

Alternatively, if you want to show your very own background / effect, you can use OnPaint event. Just write the OnPaint event handler and draw on the canvas of panel before all controls located on this panel will be displayed.

See also
OnPaint event.
acFormBackground component.

10.16.2.2.1 Active

Applies to
acHeaderPanel component as sub-property of BackgroundHeader structure.

Declaration
```
property Active: Boolean;
```

Description
The Active property of BackgroundHeader structure specifies whether the control should have custom background (gradient or picture).

Set Active to True to show the custom background (with smooth gradient effect or wallpaper specified in Picture property). Otherwise, set it to False to show standard single-tone background with color specified in Color property of control.

To specify the kind of background (gradient or picture) — use Kind property.

Alternatively, for better customization, you can use OnPaintHeader event.

See also
Kind property;
OnPaintHeader event.

10.16.2.2.2 GradientColor1

Applies to
acHeaderPanel component as sub-property of BackgroundHeader structure.

Declaration
```
property GradientColor1: TColor;
```

Description
Specified the first (left if GradientDirection is horizontal or top if GradientDirection is vertical) gradient color of your form.

See also
GradientColor2, GradientDirection and Kind properties.

© 1998-2006, UtilMind Solutions®

10.16.2.2.3 GradientColor2

Applies to
acHeaderPanel component as sub-property of BackgroundHeader structure.

Declaration
```
property GradientColor2: TColor;
```

Description
Specified the second (right if GradientDirection is horizontal or bottom if GradientDirection is vertical) gradient color of your form.

See also
GradientColor1, GradientDirection and Kind properties.

10.16.2.2.4 GradientDirection

Applies to
acHeaderPanel component as sub-property of BackgroundHeader structure.

Declaration
```
type
  TacGradientDirection = (gdVertical, gdHorizontal, gdCentral);

property GradientDirection: TacGradientDirection;
```

Description
Specifies the direction of the gradient effect. Direction can be vertical, horizontal or central. To specify the colors for gradient effect — use GradientColor1 and GradientColor2 properties.

Snapshots
```
GradientDirection = gdVertical
```

```
GradientDirection = gdHorizontal
```

```
GradientDirection = gdCentral
```

See also
GradientColor1, GradientColor2 and Kind properties.

10.16.2.2.5 Kind

Applies to
acHeaderPanel component as sub-property of BackgroundHeader structure.

© 1998-2006, UtilMind Solutions®

Declaration

```
type
  TacFormBackgroundKind = (bkPicture, bkGradient);

property Kind: TacFormBackgroundKind;
```

Description

The Kind property specifies how to paint the form background: using the picture (bkPicture), or drawing the gradient effect (bkGradient).

If you would like to paint background using the **picture**:

1. Set Kind property to `bkPicture`;
2. Specify the bitmap image in Picture property;
3. Select the style of drawing in the PictureStyle property.

If you wish to show the **gradient effect**:

1. Set Kind property to `bkGradient`;
2. Specify the direction of the effect in GradientDirection property;
3. Choose two colors to draw the effect in GradientColor1 and GradientColor2 properties.

See also

Active, GradientDirection and PictureStyle properties;
OnPaintHeader event.

10.16.2.2.6 Picture

Applies to

acHeaderPanel component as sub-property of BackgroundHeader structure.

Declaration

```
property Picture: TPicture;
```

Description

The Picture property specifies the image that appears as form background. The Picture can contains an icon, betafile, bitmap graphic, or user-defined graphic to be displayed as the background (if Kind = bkPicture).

Setting Picture at design time brings up the Picture Editor, which can be used to specify the file that contains the image.

To specify the style of drawing — use PictureStyle property.

See also

PictureStyle and Kind properties.

10.16.2.2.7 PictureStyle

Applies to

acHeaderPanel component as sub-property of BackgroundHeader structure.

Declaration

```
property PictureStyle: TacBackgroundPictureStyle;
```

Description

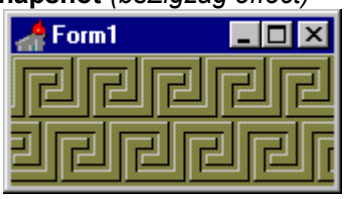
The PictureStyle property specifies the effect used to draw the Picture. There are possible values:

Value	Meaning

© 1998-2006, UtilMind Solutions®

bsTile	tiles the bitmap image on form's background;
bsStretch	stretches the image so that it exactly fits the bounds of the form. When the form resized, the image will be resized as well;
bsZigzag	almost the same as tile effect, but every next row of tile is shifted relatively to previous row, like bricks in the wall;
bsCenter	draws the image in the center of form.

Snapshot *(bsZigzag effect)*



See also

Picture and Kind properties.

10.16.2.3CaptionMenu

Applies to

acHeader and acHeaderPanel components.

Declaration

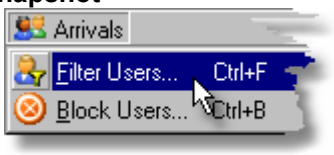
```
property CaptionMenu: TPopupMenu;
```

Description

The CaptionMenu property used to specify the popup menu for the header caption.

When user clicks the left mouse button over the caption on header panel, it will shows the menu (if specified) at the bottom-left position of caption, and trigger the OnHeaderCaptionClick event.

Snapshot



See also

OnHeaderCaptionClick event.
CaptionMenu property of TacHeaderPanel component.

10.16.2.4Flat

Applies to

acHeaderPanel component.

Declaration

```
property Flat: Boolean;
```

Description

See also

© 1998-2006, UtilMind Solutions®

FlatBorderColor property.

10.16.2.5FlatBorderColor

Applies to
acHeaderPanel component.

Declaration
```
property FlatBorderColor: TColor;
```

Description

See also
Flat property.

10.16.2.6HeaderHeight

Enter topic text here.

not published and read-only

10.16.2.7Image

Enter topic text here.

10.16.2.8ImageAlignment

```
type
  TacImageAlignment = (iaLeft, iaRight);

property ImageAlignment: TacImageAlignment;
```

10.16.2.9ImageIndex

Enter topic text here.

10.16.2.10 ImageList

Enter topic text here.

10.16.2.11 ImageMenu

Enter topic text here.

10.16.2.12 ImageTransparent

Enter topic text here.

Note: Available since Delphi 3

10.16.2.13 PopupMenuHeader

Enter topic text here.

© 1998-2006, UtilMind Solutions®

RightButton
10.16.2.14

Enter topic text here.

10.16.2.14.1 AutoRoll

Enter topic text here.

10.16.2.14.2 Enabled

Enter topic text here.

10.16.2.14.3 Glyph

Enter topic text here.

10.16.2.14.4 Kind

Enter topic text here.

10.16.2.14.5 UseCaptionColor

Enter topic text here.

10.16.2.14.6 Visible

Enter topic text here.

RightButton2
10.16.2.15

Enter topic text here.

10.16.2.15.1 AutoRoll

Enter topic text here.

10.16.2.15.2 Enabled

Enter topic text here.

10.16.2.15.3 Glyph

Enter topic text here.

10.16.2.15.4 Kind

Enter topic text here.

10.16.2.15.5 UseCaptionColor

Enter topic text here.

10.16.2.15.6 Visible

Enter topic text here.

RightCaption
10.16.2.16

Enter topic text here.

© 1998-2006, UtilMind Solutions®

RightCaptionFont
10.16.2.17

Enter topic text here.

RollOnDblClick
10.16.2.18

Enter topic text here.

RollRegistrySaver
10.16.2.19

Enter topic text here.

10.16.2.19.1 Enabled

Enter topic text here.

10.16.2.19.2 IniFileName

Enter topic text here.

10.16.2.19.3 IniSection

Enter topic text here.

10.16.2.19.4 Options

Enter topic text here.

10.16.2.19.5 RegKey

Enter topic text here.

10.16.2.19.6 RegLocation

Enter topic text here.

10.16.2.19.7 Storage

Enter topic text here.

ShowHeader
10.16.2.20

Enter topic text here.

ShowSizeGrip
10.16.2.21

Enter topic text here.

Text
10.16.2.22

Enter topic text here.

TextAlignment
10.16.2.23

Enter topic text here.

© 1998-2006, UtilMind Solutions®

TextFont
10.16.2.24

Enter topic text here.

TextStyle
10.16.2.25

Applies to
acRichLabel, acFormHelp, acInfoBox, acHeader, acHeaderPanel and acPicturePanel components.

Declaration
```
type
  TacTextStyle = class(TPersistent)
  published
    property PlainText: Boolean default False;
    property TagOpen: Char default '[';
    property TagClose: Char default ']';
  end;
```

Description
The TextStyle structure used to specify the style of text displayed by this component.

The PlainText property specifies whether you would like to use right-formatted text or just flat, plain text. TagOpen and TagClose properties used to specify prefered signs for brackets which identifies begin and end of tags used for rich formatting. By default every tag beginning with '**[**' sign and ending with '**]**' sign.

For example, when PlainText is False, TagOpen is '[' and TagClose is ']', following text:

```
[i]Hello[] [red][b]World[def]!
```

will looks like:

Hello **World**!

Otherwise, if PlainText is True, the tags will not be used and users will see just flat text, including all tags.

See also
acRichLabel component and Rich-formatting tags overview.

10.16.2.25.1 PlainText

Applies to
acRichLabel, acInfoBox, acFormHelp, acHeader, acHeaderPanel and acPicturePanel components as subproperty of TextStyle structure.

Declaration
```
property PlainText: Boolean;
```

Description
The PlainText property controls whether the text should be displayed as rich text (if PlainText is False) or as plain, non-formatted text (if PlainText is True). When PlainText is True, the text will be displayed as usual flat text and nor tags will be used.

See also
TagOpen and TagClose properties;
Using rich-formatting tags topic.

© 1998-2006, UtilMind Solutions®

10.16.2.25.2 TagClose

Applies to
acRichLabel, acInfoBox, acFormHelp, acHeader, acHeaderPanel and acPicturePanel components as subproperty of TextStyle structure.

Declaration
```
property TagClose: Char;
```

Description
The TagClose property specifies the sign which marks the ending of formatting area of rich-text. The rich formatting tags of AppControls pack are similar to brackets "<>" of tags in HTML format. Default TagClose value is "**]**" sign.

Examples of standard tags
`[b]` - makes text after this tag **bold**.
`[i]` - makes text after this tag *italic*
`[biu]` - makes text after this tag ***bold, italic and underlined***.
`[]` - clears text formatting.
`[red]` - makes text after this tag red.
`[lime]` - makes text after this tag light green (lime).
`[def]` or `[default]` - returns the text formatting to default state as specified in Font property.

Note
If PlainText property is True, nor rich text will be displayed. All formating will be shown as usual flat text.

See also
TagOpen and PlainText properties and example of using rich-formatting tags.

10.16.2.25.3 TagOpen

Applies to
acRichLabel, acInfoBox, acFormHelp, acHeader, acHeaderPanel and acPicturePanel components as subproperty of TextStyle structure.

Declaration
```
property TagOpen: Char;
```

Description
The TagOpen property specifies the sign which marks the beginning of rich-text formatting area. The rich formatting tags of AppControls pack are similar to brackets "<>" of tags in HTML format. Default TagOpen value is "**[**" sign.

Examples of standard tags
`[b]` - makes text after this tag **bold**.
`[i]` - makes text after this tag *italic*
`[biu]` - makes text after this tag ***bold, italic and underlined***.
`[]` - clears text formatting.
`[red]` - makes text after this tag red.
`[lime]` - makes text after this tag light green (lime).
`[def]` or `[default]` - returns the text formatting to default state as specified in Font property.

See also
TagClose and PlainText properties and example of using rich-formatting tags.

© 1998-2006, UtilMind Solutions®

TextLeft
10.16.2.26

Enter topic text here.

TextTop
10.16.2.27

Enter topic text here.

WordWrap
10.16.2.28

Enter topic text here.

10.16.3 Events

10.16.3.1 OnHeaderCaptionClick

Applies to
acHeaderPanel component.

Declaration
```
property OnHeaderCaptionClick: TNotifyEvent;
```

Description
The OnHeaderCaptionClick event occurs when user clicks the caption text on the header.

See also
CaptionMenu property;
OnHeaderClick and OnHeaderImageClick events.

10.16.3.2 OnHeaderClick

Applies to
acHeaderPanel component.

Declaration
```
property OnHeaderClick: TNotifyEvent;
```

Description

See also
OnHeaderCaptionClick and OnHeaderImageClick events.

10.16.3.3 OnHeaderImageClick

Applies to
acHeaderPanel component.

Declaration
```
property OnHeaderImageClick: TNotifyEvent;
```

Description

See also
Image and ImageMenu properties;
OnHeaderClick and OnHeaderCaptionClick events.

© 1998-2006, UtilMind Solutions®

10.16.3.4OnHeaderMouseEnter

Applies to
acHeaderPanel component.

Declaration
```
property OnHeaderMouseEnter: TNotifyEvent;
```

Description
The OnHeaderMouseEnter event occurs when mouse hovers the header of the header panel.

See also
OnHeaderMouseLeave, OnMouseEnter and OnMouseLeave events.

10.16.3.5OnHeaderMouseLeave

Applies to
acHeaderPanel component.

Declaration
```
property OnHeaderMouseLeave: TNotifyEvent;
```

Description
The OnHeaderMouseLeave event occurs when mouse leaves the header of the header panel.

See also
OnHeaderMouseEnter, OnMouseEnter and OnMouseLeave events.

10.16.3.6OnMouseEnter

Applies to
acLabel, acURLLabel, acRichLabel, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acMRUComboBox, acHeader, acHeaderPanel, acTreeView, acListView components.

Declaration
```
property OnMouseEnter: TNotifyEvent;
```

Description
The OnMouseEnter occurs when mouse pointer hovers the header panel.

If you wish to be notified when mouse hovers the header of the control — use OnHeaderMouseEnter event.

See also
OnMouseLeave, OnHeaderMouseEnter and OnHeaderMouseLeave events.

10.16.3.7OnMouseLeave

Applies to
acLabel, acURLLabel, acRichLabel, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acMRUComboBox, acHeader, acHeaderPanel, acTreeView, acListView components.

Declaration
```
property OnMouseLeave: TNotifyEvent;
```

© 1998-2006, UtilMind Solutions®

Description
The OnMouseLeave occurs when mouse pointer leaves the header panel.

If you wish to be notified when mouse leaves the header of the control — use OnHeaderMouseLeave event.

See also
OnMouseEnter, OnHeaderMouseEnter and OnHeaderMouseLeave events.

10.16.3.8 OnPaint

Applies to
acHeader, acHeaderPanel, acSplitterPanel and acPicturePanel components.

Declaration
```
property OnPaint: TNotifyEvent;
```

Description
The OnPaint event occurs when Windows requires the panel to paint, such as when the form or panel receives focus or becomes visible when it wasn't previously.

Your application can use this event to draw on the canvas of the panel. For example, to draw custom background.

This event works to draw background for the main panel only. If you wish to paint something on the header — write the OnPaintHeader event handler.

See also
Background and BackgroundHeader structures;
OnPaintHeader event.

10.16.3.9 OnPaintHeader

Applies to
acHeaderPanel component.

Declaration
```
property OnPaintHeader: TNotifyEvent;
```

Description
The OnPaintHeader event occurs when Windows requires the header of panel to paint, such as when the form or panel receives focus or becomes visible when it wasn't previously.

Your application can use this event to draw on the canvas of the header of panel. For example, to draw custom background.

This event works to draw background for the *header* only. If you wish to paint something on the main panel — write the OnPaint event handler.

See also
BackgroundHeader and Background structures;
OnPaint event.

© 1998-2006, UtilMind Solutions®

OnRightButtonClick
10.16.3.10

Enter topic text here.

OnRightButton2Click
10.16.3.11

Enter topic text here.

OnRolled
10.16.3.12

Enter topic text here.

10.17 acHostnameResolver

10.17.1 TacHostnameResolver

Overview

The acHostnameResolver component used to retrieve the host name by IP address or list of IP addresses, without freezing your application (it resolves hostnames in separate thread, because this lookup process can take rather long time and may completely freeze your application).

This component contains the internal cache for host names which arelady resolved, so it can quickly lookup for cached hostname and not use additional CPU time and bandwidth to resolve the same IP serveral times at row. Plus, the cache can be stored in system registry for further use after program restart.

Also, the acHostnameResolver can be used as simple monitor to check whether Internet connection is active, and to be notified when current PC goes online or offline (online status changes).

How to use?

You just need to specify the IP address or list of IP addresses to IPList property. If the AutoResolve is True, the specified IPs will be resolved each time when the list is updated (when you add new IP to the IPList queue). Otherwise, if AutoResolve is False, you need programmatically call Execute method to start lookup process.

Each time when component finished IP lookup, it triggers OnHostNameRetrieved event. Write this event handler to get the hostname and determinate its original IP address.

Note that component contains the internal cache for hostnames which already resolved. The size of the cache controlled by CacheSize property (maximum number of records which cache can contain). If CacheSize = 0, the component does not use cache at all. The cache can be stored to the registry for further use after program restarts. To enable saving/restoring of cached records, set RegistrySaver's Enabled property to True and specify the key where to store records in RegKey property.

Also, if you wish to quickly hostname which was already resolved from cache, you can use ResolveCachedIP method.

Additionally, the component can determinate current online status of machine (IsOnline property), and monitor it when computer goes online or offline (user connected or disconnected from Internet). If you wish to constantly check whether Internet connection available — use MonitorActive and MonitorInterval properties. To be notified when online status changes — write OnOnlineStatusChanged event handler.

© 1998-2006, UtilMind Solutions®

10.17.2 Properties

10.17.2.1AutoResolve

Applies to
acHostnameResolver and acConnectionList components.

Declaration
```
property AutoResolve: Boolean;
```

Description
The AutoResolve property determines whether the component should lookup for hostname automatically at once after specifying the IPList or adding new IP address to that list.

When AutoResolve is True, the thread which resolves hostnames will be started automatically and application can receive retrieved IPs in OnHostnameRetrieved event. When AutoResolve is False, you should programmatically start the thread using Execute method to start the lookup for hostnames.

See also
IPList property;
Execute and Abort methods;
OnHostnameRetrieved event.

10.17.2.2CacheSize

Applies to
acHostnameResolver and acConnectionList components.

Declaration
```
property CacheSize: Integer;
```

Description
The CacheSize property specifies the maximum number of IPs and resolved hostnames which can contain the internal cache of acHostnameResolver component.

The internal cache is the special list with IPs and Hostnames which are already retrieved. The component use this cache to avoid double resolving of hostname by particular IP address.

If you don't want to use cache, just set this property to 0.

The cached list can be automatically saved in registry for further use after application restarts. If you wish to store cache in the registry — just set RegistrySaver.Enabled property to True and specify the registry key which should hold records in RegistrySaver.RegKey property.

The component contains special method which helps to resolve cached IPs without executing the lookup thread. Use ResolveCachedIP to get the hostname by specified IP address.

See also
RegistrySaver structure;
ResolveCachedIP method.

10.17.2.3IPList

Applies to
acHostnameResolver component.

© 1998-2006, UtilMind Solutions®

Declaration

```
property IPList: TacStringList;
```

Description

The IPList property used to specify/add the IP address or list of IP addresses, whose host names should be retrieved and returned in OnHostnameRetrieved event (this event should be used to get resolved hostnames).

The IPList is just usual StringList and the queue for resolving of the hostnames for specified IPs in the internal thread of component. At once the host name resolved, its IP address (list item) will be automatically removed from list (queue).

If AutoResolve property set to True, the component automatically start to lookup for hostnames without additional actions. If AutoResolve is False, you need to call Execute method to start the lookup.

If the CacheSize is not 0 (that means that caching is enabled), the component will put resolved hostnames to the cache on each successful lookup. Also if you are use cache, you are able to quicly retrieve cached hostnames simply calling ResolveCachedIP method.

Example

```
var
  I: Integer;
begin
   // -- snip
  acHostnameResolver1.Abort;
  I := ConnectionList.Count;
  if I <> 0 then
    for I := 0 to I - 1 do
     begin
       acHostnameResolver1.IPList.Add(ConnectionList[I].LocalIPStr);
       acHostnameResolver1.IPList.Add(ConnectionList[I].RemoteIPStr);
     end;
    acHostnameResolver1.Execute;
    // -- snip
end;
```

See also

AutoResolve and CacheSize properties;
Execute, Abort and ResolveCachedIP method;
OnHostnameRetrieved event.

10.17.2.4IsOnline

Applies to

acHostnameResolver and acConnectionList components.

Declaration

```
property IsOnline: Boolean; // read-only!
```

Description

The IsOnline is the read-only property which determines current online status of machine. It returns True if computer connected to Internet or False otherwise.

The component is able to constantly monitor the connection and notify the application when computer goes online or offline in OnOnlineStatusChanged event. The status monitoring performed

© 1998-2006, UtilMind Solutions®

between time intervals, specified in MonitorInterval property, if MonitorActive is True. To disable monitoring, set MonitorActive property to False.

Remark

This property is useless on machines with Ethernet cable connection. IsOnline always returns True (online), even if user cut cable off. Unfortunately there is no convinient way to determinate whether computer is online. However, for sure you may try to ping some host or something. This property mostly good for Dial-up modem connections only.

See also

IsOnline, MonitorActive and MonitorInterval properties;
OnOnlineStatusChanged event.

10.17.2.5LocalHostname

Applies to

acHostnameResolver and acConnectionList components.

Declaration

```
property LocalHostname: String; // read-only!
```

Description

The LocalHostname is read-only property which determines the hostname of local machine. Also you can determinate its current IP address reading LocalIP property.

See also

LocalIP property.

10.17.2.6LocalIP

Applies to

acHostnameResolver and acConnectionList components.

Declaration

```
property LocalIP: String; // read-only!
```

Description

The LocalIP is read-only property which determines the IP address of local machine (represented by text string). Also you can determinate the hostname of current machine reading LocalHostname property.

See also

LocalHostname property.

10.17.2.7MonitorActive

Applies to

acHostnameResolver and acConnectionList components.

Declaration

```
property MonitorActive: Boolean;
```

Description

The MonitorActive property specifies whether the component should monitor the online status of current PC (whether it connected to Internet).

© 1998-2006, UtilMind Solutions®

The time periods between checking the online status should be specified in MonitorInterval property. To be notified when online status changed — write OnOnlineStatusChanged event handler.

Remark

Unfortunately component is unable to detect whether computer is not connected to Internet, if user are use Ethernet cable connection. The IsOnline property will always return True even if user cut cable off.

See also

IsOnline, MonitorActive and MonitorInterval properties;
OnOnlineStatusChanged event.

10.17.2.8MonitorInterval

Applies to

acHostnameResolver and acConnectionList components.

Declaration

```
property MonitorInterval: Integer; // 100 milliseconds by default
```

Description

The MonitorInterval property specifies the time periods, in milliseconds, between checking the online status of computer, if MonitorActive property already set to True.

To be notified when online status changed — write OnOnlineStatusChanged event handler.

Remark

Unfortunately component is unable to detect whether computer is not connected to Internet, if user are use Ethernet cable connection. The IsOnline property will always return True even if user cut cable off.

See also

IsOnline, MonitorActive and MonitorInterval properties;
OnOnlineStatusChanged event.

10.17.2.9Priority

Applies to

acHostnameResolver and acConnectionList components.

Declaration

```
property Priority: TThreadPriority;
```

Description

Priority indicates the priority used when scheduling the thread (which retrieves host names by IP addresses). Adjust the priority higher or lower as needed.

TThreadPriority type defines the possible values for the Priority property of the acThread component, as defined in the following table. The system schedules CPU cycles to each thread based on a priority scale; the Priority property adjusts a thread's priority higher or lower on the scale.

Values	Meaning
tpIdle	The thread executes only when the system is idle. The system will not interrupt other threads to execute a thread with tpIdle priority.

© 1998-2006, UtilMind Solutions®

tpLowest	The thread's priority is two points below normal.
tpLower	The thread's priority is one point below normal.
tpNormal	The thread has normal priority.
tpHigher	The thread's priority is one point above normal.
tpHighest	The thread's priority is two points above normal.
tpTimeCritical	The thread gets highest priority.

Warning

Boosting the thread priority of a CPU intensive operation may "starve" the other threads in the application. Only apply priority boosts to threads that spend most of their time waiting for external events.

See also

acThread component.

RegistrySaver

10.17.2.10

Applies to

acHostnameResolver, acListView and acAppBar components.

Declaration

```
type
  TacHostnameResolverRegistrySaver = class
  published
    property Enabled: Boolean;
    property RegKey: String;
    property RegLocation: TacRegLocation;
  end;

property RegistrySaver: TacHostnameResolverRegistrySaver;
```

Description

The RegistrySaver structure used to save and restore the cached hostnames to/from system registry.

If you wish to save the internal cache and restore it on next allication restart — just specify the RegLocation and RegKey where to store these settings, and set Enabled to True.

10.17.2.10.1 Enabled

Applies to

acHostnameResolver component as subproperty of RegistrySaver structure.

Declaration

```
property Enabled: Boolean;
```

Description

The Enabled property of the RegistrySaver structure controls whether you would like to save the content of the internal cache (with IPs and their resolved hostnames) to the registry, and restore the cache on each program restart.

If the RegistrySaver is Enabled, the cache will be stored to the registry key specified in RegKey property.

© 1998-2006, UtilMind Solutions®

Specify maximum size of cache in CacheSize property.

See also
CacheSize, RegKey and RegLocation properties.

10.17.2.10.2 RegKey

Applies to
acHostnameResolver component as subproperty of RegistrySaver structure.

Declaration
```
property RegKey: String;
```

Description
The RegKey property defines the path to the key in the system registry where you would like to store the content of the internal cache (with IPs and their resolved hostnames).

Default property value is `'\Software\CompanyName\ProgramName\Default'`. You should point the RegKey to location where stored the configuration settings of your own application. For example: `'\Software\UtilMind Solutions\The Cool Program\Hostname Cache'`.

Specify maximum size of cache in CacheSize property.

See also
CacheSize, RegLocation and Enabled properties.

10.17.2.10.3 RegLocation

Applies to
acHostnameResolver component as subproperty of RegistrySaver structure.

Declaration
```
type
  TacRegLocation = (rlCurrentUser, rlLocalMachine);

property RegLocation: TacRegLocation;
```

Description
The RegLocation specifies the root key of the hierarchy of subkeys, where you would like to store the the content of the internal cache (with IPs and their resolved hostnames). RegLocation property can point to HKEY_CURRENT_USER (rlCurrentUser) or HKEY_LOCAL_MACHINE (rlLocalMachine) root keys.

When RegLocation is `rlCurrentUser`, acHostnameResolver stores all settings in the registry section for current logged user. When RegLocation is `rlLocalMachine` settings will be saved for ALL users of current machine.

Specify maximum size of cache in CacheSize property.

See also
CacheSize, RegKey and Enabled properties.

© 1998-2006, UtilMind Solutions®

10.17.3 Methods

10.17.3.1Execute

Applies to
acHostnameResolver component.

Declaration
```
procedure Execute;
```

Description
The Execute method starts the thread which resolves hostnames by IP addresses specified in IPList property.

To get resolved hostnames — write OnHostnameRetrieved event.

To terminate hostname lookup process "manually", without awaiting finishing of the process — call Abort method.

if AutoResolve property set to True, you don't need to call Execute method, because it will retrieve IPs automatically at once you add or specify something in IPList.

Example
```
var
  I: Integer;
begin
   // -- snip
  acHostnameResolver1.Abort;
  I := ConnectionList.Count;
  if I <> 0 then
    for I := 0 to I - 1 do
     begin
       acHostnameResolver1.IPList.Add(ConnectionList[I].LocalIPStr);
       acHostnameResolver1.IPList.Add(ConnectionList[I].RemoteIPStr);
     end;
    acHostnameResolver1.Execute;
    // -- snip
end;
```

See also
AutoResolve and IPList properties;
Execute, Abort and ResolveCachedIP methods;
OnHostnameRetrieved event.

10.17.3.2Abort

Applies to
acHostnameResolver component.

Declaration
```
procedure Abort;
```

Description
The Abort method imediately terminate the thread and process of resolving the hostnames by IP adresses specified in IPList property.

To execute thread again to continue hostname lookup — call Execute method OR set AutoResolve

© 1998-2006, UtilMind Solutions®

property to True and specify/add something to IPList.

See also
AutoResolve and IPList properties;
Execute, Abort and ResolveCachedIP methods;
OnHostnameRetrieved event.

10.17.3.3ResolveCachedIP

Applies to
acHostnameResolver component.

Declaration
```
function ResolveCachedIP(const IP: String; PutToQueueIfNotYetResolved:
Boolean = False): String;
```

Description
The ResolveCachedIP method tries to retrieve the cached hostname by IP address specified in *IP* parameter.

If the hostname still not in the cache, it can put it to the queue for further resolving (IPList property), if *PutToQueueIfNotYetResolved* parameter is True.

However, this method only retrieves correct host name if it ALREADY was resolved and currently are in internal cache of component. If the hostname can not be retrieved IMEDIATELY (it was not found in cache), function just returns the same IP address as specified in parameter.

See also
AutoResolve and IPList properties;
Execute, Abort and ResolveCachedIP methods;
OnHostnameRetrieved event.

10.17.4 Events

10.17.4.1OnHostnameRetrieved

Applies to
acHostnameResolver and acConnectionList components.

Declaration
```
type
  TacHostnameRetrievedEvent = procedure(Sender: TObject; const IPStr,
Hostname: String) of object;

property OnHostnameRetrieved: TacHostnameRetrievedEvent;
```

Description
The OnHostnameRetrieved event occurs when component has retrieved some IP address from the list (queue) specified in IPList property.

The event passes to its handler following parameters:

Parameter	Meaning
IPStr	the original IP address which has been specified in IPList property, but already remoted from that queue because the host name was retrieved.
Hostname	the name of machine retrieved by IP address after lookup.

© 1998-2006, UtilMind Solutions®

If you are use cache (CacheSize > 0), the retrieved hostname will be put to the cache, so next time when you will resolve it, the process will be much quicker. Also you can use ResolveCachedIP function to get it very quickly.

See also
AutoResolve and IPList properties;
Execute, Abort and ResolveCachedIP methods;
OnHostnameRetrieved event.

10.17.4.2OnOnlineStatusChanged

Applies to
acHostnameResolver and acConnectionList components.

Declaration
```
property OnOnlineStatusChanged: TNotifyEvent;
```

Description
The OnOnlineStatusChanged event occurs when the component detects that machine become connected to Internet or connection lost.

To get current online status at any time — read IsOnline property. True means that machine currently connected to Internet and False means that it disconnected.

Remarks
1. The online status monitoring only performed if MonitorActive property is True.
2. Component checks online status with intervals specified in MonitorInterval property.

See also
IsOnline, MonitorActive and MonitorInterval properties.

10.18 acHTMLViewer

10.18.1 TacHTMLViewer

Overview
The acHTMLViewer component is the easy to use HTML viewer, based on WebBrowser control of Internet Explorer. It allows to specify the HTML content without navigation through the Web, and provides easy access to some frequently used properties, intended to change the look and behavior of the emdedded browser window.

This component available only in Delphi/BCB 5 and higher versions.

How to use?
If you wish to show some HTML-formatted text with effects of Internet Explorer — just drop the control onto your form, and specify the contents to HTML property. To change the color and style of the frame around the viewer — use BorderColor and BorderStyle properties.

If you don't want to show the default Internet Explorer's popup menu — set NoStdPopupMenu to True (and use PopupMenu property to specify custom menu). To hide the scroll bars — set NoScroll to True.

If you think that this wrapper over TWebBrowser control does not let you enough functionality (for example, you want to use some additional properties of TWebBrowser control or even navigate over the Web) — just use WebBrowser property for full access of all its properties and methods.

© 1998-2006, UtilMind Solutions®

Remember that this is just wrapper over Internet Explorer, not native VCL implementation.

Note for C++ Builder developers

When you will build the project with HTMLViewer, you may get following linker error:
[C++ Error] SHDocVw.hpp(893): E2293) expected, at the following line:

```
/* TWinControl.CreateParented */ inline __fastcall TWebBrowser(HWND ParentWindow)
: Olectrls::TOleControl(ParentWindow) { }
```

Please manually edit the SHDocVw.hpp and change this line to

```
/* TWinControl.CreateParented */ inline __fastcall TWebBrowser(HANDLE
ParentWindow) : Olectrls::TOleControl(ParentWindow) { }
```

Also you can get another linker error:
IESniffer.hpp E2209 Unable to open include file 'MSHTML.hpp'.
Please just replace MSHTML.hpp to MSHTML.h. For some reason BCB comes with MSHTML.h file instead of *.hpp.

See also

acRichLabel and acIESniffer components.

10.18.2 Properties

10.18.2.1BorderColor

Applies to

acHTMLViewer component.

Declaration

```
property BorderColor: TColor;
```

Description

The BorderColor property specifies the color of frame around the HTML viewer (embedded Internet Explorer window).

See also

Color and BorderStyle properties.

10.18.2.2BorderStyle

Applies to

acHTMLViewer component.

Declaration

```
type
  TacHTMLViewerBorderStyle = (hvbsDefault, hvbsNone,
                              hvbsDotted,  hvbsDashed,
                              hvbsSolid,   hvbsDouble,
                              hvbsGroove,  hvbsRidge,
                              hvbsInset,   hvbsWndInset,
                              hvbsOutset);

property BorderStyle: TacHTMLViewerBorderStyle; // hvbsNone by default
```

Description

The BorderStyle property specifies the style of frame around the HTML viewer (embedded Internet Explorer window).

© 1998-2006, UtilMind Solutions®

Set BorderStyle to `hvbsNone` to hide the border at all, or set it to `hvbsDefault` to return to default 3D-style border.

See also
BorderColor and NoScroll properties.

10.18.2.3Color

Applies to
acHTMLViewer component.

Declaration
```
property Color: TColor;
```

Description
The Color property specifies the default background color for embedded Internet Explorer window.

However the background color can be changed in the HTML code, if you specify the custom color in `<body>` tag or the CSS template, which path specified in HTML code.

See also
HTML, BorderColor and BorderStyle properties.

10.18.2.4HTML

Applies to
acHTMLViewer component.

Declaration
```
property HTML: TStringList;
```

Description
The HTML property specifies the contents for embedded Internet Explorer window. You can specify any HTML code or even plain text, and it imediately will appear in viewer.

See also
WebBrowser property.

10.18.2.5Margins

Applies to
acHTMLViewer component.

Declaration
```
type
  TacHTMLViewerMargin = -1..MaxInt;
  TacHTMLViewerMargins = class(TPersistent)
    property Left: TacHTMLViewerMargin;
    property Top: TacHTMLViewerMargin;
    property Right: TacHTMLViewerMargin;
    property Bottom: TacHTMLViewerMargin;
  end;

property Margins: TacHTMLViewerMargins;
```

Description
The Margins structure controls the spacing between the border of HTMLViewer and its contents.

© 1998-2006, UtilMind Solutions®

Set all margins to 0, if you wish to show the contents of HTMLViewer without any spacing.

See also
HTML and BorderStyle properties.

10.18.2.5.1 Left

Applies to
acHTMLViewer component as subproperty of Margins structure.

Declaration
```
property Left: Integer; // -1..MaxInt
```

Description
The Left property specifies the spacing between the left border of HTMLViewer and the left side of its contents.

See also
Top, Right and Bottom properties.

10.18.2.5.2 Top

Applies to
acHTMLViewer component as subproperty of Margins structure.

Declaration
```
property Top: Integer; // -1..MaxInt
```

Description
The Top property specifies the spacing between the top border of HTMLViewer and the top side of its contents.

See also
Left, Right and Bottom properties.

10.18.2.5.3 Right

Applies to
acHTMLViewer component as subproperty of Margins structure.

Declaration
```
property Right: Integer; // -1..MaxInt
```

Description
The Right property specifies the spacing between the right border of HTMLViewer and the right side of its contents.

See also
Left, Top and Bottom properties.

10.18.2.5.4 Bottom

Applies to
acHTMLViewer component as subproperty of Margins structure.

Declaration
```
property Bottom: Integer; // -1..MaxInt
```

© 1998-2006, UtilMind Solutions®

Description
The Bottom property specifies the spacing between the bottom border of HTMLViewer and the bottom side of its contents.

See also
Left, Top and Right properties.

10.18.2.6NoScroll

Applies to
acHTMLViewer component.

Declaration
```
property NoScroll: Boolean;
```

Description
The NoScroll property specifies whether you would like to show or hide the scroll bars from the HMTL Viewer (embedded Internet Explorer window).

Set NoScroll to True, if you don't want to show the scroll bars in viewer.

See also
NoStdPopupMenu and BorderStyle properties.

10.18.2.7NoStdPopupMenu

Applies to
acHTMLViewer component.

Declaration
```
property NoStdPopupMenu: Boolean;
```

Description
The NoStdPopupMenu property specifies whether you want to make the default popup menu appears on screen, when user clicks right mouse button over HTML viewer (embedded Internet Explorer window).

Set NoStdPopupMenu to True if you don't want to show the default IE menu, or want to specify your own, custom popup menu.

See also
PopupMenu and NoScroll properties.

10.18.2.8PopupMenu

Applies to
acHTMLViewer component.

Declaration
```
property PopupMenu: TPopupMenu;
```

Description
The PopupMenu property specifies the custom popup menu for HTMLViewer. This menu will appears when user clicks right mouse button over the embedded Internet Explorer window, instead of standard IE menu.

© 1998-2006, UtilMind Solutions®

Note
Before specifying your custom popup menu — set NoStdPopupMenu property to True.

See also
NoStdPopupMenu properties.

10.18.2.9 WebBrowser

Applies to
acHTMLViewer component.

Declaration
```
property WebBrowser: TWebBrowser; // read-only!
```

Description
The WebBrowser property used to access the properties and methods of the TWebBrowser control, which stays in the basis of this HTMLViewer component.

Using this property you can read or specify many additional properties, which not implemented in this simple wrapper over TWebBrowser, and call the methods (i.e: Navigate method to view the Web contents). See manuals on TWebBrowser component for more details.

See also
HTML property.

10.19 acImagesComboBox

10.19.1 TacImagesComboBox

Overview
The acImagesComboBox is the enhanced combo box which lets you to specify images for every item of the drop-down list. Also it contains the AttachedLabel structure which lets to operate with the label associated with the edit control and specify its behavior and position near the control.

How to use?
First you need to specify the ImageList which contains the images which should appears beside individual items of combo box. Point the list of images at Images property (note that if image list is not specified the combo box will show just plain strings). The height of the combo box will be automatically adjusted accordingly to the height of images.

To specify the text and image indexes for portions of the drop-down list — use Items property. The image index should be specified as integer number and separated from the text of item by "|" character. For example, if the Items contains the line: '0|The Item', this means that item should display text "The Item" and use first image specified in the image list (with index = 0). If the image index is not specified it will show only text without image.

Example
```
acImagesComboBox1.Items.Add('0|First item (imageindex=0)');
acImagesComboBox1.Items.Add('Item without image');
acImagesComboBox1.Items.Add('1|Another item (imageindex = 1)');
```

Snapshot

© 1998-2006, UtilMind Solutions®

See also
acMRUComboBox component.

10.19.2 Properties

10.19.2.1AttachedLabel

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components.

Unit
acAttachedLabel

Declaration

```
type
  TacAttachedLabelPosition = (lpAbove, lpBelow, lpLeft, lpRight);
  TacAttachedLabelSpacingKind = (skNearPoint, skFarPoint);
  TacAttachedLabel = class
    property Caption: TCaption;
    property Color: TColor default clBtnFace;
    property Cursor: TCursor default crDefault;
    property Enabled: Boolean;
    property FocusOnClick: Boolean default True;
    property Font: TFont;
    property Hint: String;
    property ParentColor: Boolean default True;
    property ParentFont: Boolean default True;
    property ParentShowHint: Boolean default True;
    property PopupMenu: TPopupMenu;
    property Position: TacAttachedLabelPosition default lpLeft;
    property ShowAccelChar: Boolean default True;
    property ShowHint: Boolean default False;
    property Spacing: Integer default 4;
    property SpacingKind: TacAttachedLabelSpacingKind default
skNearPoint;
    property Transparent: Boolean default False;
    property Visible: Boolean;
    property WordWrap: Boolean default False;
  end;
```

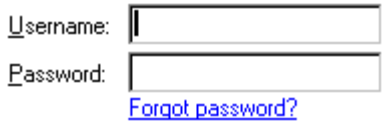
© 1998-2006, UtilMind Solutions®

Description

The AttachedLabel structure lets you to operate with the label attached to the some side of control (at the left or right side, above or below). You can specify the behaviour of label, its position near the control, and accelerator key for the control (character in caption, after ampersand (&)).

The label always associated with the control. When user press the accelerator key (Alt + Character, specified after ampersand (&)), the input focus will be imediately switched to the labeled control.

Screenshot (the labels at the left side is attached to the edit control):



See also

OnLabelClick and OnLabelDblClick events.

10.19.2.1.1 Caption

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property Caption: TCaption;
```

Description

The Caption property specified a text string which appers on label attached to control (attached label).

To underline a character in a Caption that labels a component, include an ampersand (&) before the character. This type of character is called an accelerator character. The user can then select the component by pressing Alt while typing the underlined character. To display an ampersand character in the caption, use two ampersands (&&).

The label always associated with the control. When user press the accelerator key (Alt + Character, specified after ampersand (&)), the input focus will be imediately switched to the labeled control.

Screenshot (the labels at the left side is attached to the edit control):

See also

ShowAccelChar, Color, Font, Transparent, Visible and WordWrap properties.

10.19.2.1.2 Color

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration

© 1998-2006, UtilMind Solutions®

```
property Color: TColor;
```

Description

The Color property specifies the background color for label attached to control. Use Color to read or change the background color of the control.

If a control's ParentColor property is true, then changing the Color property of the control's parent automatically changes the Color property of the control. When the value of the Color property is changed, the control's ParentColor property is automatically set to False.

If you don't want to show background — set Transparent property to True.

See also

ParentColor and Transparent properties.

10.19.2.1.3 Cursor

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property Cursor: TCursor;
```

Description

The Cursor property specifies the image used to represent the mouse pointer when it passes into the region covered by this label attached to control.

Change the value of Cursor to provide feedback to the user when the mouse pointer enters the control. The value of Cursor is the index of the cursor in the list of cursors maintained by the global variable, Screen. In addition to the built-in cursors provided by TScreen, applications can add custom cursors to the list.

See also

Additional Cursors

10.19.2.1.4 Enabled

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property Enabled: Boolean;
```

Description

The Enabled property controls whether the control responds to mouse, keyboard, and timer events.

Use Enabled to change the availability of the control to the user. To disable a control, set Enabled to false. Disabled controls appear dimmed. If Enabled is false, the control ignores mouse, keyboard, and timer events.

To re-enable a control, set Enabled to true. The control is no longer dimmed, and the user can use the control.

See also

© 1998-2006, UtilMind Solutions®

Caption and Color properties.

10.19.2.1.5 FocusOnClick

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property FocusOnClick: Boolean; // True by default
```

Description
The FocusOnClick property controls whether the control, associated with the attached label, should accept focus when user clicks the label with mouse.

Set FocusOnClick to True if you want to switch focus to the control when user clicks the label, or set it to False otherwise.

Note
Additionally you can use OnLabelClick and OnLabelDblClick events of the control.

10.19.2.1.6 Font

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property Font: TFont;
```

Description
The Font property controls the attributes of text written on the attached label of control.

To change to a new font, specify a new TFont object. To modify a font, change the value of the Charset, Color, Height, Name, Pitch, Size, or Style of the TFont object.

See also
ParentFont property.

10.19.2.1.7 Hint

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property Hint: String;
```

Description
The Hint property contains the text string that can appear when the user moves the mouse over the label attached control.

If you are using the FormHelp component to show the context-sensitive help taken from secondary part of hint AND the Hint for attached label is not specified, the FormHelp will use the hint of main control.

© 1998-2006, UtilMind Solutions®

See also
ShowHint and ParentShowHint properties;
acFormHelp and acFormHints components.

10.19.2.1.8 ParentColor

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property ParentColor: Boolean;
```

Description
The ParentColor determines where an attached label looks for its color information.

To have a control use the same color as its parent control, set ParentColor to true If ParentColor is false the control uses its own Color property.

Set ParentColor to true for all controls in order to ensure that all the controls on a form have a uniform appearance. For example, if ParentColor is true for all controls in a form, changing the background color of the form to gray causes all the controls on the form to also have a gray background.

When the value of a control's Color property changes, ParentColor becomes false automatically.

See also
Color and Transparent properties.

10.19.2.1.9 ParentFont

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property ParentFont: Boolean;
```

Description
The ParentFont property determines where a control looks for its font information.

To have a control use the same font as its parent control, set ParentFont to true If ParentFont is false the label uses its own Font property.

Set ParentFont to true for all controls in order to ensure that all the controls on a form have a uniform appearance. For example, if ParentFont is true for all controls in a form, changing the form's Font property to 12-point Courier causes all controls on the form to use that font.

When the value of a control's Font property changes, ParentFont becomes false automatically.

When ParentFont is true for a form, the form uses the default font.

See also
Font property.

© 1998-2006, UtilMind Solutions®

10.19.2.1.10 ParentShowHint

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property ParentShowHint: Boolean;
```

Description
The ParentShowHint property determines where an attached label looks to find out if its Help Hint should be shown.

Use ParentShowHint to ensure that all the controls on a form either uniformly show their Help Hints or uniformly do not show them.

If ParentShowHint is true, the label uses the ShowHint property value of its parent. If ParentShowHint is false, the label uses the value of its own ShowHint property.

To provide Help Hints for only selected controls on a form, set the ShowHint property for those controls that should have Help Hints to true, and ParentShowHint becomes false automatically.

Note
Enable or disable all Help Hints for the entire application using the ShowHint property of the application object.

See also
ShowHint and Hint properties;
acFormHelp and acFormHints components.

10.19.2.1.11 PopupMenu

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property PopupMenu: TPopupMenu;
```

Description
The PopupMenu property Identifies the pop-up menu associated with the label attached to control.

Assign a value to PopupMenu to make a pop-up menu appear when the user selects the label and clicks the right mouse button. If the TPopupMenu's AutoPopup property is True, the pop-up menu appears automatically. If the menu's AutoPopup property is False, display the menu with a call to its Popup method from the control's OnContextPopup event handler.

10.19.2.1.12 Position

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
type
  TacAttachedLabelPosition = (lpAbove, lpBelow, lpLeft, lpRight);
```

© 1998-2006, UtilMind Solutions®

```
property Position: TacAttachedLabelPosition;
```

Description

The Position property of the attached label specifies the layout of the label near the control.

You can make the label visible above, below, at the left or the right side of control.

See also

Spacing and SpacingKind properties.

10.19.2.1.13 ShowAccelChar

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property ShowAccelChar: Boolean;
```

Description

The ShowAccelChar property determines how an ampersand in the label text is displayed.

Set ShowAccelChar to True to allow the label to display an underlined accelerator key value. When ShowAccelChar is True, any character preceded by an ampersand (&) appears underlined. To display an ampersand when ShowAccelChar is True, use two ampersands (&&) to stand for the single ampersand that is displayed.

Set ShowAccelChar to False to display the label text with all ampersands appearing as ampersands.

Example

This example uses two labels on a form. The first label has a caption with an accelerator character in it. The second label also includes an ampersand, but it does not appear as an accelerator character.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Label1.ShowAccelChar := True;
  Label1.Caption := 'An &Underlined character appears here';
  Label2.ShowAccelChar := False;
  Label2.Caption := 'An ampersand (&) appears here';
end;
```

See also

Caption property.

10.19.2.1.14 ShowHint

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property ShowHint: Boolean;
```

Description

© 1998-2006, UtilMind Solutions®

The ShowHint property determines whether the label attached to control displays a Help Hint when the mouse pointer rests momentarily on the label.

The Help Hint is the value of the Hint property, which is displayed in a box just beneath the control. Use ShowHint to determine whether a Help Hint appears for the control.

To enable Help Hint for a particular control, the application ShowHint property must be true and either the control's own ShowHint property must be true, or the control's ParentShowHint property must be true and its parent's ShowHint property must be true.

For example, imagine a check box within a group box. If the ShowHint property of the group box is true and the ParentShowHint property of the check box is true, but the ShowHint property of the check box is false, the check box still displays its Help Hint.

Changing the ShowHint value automatically sets the ParentShowHint property to false.

See also
Hint property.

10.19.2.1.15 Spacing

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

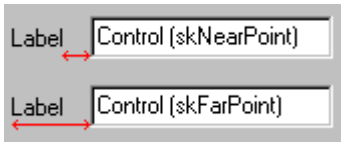
Declaration
```
property Spacing: Integer;
```

Description
The Spacing property determines the space (in pixels) between the control and attached label.

There is two spacing kinds which have an influence on distance between the control and label (see SpacingKind property).

When SpacingKind is `skNearPoint`, the Spacing property signifies the distance between control and nearest point of attached label.

When SpacingKind is `skFarPoint`, the Spacing property signigies the distance between control and far point of label.

Screenshot


See also
SpacingKind and Position properties.

10.19.2.1.16 SpacingKind

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration

© 1998-2006, UtilMind Solutions®

```
type
  TacAttachedLabelSpacingKind = (skNearPoint, skFarPoint);

property SpacingKind: TacAttachedLabelSpacingKind;
```

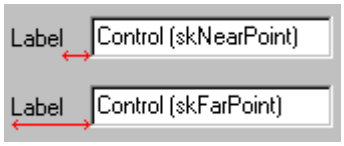
Description

The SpacingKind property specifies how to measure the spacing between the control and attached label.

There is two possible values for SpacingKind property:

Value	Meaning
skNearPoint	the Spacing property signifies the distance between control and nearest point of attached label;
skFarPoint	the Spacing property signigies the distance between control and far point of label.

Screenshot



See also

Spacing and Position properties.

10.19.2.1.17 Transparent

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property Transparent: Boolean;
```

Description

The Transparent property specifies whether controls that sit below the label on a form can be seen through the label.

Set Transparent to True to prevent the label from obscuring other controls on the form. For example, if the label is used to add text to a graphic, set Transparent to True so that the label does not stand out as a separate object.

Writing text so that the background is transparent is slower than writing text when Transparent is False. If the label is not obscuring a complicated image, performance can be improved by setting the background color of the label to match the object beneath it and setting Transparent to False.

See also

Color property.

10.19.2.1.18 Visible

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration

© 1998-2006, UtilMind Solutions®

```
property Visible: Boolean;
```

Description
The Visible property determines whether the label attached to the control appears on screen.

Use the Visible property to control the visibility of the control at runtime. If Visible is true, the control appears. If Visible is false, the control is not visible.

10.19.2.1.19 WordWrap

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property WordWrap: Boolean;
```

Description
The WordWrap property specifies whether the label text wraps when it is too long for the width of the label.

Set WordWrap to True to allow the label to display multiple line of text. When WordWrap is True, text that is too wide for the label control wraps at the right margin and continues in additional lines.

Set WordWrap to False to limit the label to a single line. When WordWrap is False, text that is too wide for the label appears truncated.

See also
Caption property.

10.19.2.2Images

Applies to
acImagesComboBox component.

Declaration
```
property Images: TImageList;
```

Description
The Images property specifies the images that can appear beside individual items of combo box.

If image list is not specified then the combo box will show just plain strings. The height of the combo box will be automatically adjusted accordingly to the height of images.

See also
Items property.

10.19.2.3Items

Applies to
acImagesComboBox component.

Declaration
```
property Items: TStringList;
```

Description
The Items property provides access to the list of items (strings) in the list portion of the combo box.

© 1998-2006, UtilMind Solutions®

Use Items property to specify the text and image index for portions of drop-down list. The image index should be specified as integer number and separated from the text of item by "|" character. For example, if the Items contains the line: '0|The Item', this means that item should display text "The Item" and use first image specified in the image list (with index = 0). If the image index is not specified it will show only text without image.

Read Items to access the list of items that appears in the combo box. Use the methods of Items to add, insert, delete and move items. Set the value of Items to copy the items from another string list.

Example

```
acImagesComboBox1.Items.Clear;
acImagesComboBox1.Items.Add('0|First item (imageindex=0)');
acImagesComboBox1.Items.Add('Item without image');
acImagesComboBox1.Items.Add('1|Another item (imageindex = 1)');
```

See also

Images property.

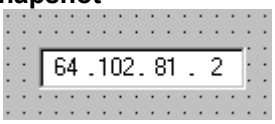
10.20 acIPEdit

10.20.1 TacIPEdit

Overview

The acIPEdit component is the control for editing the IP addresses. This is 100% native VCL component and contains a little bit more features than ActiveX "IPAddress". Also it contains the AttachedLabel structure which lets to operate with the label associated with the edit control and specify its behavior and position near the control.

Snapshot



You can specify and determinate the IP address as string (Text property), as integer value (IPLong property), or separately as four byte values (IP1, IP2, IP3, IP4 properties).

Additional / improved properties

Alignment	controls horizontal placement of the text within every section of the acIPEdit control;
AttachedLabel	controls the behavior of the label attached to the control and its position;
AutoSelect	controls whether the text in the IP section should be automatically selected when user moves to the section using arrow keys, activating input focus in the section of acIPEdit control;
IP1, IP2, IP3, IP4	determines or specifies the byte values in separate sections of IP address;
IPLong	represents the IP address as integer value;
Text	represents the IP address as text string.

See also

acEdit and acNumberEdit components.

© 1998-2006, UtilMind Solutions®

10.20.2 Properties

10.20.2.1Align

Applies to

acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acIPEdit and acMRUComboBox components *(in generally to ALL successors of TControl)*.

Declaration

```
type
  TAlign = (alNone, alTop, alBottom, alLeft, alRight, alClient);

property Align: TAlign;
```

Declaration

The Align property determines how the control aligns within its container (or parent control).

Use Align to align a control to the top, bottom, left, or right of a form or panel and have it remain there even if the size of the form, panel, or component that contains the control changes.

For example, to use a panel component with various controls on it as a tool palette, change the panel's Align value to alLeft. The value of alLeft for the Align property of the panel guarantees that the tool palette remains on the left side of the form and always equals the client height of the form.

The default value of Align is alNone, which means a control remains where it is positioned on a form or panel. Specify another value to align the control. Use one of these possible values:

Value	Meaning
alNone	The component remains where it was placed on the form or panel. This is the default value.
alTop	The component moves to the top of the form and resizes to fill the width of the form. The height of the component is not affected.
alBottom	The component moves to the bottom of the form and resizes to fill the width of the form. The height of the component is not affected.
alLeft	The component moves to the left side of the form and resizes to fill the height of the form. The width of the component is not affected.
alRight	The component moves to the right side of the form and resizes to fill the height of the form. The width of the component is not affected.
alClient	The component resizes to fill the client area of a form. If a component already occupies part of the client area, the component resizes to fit within the remaining client area.

Note

If Align is set to alClient, the control fills the entire client area so that it is impossible to select the parent form by clicking on it. In this case, select the parent by selecting the control on the form and pressing Esc, or by using the Object Inspector.

Any number of child components within a single parent can have the same Align value, in which case they stack up along the edge of the parent form or panel. The child controls stack up in z-order. To adjust the order in which the controls stack up, drag the controls into their desired positions.

© 1998-2006, UtilMind Solutions®

10.20.2.2Alignment

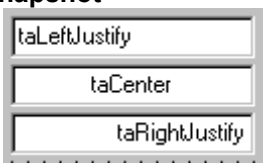
Applies to
acIPEdit, acEdit and acNumberEdit components.

Declaration
```
type
  TAlignment = (taLeftJustify, taRightJustify, taCenter);

property Alignment: TAlignment;
```

Description
The Alignment property controls horizontal placement of the text within every section of the acIPEdit control. Set Alignment to specify how the text of the edit control is aligned within the client area (ClientRect property) of this control. Alignment can be one of the following values:

Value	Meaning
taLeftJustify	The text appears lined up along the left edge of the edit control;
taRightJustify	The text appears lined up along the right edge of the edit control;
taCenter	The text is horizontally centered in the edit control.

Snapshot



10.20.2.3AttachedLabel

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components.

Unit
acAttachedLabel

Declaration
```
type
  TacAttachedLabelPosition = (lpAbove, lpBelow, lpLeft, lpRight);
  TacAttachedLabelSpacingKind = (skNearPoint, skFarPoint);
  TacAttachedLabel = class
    property Caption: TCaption;
    property Color: TColor default clBtnFace;
    property Cursor: TCursor default crDefault;
    property Enabled: Boolean;
    property FocusOnClick: Boolean default True;
    property Font: TFont;
    property Hint: String;
    property ParentColor: Boolean default True;
    property ParentFont: Boolean default True;
    property ParentShowHint: Boolean default True;
    property PopupMenu: TPopupMenu;
    property Position: TacAttachedLabelPosition default lpLeft;
    property ShowAccelChar: Boolean default True;
    property ShowHint: Boolean default False;
```

© 1998-2006, UtilMind Solutions®

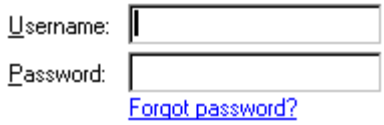
```
    property Spacing: Integer default 4;
    property SpacingKind: TacAttachedLabelSpacingKind default
skNearPoint;
    property Transparent: Boolean default False;
    property Visible: Boolean;
    property WordWrap: Boolean default False;
  end;
```

Description

The AttachedLabel structure lets you to operate with the label attached to the some side of control (at the left or right side, above or below). You can specify the behaviour of label, its position near the control, and accelerator key for the control (character in caption, after ampersand (&)).

The label always associated with the control. When user press the accelerator key (Alt + Character, specified after ampersand (&)), the input focus will be imediately switched to the labeled control.

Screenshot (the labels at the left side is attached to the edit control):



See also

OnLabelClick and OnLabelDblClick events.

10.20.2.3.1 Caption

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property Caption: TCaption;
```

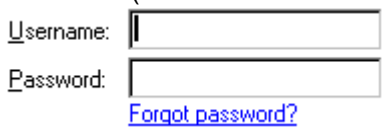
Description

The Caption property specified a text string which appers on label attached to control (attached label).

To underline a character in a Caption that labels a component, include an ampersand (&) before the character. This type of character is called an accelerator character. The user can then select the component by pressing Alt while typing the underlined character. To display an ampersand character in the caption, use two ampersands (&&).

The label always associated with the control. When user press the accelerator key (Alt + Character, specified after ampersand (&)), the input focus will be imediately switched to the labeled control.

Screenshot (the labels at the left side is attached to the edit control):



See also

ShowAccelChar, Color, Font, Transparent, Visible and WordWrap properties.

© 1998-2006, UtilMind Solutions®

10.20.2.3.2 Color

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property Color: TColor;
```

Description
The Color property specifies the background color for label attached to control. Use Color to read or change the background color of the control.

If a control's ParentColor property is true, then changing the Color property of the control's parent automatically changes the Color property of the control. When the value of the Color property is changed, the control's ParentColor property is automatically set to False.

If you don't want to show background — set Transparent property to True.

See also
ParentColor and Transparent properties.

10.20.2.3.3 Cursor

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property Cursor: TCursor;
```

Description
The Cursor property specifies the image used to represent the mouse pointer when it passes into the region covered by this label attached to control.

Change the value of Cursor to provide feedback to the user when the mouse pointer enters the control. The value of Cursor is the index of the cursor in the list of cursors maintained by the global variable, Screen. In addition to the built-in cursors provided by TScreen, applications can add custom cursors to the list.

See also
Additional Cursors

10.20.2.3.4 Enabled

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property Enabled: Boolean;
```

Description
The Enabled property controls whether the control responds to mouse, keyboard, and timer events.

Use Enabled to change the availability of the control to the user. To disable a control, set Enabled to

© 1998-2006, UtilMind Solutions®

false. Disabled controls appear dimmed. If Enabled is false, the control ignores mouse, keyboard, and timer events.

To re-enable a control, set Enabled to true. The control is no longer dimmed, and the user can use the control.

See also
Caption and Color properties.

10.20.2.3.5 FocusOnClick

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property FocusOnClick: Boolean; // True by default
```

Description
The FocusOnClick property controls whether the control, associated with the attached label, should accept focus when user clicks the label with mouse.

Set FocusOnClick to True if you want to switch focus to the control when user clicks the label, or set it to False otherwise.

Note
Additionally you can use OnLabelClick and OnLabelDblClick events of the control.

10.20.2.3.6 Font

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property Font: TFont;
```

Description
The Font property controls the attributes of text written on the attached label of control.

To change to a new font, specify a new TFont object. To modify a font, change the value of the Charset, Color, Height, Name, Pitch, Size, or Style of the TFont object.

See also
ParentFont property.

10.20.2.3.7 Hint

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property Hint: String;
```

Description
The Hint property contains the text string that can appear when the user moves the mouse over the

© 1998-2006, UtilMind Solutions®

label attached control.

If you are using the FormHelp component to show the context-sensitive help taken from secondary part of hint AND the Hint for attached label is not specified, the FormHelp will use the hint of main control.

See also
ShowHint and ParentShowHint properties;
acFormHelp and acFormHints components.

10.20.2.3.8 ParentColor

Enter topic text here.

10.20.2.3.9 ParentFont

Enter topic text here.

10.20.2.3.10 ParentShowHint

Enter topic text here.

10.20.2.3.11 PopupMenu

Enter topic text here.

10.20.2.3.12 Position

Enter topic text here.

10.20.2.3.13 ShowAccelChar

Enter topic text here.

10.20.2.3.14 ShowHint

Enter topic text here.

10.20.2.3.15 Spacing

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property Spacing: Integer;
```

Description
The Spacing property determines the space (in pixels) between the control and attached label.

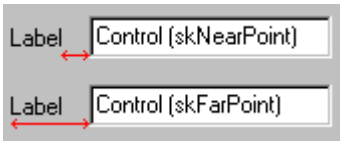
There is two spacing kinds which have an influence on distance between the control and label (see SpacingKind property).

When SpacingKind is `skNearPoint`, the Spacing property signifies the distance between control and nearest point of attached label.

When SpacingKind is `skFarPoint`, the Spacing property signigies the distance between control and far point of label.

Screenshot

© 1998-2006, UtilMind Solutions®



See also
SpacingKind and Position properties.

10.20.2.3.16 SpacingKind

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration

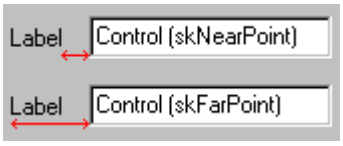
```
type
  TacAttachedLabelSpacingKind = (skNearPoint, skFarPoint);

property SpacingKind: TacAttachedLabelSpacingKind;
```

Description
The SpacingKind property specifies how to measure the spacing between the control and attached label.

There is two possible values for SpacingKind property:

Value	Meaning
skNearPoint	the Spacing property signifies the distance between control and nearest point of attached label;
skFarPoint	the Spacing property signigies the distance between control and far point of label.

Screenshot



See also
Spacing and Position properties.

10.20.2.3.17 Transparent

Enter topic text here.

10.20.2.3.18 Visible

Enter topic text here.

10.20.2.3.19 WordWrap

Enter topic text here.

10.20.2.4AutoSelect

Applies to
acIPEdit, acEdit and acNumberEdit components.

© 1998-2006, UtilMind Solutions®

Declaration

```
property AutoSelect: Boolean; // False by default
```

Description

The AutoSelect property controls whether the text in the IP section should be automatically selected when user moves to the section using arrow keys, activating input focus in the section of acIPEdit control.

When the AutoSelect is False, the cursor caret will be moved as in usual edit control, without selecting text in IP section, like in "IPAddress", ActiveX control.

When the AutoSelect is True and user press the arrow keys, activating focus in nearby section, the text in the IP section will be automatically selected.

10.20.2.5AutoSize

Applies to

acIPEdit component.

Declaration

```
property AutoSize: Boolean;
```

Description

AutoSize determines whether the width and height of the edit control automatically resizes to accommodate the text (IP address).

Use AutoSize to make the edit control adjust its size automatically so the client area accommodates the height of the text and width of IP sections. When AutoSize is False, the edit control has a fixed size. When AutoSize is True, the size of the control is readjusted whenever a change occurs that could affect the size of the control, such as a change to the font or border style.

10.20.2.6Color

Applies to

acEdit, acNumberEdit and acIPEdit components (in general, to all successors of TControl class).

Declaration

```
property Color: TColor;
```

Description

The Color property for acIPEdit component is the background color of edit control and every section of IP address. Use Color to read or change the background color of the control.

If a control's ParentColor property is True, then changing the Color property of the control's parent automatically changes the Color property of the control. When the value of the Color property is changed, the control's ParentColor property is automatically set to False.

See also

ColorDisabled property.

10.20.2.7ColorDisabled

Applies to

acIPEdit, acEdit and acNumberEdit components.

Declaration

```
property ColorDisabled: TColor; // clBtnFace by default
```

© 1998-2006, UtilMind Solutions®

Description
The ColorDisabled property for acIPEdit component is the background color of the control at disabled state (when Enabled is False). Use the ColorDisabled property to specify color for disabled edit control.

See also
Color property.

10.20.2.8IP1

Applies to
acIPEdit component.

Declaration
```
property IP1: Byte;
```

Description
The IP1 property determines or specifies the first section of IP address.

See also
IP1, IP2, IP3, IP4, IPLong, Text properties and OnChange event.

10.20.2.9IP2

Applies to
acIPEdit component.

Declaration
```
property IP2: Byte;
```

Description
The IP2 property determines or specifies the second section of IP address.

See also
IP1, IP2, IP3, IP4, IPLong, Text properties and OnChange event.

IP3
10.20.2.10

Applies to
acIPEdit component.

Declaration
```
property IP3: Byte;
```

Description
The IP3 property determines or specifies the third section of IP address.

See also
IP1, IP2, IP3, IP4, IPLong, Text properties and OnChange event.

IP4
10.20.2.11

Applies to
acIPEdit component.

© 1998-2006, UtilMind Solutions®

Declaration
```
property IP4: Byte;
```

Description
The IP4 property determines or specifies the fourth section of IP address.

See also
IP1, IP2, IP3, IP4, IPLong, Text properties and OnChange event.

IPLong
10.20.2.12

Applies to
acIPEdit component.

Declaration
```
property IPLong: LongInt;
```

Description
The IPLong property represents the IP address, specified by IP1, IP2, IP3 and IP4 properties, in integer format. Use the Text property to read or specify the IP address as integer value.

See also
IP1, IP2, IP3, IP4, Text properties and OnChange event.

Text
10.20.2.13

Applies to
acIPEdit component.

Declaration
```
property Text: String;
```

Description
The Text property represents the IP address, specified by IP1, IP2, IP3 and IP4 properties, in string format. Use the Text property to read or specify the IP address as string.

See also
IP1, IP2, IP3, IP4, IPLong properties and OnChange event.

10.20.3 Events

10.20.3.1 OnChange

Applies to
acIPEdit and acNumberEdit components.

Declaration
```
property OnChange: TNotifyEvent;
```

Description
The OnChange event occurs whenever the IP address value has been changed.

Write an OnChange event handler to take specific action after changing of the IP address represented by IP1, IP2, IP3, IP4, IPLong and Text properties. All these properties will already be updated to reflect any changes. This event provides the first opportunity to respond to modifications

© 1998-2006, UtilMind Solutions®

that the user types into the edit control.

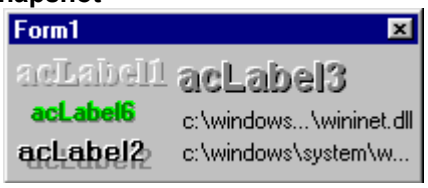
See also
IP1, IP2, IP3, IP4, IPLong and Text properties.

10.21 acLabel

10.21.1 TacLabel

Overview
The acLabel component lets you to create eye-catching captions for your application. This is improved label control which able to show the shadow and highlight effects, twinkle captions, and change the appearance when the mouse hovers or clicks the label.

Snapshot



Additional / improved properties

Blinking	set of properties to make the label text twinkle and configure the blinking intervals;
CursorClick	cursor image user clicks the label with mouse;
FontClick	font color and style for text when user clicks the label with mouse;
FontHover	font color and style for text when user moves mouse pointer over label;
Highlight	color, style and depth of highlight effect;
Shadow	color, style and depth of shadow effect;
Options	specifies optional behaivor of text. For example, the label can looks like the path to some disk object (with an ellipsis (…) in the middle of label if the text is longer than width);
WordWrap	whether the text should wraps to the next line if it doesn't fits to the control's bounds.

New events

OnMouseEnter	occurs when mouse pointer hovers the label;
OnMouseLeave	occurs when mouse pointer leaves the label.

See also
acURLLabel and acRichLabel components.

10.21.2 Properties

10.21.2.1 Blinking

Applies to
acLabel and acURLLabel components.

Declaration

```
type
  TacLabelBlinking = class
  published
    property Enabled: Boolean;
    property DelayVisible: Integer; // default 700
```

© 1998-2006, UtilMind Solutions®

```
    property DelayInvisible: Integer; // default 400
  end;

property Blinking: TacLabelBlinking;
```

Description
The Blinking structure

See also

10.21.2.1.1 DelayInvisible

Applies to
acLabel and acURLLabel components as subproperty of Blinking structure.

Declaration
```
property DelayInvisible: Integer;
```

Description

See also
DelayVisible and Enabled properties.

10.21.2.1.2 DelayVisible

Applies to
acLabel and acURLLabel components as subproperty of Blinking structure.

Declaration
```
property DelayVisible: Integer;
```

Description

See also
DelayVisible and Enabled properties.

10.21.2.1.3 Enabled

Applies to
acLabel and acURLLabel components as subproperty of Blinking, Highlight and Shadow structures (and for all visible components in general).

Declaration
```
property Enabled: Boolean;
```

Description

10.21.2.2CursorClick

Applies to
acLabel and acURLLabel components.

Declaration
```
property CursorClick: TCursor;
```

© 1998-2006, UtilMind Solutions®

Description
The CursorClick property specifies the image for cursror when user clicks the label with mouse.

10.21.2.3FontClick

Applies to
acLabel and acURLLabel components.

Declaration

```
type
  TStyleFont = class
  published
    property Color: TColor;
    property Style: TFontStyles;
    property Used: Boolean;
  end;

property FontClick: TStyleFont;
```

Description
The FontClick structure specifies

See also

10.21.2.3.1 Color

© 1998-2006, UtilMind Solutions®

10.21.2.3.2 Style

© 1998-2006, UtilMind Solutions®

10.21.2.3.3 Used

Applies to
acLabel and acURLLabel components as subproperty of FontClick and FontHover structures.

Declaration
```
property Used: Boolean;
```

Description

See also

10.21.2.4FontHover

Applies to
acLabel and acURLLabel components.

Declaration
```
type
  TStyleFont = class
  published
    property Color: TColor;
    property Style: TFontStyles;
    property Used: Boolean;
  end;

property FontClick: TStyleFont;
```

Description

See also

10.21.2.4.1 Color

© 1998-2006, UtilMind Solutions®

10.21.2.4.2 Style

© 1998-2006, UtilMind Solutions®

10.21.2.4.3 Used

Applies to
acLabel and acURLLabel components as subproperty of FontHover and FontClick structures.

Declaration
```
property Used: Boolean;
```

Description

See also

10.21.2.5Highlight

Applies to
acLabel and acURLLabel components.

Declaration
```
type
  TacLabelEffectDirection = (edUp, edDown, edLeft, edRight,
                             edUpLeft, edUpRight,
                             edDownLeft, edDownRight);
  TacLabelEffect = class
  published
    property Enabled: Boolean;
    property Color: TColor;
    property ColorHover: TColor;
    property ColorClick: TColor;
    property Depth: Byte;
    property Direction: TacLabelEffectDirection;
  end;

property Highlight: TacLabelEffect;
```

Description

See also
Shadow structure.

10.21.2.5.1 Color

Applies to
acLabel and acURLLabel components as subproperty of Highlight, Shadow, FontClick, FontHover structures (and for all visible components in general).

Declaration
```
property Color: TColor;
```

Description

10.21.2.5.2 ColorClick

Applies to
acLabel and acURLLabel as subproperty of Highlight and Shadow structures.

© 1998-2006, UtilMind Solutions®

Declaration

```
property ColorClick: TColor;
```

Description

See also

10.21.2.5.3 ColorHover

Applies to

acLabel and acURLLabel as subproperty of Highlight and Shadow structures.

Declaration

```
property ColorHover: TColor;
```

Description

See also

10.21.2.5.4 Depth

Applies to

acLabel and acURLLabel as subproperty of Highlight and Shadow structures.

Declaration

```
property Depth: Byte;
```

Description

See also

10.21.2.5.5 Direction

Applies to

acLabel and acURLLabel as subproperty of Highlight and Shadow structures.

Declaration

```
type
  TacLabelEffectDirection = (edUp, edDown, edLeft, edRight,
                             edUpLeft, edUpRight,
                             edDownLeft, edDownRight);

property Direction: TacLabelEffectDirection;
```

Description

See also

10.21.2.5.6 Enabled

Applies to

acLabel and acURLLabel components as subproperty of Highlight, Shadow and Blinking structures (and for all visible components in general).

© 1998-2006, UtilMind Solutions®

Declaration

```
property Enabled: Boolean;
```

Description

10.21.2.6Shadow

Applies to

acLabel and acURLLabel components.

Declaration

```
type
  TacLabelEffectDirection = (edUp, edDown, edLeft, edRight,
                             edUpLeft, edUpRight,
                             edDownLeft, edDownRight);
  TacLabelEffect = class
  published
    property Enabled: Boolean;
    property Color: TColor;
    property ColorHover: TColor;
    property ColorClick: TColor;
    property Depth: Byte;
    property Direction: TacLabelEffectDirection;
  end;

property Shadow: TacLabelEffect;
```

Description

See also

Highlight structure.

10.21.2.6.1 Color

Applies to

acLabel and acURLLabel components as subproperty of Shadow, Highlight, FontClick, FontHover structures (and for all visible components in general).

Declaration

```
property Color: TColor;
```

Description

10.21.2.6.2 ColorClick

Applies to

acLabel and acURLLabel as subproperty of Shadow and Highlight structures.

Declaration

```
property ColorClick: TColor;
```

Description

See also

© 1998-2006, UtilMind Solutions®

10.21.2.6.3 ColorHover

Applies to
acLabel and acURLLabel as subproperty of Shadow and Highlight structures.

Declaration
```
property ColorHover: TColor;
```

Description

See also

10.21.2.6.4 Depth

Applies to
acLabel and acURLLabel as subproperty of Shadow and Highlight structures.

Declaration
```
property Depth: Byte;
```

Description

See also

10.21.2.6.5 Direction

Applies to
acLabel and acURLLabel as subproperty of Shadow and Highlight structures.

Declaration
```
type
  TacLabelEffectDirection = (edUp, edDown, edLeft, edRight,
                             edUpLeft, edUpRight,
                             edDownLeft, edDownRight);

property Direction: TacLabelEffectDirection;
```

Description

See also

10.21.2.6.6 Enabled

Applies to
acLabel and acURLLabel components as subproperty of Shadow, Highlight and Blinking structures (and for all visible components in general).

Declaration
```
property Enabled: Boolean;
```

Description

© 1998-2006, UtilMind Solutions®

10.21.2.7Options

Applies to
acLabel and acURLLabel components.

Declaration
```
type
  TacLabelOptions = set of (loPathEllipsis,
                            loEndEllipsis,
                            loEditControl);

property Options: TacLabelOptions;
```

Description
The Options property controls the label's behavior and how label should display the caption.

There is possible options:

Value	Meaning
loPathEllipsis	shows the caption as path to some disk object. For example, if label's text doesn;
loEndEllipsis	s
loEditControl	s

See also
WordWrap property.

10.21.2.8WordWrap

Applies to
acLabel, acURLLabel, acRichLabel, acButton, acBitBtn, acCheckBox and acRadioButton components.

Declaration
```
property WordWrap: Boolean;
```

Description
The WordWrap property specifies whether the text string wraps when it is too long for the width of the label. When WordWrap is True, caption can be multiline, and single line when WordWrap is False.

See also
Options property.

10.21.3 Events

10.21.3.1OnMouseEnter

Applies to
acLabel, acURLLabel, acRichLabel, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acMRUComboBox, acHeader, acHeaderPanel, acTreeView, acListView components.

Declaration
```
property OnMouseEnter: TNotifyEvent;
```

Description

© 1998-2006, UtilMind Solutions®

The OnMouseEnter occurs when mouse pointer hovers the label.

See also
OnMouseLeave event.

10.21.3.2 OnMouseLeave

Applies to
acLabel, acURLLabel, acRichLabel, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acMRUComboBox, acHeader, acHeaderPanel, acTreeView, acListView components.

Declaration
```
property OnMouseLeave: TNotifyEvent;
```

Description
The OnMouseLeave occurs when mouse pointer leaves the label.

See also
OnMouseEnter event.

10.22 acLabeledComboBox

10.22.1 TacLabeledComboBox

Overview
The acLabeledComboBox is the ComboBox control with the label attached to control.

The label always associated with the control. When user press the accelerator key (Alt + Character, specified after ampersand (&)), the input focus will be imediately switched to the labeled control. The input focus also will be moved to the control if user clicks the label with mouse.

Snapshot

How to use?
Just drop acLabeledComboBox onto your form, and open AttachedLabel property in the Object Inspector to specify the appearance and behaviours of the label attached to control.

To specify the text for attached label — use AttachedLabel.Caption property. To displace the label to another position (above, below or at the right side of control) — specify Position property.

See also (other controls with attached label)
acEdit, acNumberEdit, acIPEdit;
acMRUComboBox, acImagesComboBox, acTreeComboBox.

10.22.2 Properties

10.22.2.1 AttachedLabel

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components.

© 1998-2006, UtilMind Solutions®

Unit
acAttachedLabel

Declaration
```
type
  TacAttachedLabelPosition = (lpAbove, lpBelow, lpLeft, lpRight);
  TacAttachedLabelSpacingKind = (skNearPoint, skFarPoint);
  TacAttachedLabel = class
    property Caption: TCaption;
    property Color: TColor default clBtnFace;
    property Cursor: TCursor default crDefault;
    property Enabled: Boolean;
    property FocusOnClick: Boolean default True;
    property Font: TFont;
    property Hint: String;
    property ParentColor: Boolean default True;
    property ParentFont: Boolean default True;
    property ParentShowHint: Boolean default True;
    property PopupMenu: TPopupMenu;
    property Position: TacAttachedLabelPosition default lpLeft;
    property ShowAccelChar: Boolean default True;
    property ShowHint: Boolean default False;
    property Spacing: Integer default 4;
    property SpacingKind: TacAttachedLabelSpacingKind default
skNearPoint;
    property Transparent: Boolean default False;
    property Visible: Boolean;
    property WordWrap: Boolean default False;
  end;
```

Description
The AttachedLabel structure lets you to operate with the label attached to the some side of control (at the left or right side, above or below). You can specify the behaviour of label, its position near the control, and accelerator key for the control (character in caption, after ampersand (&)).

The label always associated with the control. When user press the accelerator key (Alt + Character, specified after ampersand (&)), the input focus will be imediately switched to the labeled control. The input focus also will be moved to the control if user clicks the label with mouse.

Screenshot (the labels at the left side is attached to the edit control):

See also
OnLabelClick and OnLabelDblClick events.

10.22.2.1.1 Caption

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property Caption: TCaption;
```

© 1998-2006, UtilMind Solutions®

Description

The Caption property specified a text string which appers on label attached to control (attached label).

To underline a character in a Caption that labels a component, include an ampersand (&) before the character. This type of character is called an accelerator character. The user can then select the component by pressing Alt while typing the underlined character. To display an ampersand character in the caption, use two ampersands (&&).

The label always associated with the control. When user press the accelerator key (Alt + Character, specified after ampersand (&)), the input focus will be imediately switched to the labeled control.

Screenshot (the labels at the left side is attached to the edit control):

See also

ShowAccelChar, Color, Font, Transparent, Visible and WordWrap properties.

10.22.2.1.2 Color

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property Color: TColor;
```

Description

The Color property specifies the background color for label attached to control. Use Color to read or change the background color of the control.

If a control's ParentColor property is true, then changing the Color property of the control's parent automatically changes the Color property of the control. When the value of the Color property is changed, the control's ParentColor property is automatically set to False.

If you don't want to show background — set Transparent property to True.

See also

ParentColor and Transparent properties.

10.22.2.1.3 Cursor

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property Cursor: TCursor;
```

Description

The Cursor property specifies the image used to represent the mouse pointer when it passes into

© 1998-2006, UtilMind Solutions®

the region covered by this label attached to control.

Change the value of Cursor to provide feedback to the user when the mouse pointer enters the control. The value of Cursor is the index of the cursor in the list of cursors maintained by the global variable, Screen. In addition to the built-in cursors provided by TScreen, applications can add custom cursors to the list.

See also
Additional Cursors

10.22.2.1.4 Enabled

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property Enabled: Boolean;
```

Description
The Enabled property controls whether the control responds to mouse, keyboard, and timer events.

Use Enabled to change the availability of the control to the user. To disable a control, set Enabled to false. Disabled controls appear dimmed. If Enabled is false, the control ignores mouse, keyboard, and timer events.

To re-enable a control, set Enabled to true. The control is no longer dimmed, and the user can use the control.

See also
Caption and Color properties.

10.22.2.1.5 FocusOnClick

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property FocusOnClick: Boolean; // True by default
```

Description
The FocusOnClick property controls whether the control, associated with the attached label, should accept focus when user clicks the label with mouse.

Set FocusOnClick to True if you want to switch focus to the control when user clicks the label, or set it to False otherwise.

Note
Additionally you can use OnLabelClick and OnLabelDblClick events of the control.

10.22.2.1.6 Font

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

© 1998-2006, UtilMind Solutions®

Declaration

```
property Font: TFont;
```

Description

The Font property controls the attributes of text written on the attached label of control.

To change to a new font, specify a new TFont object. To modify a font, change the value of the Charset, Color, Height, Name, Pitch, Size, or Style of the TFont object.

See also

ParentFont property.

10.22.2.1.7 Hint

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property Hint: String;
```

Description

The Hint property contains the text string that can appear when the user moves the mouse over the label attached control.

If you are using the FormHelp component to show the context-sensitive help taken from secondary part of hint AND the Hint for attached label is not specified, the FormHelp will use the hint of main control.

See also

ShowHint and ParentShowHint properties;
acFormHelp and acFormHints components.

10.22.2.1.8 ParentColor

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property ParentColor: Boolean;
```

Description

The ParentColor determines where an attached label looks for its color information.

To have a control use the same color as its parent control, set ParentColor to true If ParentColor is false the control uses its own Color property.

Set ParentColor to true for all controls in order to ensure that all the controls on a form have a uniform appearance. For example, if ParentColor is true for all controls in a form, changing the background color of the form to gray causes all the controls on the form to also have a gray background.

When the value of a label's Color property changes, ParentColor becomes false automatically.

See also

© 1998-2006, UtilMind Solutions®

Color and Transparent properties.

10.22.2.1.9 ParentFont

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property ParentFont: Boolean;
```

Description
The ParentFont property determines where a control looks for its font information.

To have a control use the same font as its parent control, set ParentFont to true If ParentFont is false the label uses its own Font property.

Set ParentFont to true for all controls in order to ensure that all the controls on a form have a uniform appearance. For example, if ParentFont is true for all controls in a form, changing the form's Font property to 12-point Courier causes all controls on the form to use that font.

When the value of a control's Font property changes, ParentFont becomes false automatically.

When ParentFont is true for a form, the form uses the default font.

See also
Font property.

10.22.2.1.10 ParentShowHint

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property ParentShowHint: Boolean;
```

Description
The ParentShowHint property determines where an attached label looks to find out if its Help Hint should be shown.

Use ParentShowHint to ensure that all the controls on a form either uniformly show their Help Hints or uniformly do not show them.

If ParentShowHint is true, the label uses the ShowHint property value of its parent. If ParentShowHint is false, the label uses the value of its own ShowHint property.

To provide Help Hints for only selected controls on a form, set the ShowHint property for those controls that should have Help Hints to true, and ParentShowHint becomes false automatically.

Note
Enable or disable all Help Hints for the entire application using the ShowHint property of the application object.

See also
ShowHint and Hint properties;
acFormHelp and acFormHints components.

© 1998-2006, UtilMind Solutions®

10.22.2.1.11 PopupMenu

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property PopupMenu: TPopupMenu;
```

Description
The PopupMenu property Identifies the pop-up menu associated with the label attached to control.

Assign a value to PopupMenu to make a pop-up menu appear when the user selects the label and clicks the right mouse button. If the TPopupMenu's AutoPopup property is True, the pop-up menu appears automatically. If the menu's AutoPopup property is False, display the menu with a call to its Popup method from the control's OnContextPopup event handler.

10.22.2.1.12 Position

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
type
  TacAttachedLabelPosition = (lpAbove, lpBelow, lpLeft, lpRight);

property Position: TacAttachedLabelPosition;
```

Description
The Position property of the attached label specifies the layout of the label near the control.

You can make the label visible above, below, at the left or the right side of control.

See also
Spacing and SpacingKind properties.

10.22.2.1.13 ShowAccelChar

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property ShowAccelChar: Boolean;
```

Description
The ShowAccelChar property determines how an ampersand in the label text is displayed.

Set ShowAccelChar to True to allow the label to display an underlined accelerator key value. When ShowAccelChar is True, any character preceded by an ampersand (&) appears underlined. To display an ampersand when ShowAccelChar is True, use two ampersands (&&) to stand for the single ampersand that is displayed.

Set ShowAccelChar to False to display the label text with all ampersands appearing as ampersands.

© 1998-2006, UtilMind Solutions®

Example

This example uses two labels on a form. The first label has a caption with an accelerator character in it. The second label also includes an ampersand, but it does not appear as an accelerator character.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Label1.ShowAccelChar := True;
  Label1.Caption := 'An &Underlined character appears here';
  Label2.ShowAccelChar := False;
  Label2.Caption := 'An ampersand (&) appears here';
end;
```

See also

Caption property.

10.22.2.1.14 ShowHint

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property ShowHint: Boolean;
```

Description

The ShowHint property determines whether the label attached to control displays a Help Hint when the mouse pointer rests momentarily on the label.

The Help Hint is the value of the Hint property, which is displayed in a box just beneath the control. Use ShowHint to determine whether a Help Hint appears for the control.

To enable Help Hint for a particular control, the application ShowHint property must be true and either the control's own ShowHint property must be true, or the control's ParentShowHint property must be true and its parent's ShowHint property must be true.

For example, imagine a check box within a group box. If the ShowHint property of the group box is true and the ParentShowHint property of the check box is true, but the ShowHint property of the check box is false, the check box still displays its Help Hint.

Changing the ShowHint value automatically sets the ParentShowHint property to false.

See also

Hint and ParentShowHint properties.

10.22.2.1.15 Spacing

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property Spacing: Integer;
```

Description

The Spacing property determines the space (in pixels) between the control and attached label.

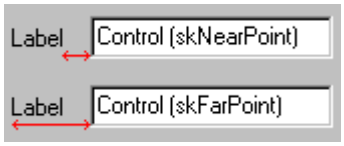
© 1998-2006, UtilMind Solutions®

There is two spacing kinds which have an influence on distance between the control and label (see SpacingKind property).

When SpacingKind is `skNearPoint`, the Spacing property signifies the distance between control and nearest point of attached label.

When SpacingKind is `skFarPoint`, the Spacing property signigies the distance between control and far point of label.

Screenshot



See also

SpacingKind and Position properties.

10.22.2.1.16 SpacingKind

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
type
  TacAttachedLabelSpacingKind = (skNearPoint, skFarPoint);

property SpacingKind: TacAttachedLabelSpacingKind;
```

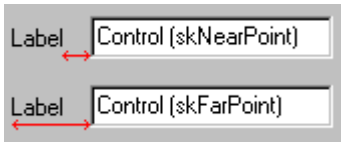
Description

The SpacingKind property specifies how to measure the spacing between the control and attached label.

There is two possible values for SpacingKind property:

Value	Meaning
skNearPoint	the Spacing property signifies the distance between control and nearest point of attached label;
skFarPoint	the Spacing property signigies the distance between control and far point of label.

Screenshot



See also

Spacing and Position properties.

10.22.2.1.17 Transparent

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

© 1998-2006, UtilMind Solutions®

Declaration

```
property Transparent: Boolean;
```

Description

The Transparent property specifies whether controls that sit below the label on a form can be seen through the label.

Set Transparent to True to prevent the label from obscuring other controls on the form. For example, if the label is used to add text to a graphic, set Transparent to True so that the label does not stand out as a separate object.

Writing text so that the background is transparent is slower than writing text when Transparent is False. If the label is not obscuring a complicated image, performance can be improved by setting the background color of the label to match the object beneath it and setting Transparent to False.

See also

Color property.

10.22.2.1.18 Visible

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property Visible: Boolean;
```

Description

The Visible property determines whether the label attached to the control appears on screen.

Use the Visible property to control the visibility of the control at runtime. If Visible is true, the control appears. If Visible is false, the control is not visible.

10.22.2.1.19 WordWrap

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property WordWrap: Boolean;
```

Description

The WordWrap property specifies whether the label text wraps when it is too long for the width of the label.

Set WordWrap to True to allow the label to display multiple line of text. When WordWrap is True, text that is too wide for the label control wraps at the right margin and continues in additional lines.

Set WordWrap to False to limit the label to a single line. When WordWrap is False, text that is too wide for the label appears truncated.

See also

Caption property.

© 1998-2006, UtilMind Solutions®

10.22.3 Methods

10.22.3.1SetSelection

Applies to
acEdit, acNumberEdit, acMRUComboBox, acLabeledComboBox and acTreeComboBox components.

Declaration
```
procedure SetSelection(SelStart: Integer = 0; SelEnd: Integer = -1);
```

Description
The SetSelection method used to select a range of characters in an edit control.

SelStart parameter specifies the starting character position of the selection.

SelEnd parameter specifies the ending character position of the selection.

Remarks
If the *SelStart* parameter is 0 and the *SelEnd* parameter is -1, all the text in the edit control is selected. If *SelStart* is -1, any current selection is removed. The caret is placed at the end of the selection indicated by the greater of the two values *SelEnd* and *SelStart*.

10.22.4 Events

10.22.4.1OnLabelClick

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and acTreeComboBox components.

Declaration
```
property OnLabelClick: TNotifyEvent;
```

Description
The OnLabelClick event occurs when user clicks the label attached to the control with a mouse.

To make the label appear near the edit control — set AttachedLabel.Visible to True, or to False if you'd like to hide the attached label.

See also
AttachedLabel structure.

10.22.4.2OnLabelDblClick

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and acTreeComboBox components.

Declaration
```
property OnLabelDblClick: TNotifyEvent;
```

Description
The OnLabelDblClick event occurs when user double clicks the label attached to the control with a mouse.

To make the label appear near the edit control — set AttachedLabel.Visible to True, or to False if

© 1998-2006, UtilMind Solutions®

you'd like to hide the attached label.

See also
AttachedLabel structure.

10.22.4.3OnMouseEnter

Applies to
acLabel, acURLLabel, acRichLabel, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox, acTreeComboBox, acHeader, acHeaderPanel, acTreeView and acListView components.

Declaration
```
property OnMouseEnter: TNotifyEvent;
```

Description
The OnMouseEnter occurs when mouse pointer hover the edit control.

See also
OnMouseLeave event.

10.22.4.4OnMouseLeave

Applies to
acLabel, acURLLabel, acRichLabel, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox, acTreeComboBox, acHeader, acHeaderPanel, acTreeView and acListView components.

Declaration
```
property OnMouseLeave: TNotifyEvent;
```

Description
The OnMouseLeave occurs when mouse pointer leave the edit control.

See also
OnMouseEnter event.

10.23 acLinksPanel

10.23.1 TacLinksPanel

Overview
The acLinksPanel is the panel with links (labels) and the small icons at the left of labels. You can specify the font and image for each label and hook the event when user clicks the particular label.

How to use?

See also

10.23.2 Properties

10.23.2.1FontClick

Applies to
acLinksPanel component.

© 1998-2006, UtilMind Solutions®

Declaration

Description

10.23.2.2FontHover

Applies to
acLinksPanel component.

Declaration

Description

10.23.2.3ImageList

Applies to
acLinksPanel component.

Declaration

Description

10.23.2.4ImageSpacing

Applies to
acLinksPanel component.

Declaration

Description

10.23.2.5Items

Applies to
acLinksPanel component.

Declaration

Description

10.23.2.6LinksProperties

Applies to
acLinksPanel component.

Declaration

© 1998-2006, UtilMind Solutions®

Description

10.23.2.7MarginLeft

Applies to
acLinksPanel component.

Declaration

Description

10.23.2.8MarginTop

Applies to
acLinksPanel component.

Declaration

Description

10.23.2.9Spacing

Applies to
acLinksPanel component.

Declaration
```
property Spacing: Byte;
```

Description
The Spacing property

10.23.3 Events

10.23.3.1OnLinkClick

Applies to
acLinksPanel component.

Declaration
```
type
  TacLinkClickEvent = procedure(Sender: TObject; LinkIndex: Integer;
LinkItem: TacLinksPanelItem) of object;

property OnLinkClick: TacLinkClickEvent;
```

Description
The OnLinkClick event occurs when user clicks the particular label (link) with mouse.

Write the OnLinkClick event hander to perform some specific action when user clicks the label or its image. The *LinkIndex* parameter specifies the index of the label in the Items collection (0 — first, 1 — second, etc).

© 1998-2006, UtilMind Solutions®

The *LinkItem* property provides an access to the clicked label item within Items collection. You can modify the properties of each LinkItem, if necessary, accessing the Items.

See also
Items property.

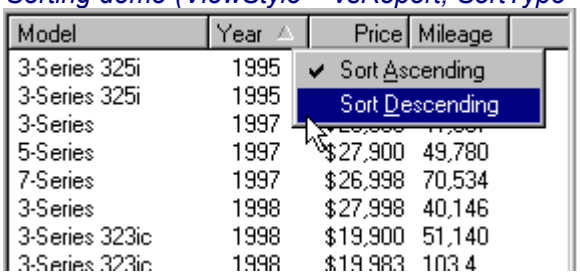
10.24 acListView

10.24.1 TacListView

Overview
The acListView component is the advanced ListView which can automatically sort columns, displaying the arrow-style sorting mark, which represents the sorting direction (like in the MS-Outlook).

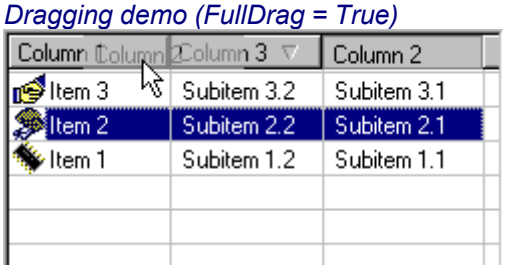
The acListView sorts items automatically and does not require any additional programming. However, the custom sorting are still supported. The acListView recognizes the format of list items and can automatically sort the **text**, **numerical** and **date/time** values when the sorting rules or any list item changes.

Sorting demo (ViewStyle = vsReport, SortType = stData)



Contains a lot of additional features, such like popup menus for the header columns, tooltips for the list items which doesn't fits the column size (complete customization of tooltip appearance and placement), custom cursors for various non-client parts of ListView.

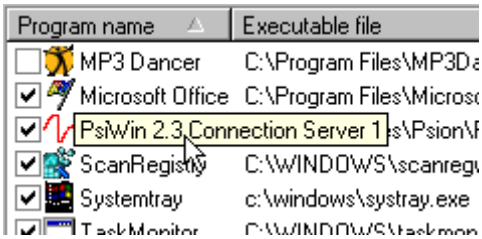
Dragging demo (FullDrag = True)



If you're still using old versions of Delphi / C++ Builder 2 or 3 you will also pleasantly surprised that the acListView supports features which become available in higher Delphi versions. This is grid lines, row selecting, flat scroll bars, hot tracking and hot track styles and so forth…

Tooltips demo (the line does not fits the column size)

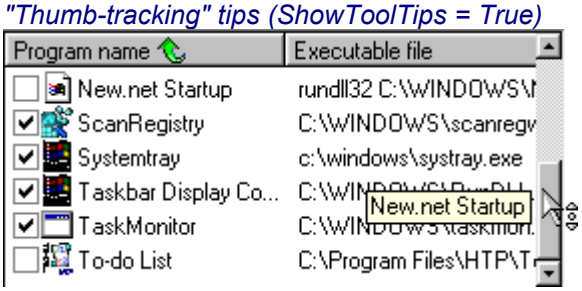
© 1998-2006, UtilMind Solutions®



new! acListView supports custom multiline tooltips for list items. Write OnToolTip event handler to specify tooltips. Also was implemented so-called "thumb-tracking" tips for vertical scroll bar. When user drags the thumb, the acListView shows the hint with caption of top item of current sorted column (see ShowScrollTips property).

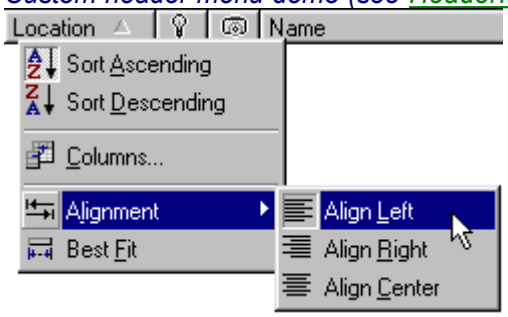
Multiline tooltips

"Thumb-tracking" tips (ShowToolTips = True)



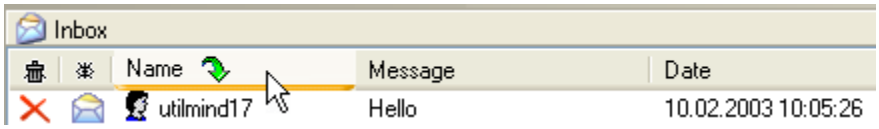
The acListView is able to automatically save and restore sorting rules and widths of the list columns on exit / startup of your program. You don't need to write any code and just select the registry key where you would like to store settings and set Enabled property of the RegistrySaver structure to True.

Custom header menu demo (see HeaderMenu property)



OnCustomDrawSubItemEx event demo

Windows XP look



In addition, the acListView contains some bugfixes of regular TListView. Was corrected a common control bug which appears when the color scheme of the system is changed. The dragging process (when user drags the columns) displayed more properly.

© 1998-2006, UtilMind Solutions®

new! The acListView automatically copies the text from selected rows/columns to clipboard when user press Ctrl+C or Ctrl+Ins keys. Alternatively you can use CopySelectionToClipboard method.

Advanced / Improved properties

Cursors	specifies custom cursor images for various parts of list view (i.e: border, divider between header sections, header, vertical and horizontal scroll bars);
EmptyListMessage	specifies the text for special label which can be visible when the list is empty (there are no items in the view);
HeaderFont	specifies the custom font for the text on the column header;
HeaderMenu	specifies custom popup menu for column header;
HeaderStyle	specifies the style of header sections (buttons or flat);
RegistrySaver	structure for saving / restoring listview's settings to the registry;
ShowEmptyListMessage	determines whether the special text label should be visible when the list is empty ((there are no items in the view);
ShowHeaderMenu	enables or disables popup menu for column headers;
ShowScrollTips	enables or disables the hints on dragging the thumb of scrollbar;
ShowSortMark	shows or hides sort mark (small triangle) on header sections;
ShowToolTips	enables or disables the tooltips for items with long text;
SortColumn	specifies the column number which must be sorted;
SortDirection	specifies the sorting direction;
SortMarkAlign	controls appearance of the sort mark on header section;
SortMarkKind	controls the style of the sort mark;
ToolTipOptions	some preferences for tooltips.

New / improved methods

BestFit	adjusts the column widths (of report-style list-view) thus the text of all subitems will fit to the widths of columns.
CopySelectionToClipboard	copies all text from selected rows/columns to clipboard;
GetColumnAt	returns the column number by X (mouse) position;
GetItemAt	returns list item from X/Y position;
ExportToTXT	saves all items and column names to TXT file;
ExportToHTML	saves all items and column names to simple HTML file;
SetColumnOrders	sets the orders of header items of the listview, or copies the orders from another listview.

New events

OnColumnDragged	occurs when user reorders the columns using a mouse;
OnColumnResizeBegin	occurs when user about resize the column. Allows to disallow resizing;
OnColumnResizeEnd	occurs when user finished resizing the column;
OnCustomDrawSubItemEx	a bug-free replacement of standard event used when a subitem must be rendered in the list-view;
OnDrawHeaderSection	event handler for custom drawing of the header section;
OnMouseEnter	occurs when mouse hovers the control;
OnMouseLeave	occurs when mouse leaves the control;
OnSettingsChanged	occurs when system settings changes (i.e: you need to refresh some list values which appearance depends on regional settings);
OnToolTip	occurs when the tooltip is about to be displayed. Write this handler to specify custom text for tooltip.

See also

acDBListView and acTreeView components.

© 1998-2006, UtilMind Solutions®

10.24.2 Properties

10.24.2.1CheckOnClick

Applies to
acListView and acDBListView components.

Declaration
```
property CheckOnClick: Boolean;
```

Description
The CheckOnClick property useful only for ListView with checkboxes (at the left side of control, Checkboxes = True), to specify that checkboxes should be checked and unchecked when user clicks the items with mouse.

See also
Checkboxes property and OnClick method;
OnCheckBoxClick event.

10.24.2.2Cursors

Applies to
acListView and acTreeView components.

Declaration
```
type
  TacListViewCursors = class
  published
    property Border: TCursor;
    property Divider: TCursor;
    property Header: TCursor;
    property HotCursor: TCursor;
    property ScrollHorz: TCursor;
    property ScrollVert: TCursor;
  end;

property Cursors: TacListViewCursors;
```

Description
Use Cursors structure to specify custom cursor images for some non-client parts of acListView control. You can specify cursor for border of control, header, columns divider, horizontal and vertical scrollbars.

See also
acAppCursors component and Additional cursors in the IDE.

10.24.2.2.1 Border

Applies to
acListView and acTreeView components as subproperty of Cursors structure.

Declaration
```
property Border: TCursor;
```

Description
The Border property specifies the image for mouse pointer, when it passes over border of acListView control.

© 1998-2006, UtilMind Solutions®

See also
Header, Divider, ScollHorz and ScrollVert properties.

10.24.2.2.2 Divider

Applies to
acListView component as subproperty of Cursors structure.

Declaration
```
property Divider: TCursor;
```

Description
The Divider property specifies the image for mouse pointer, when it passes over the area which separates the header columns.

See also
Border, Header, ScollHorz and ScrollVert properties.

10.24.2.2.3 Header

Applies to
acListView component as subproperty of Cursors structure.

Declaration
```
property Header: TCursor;
```

Description
The Header property specifies the image for mouse pointer, when it passes over the header of acListView control (when ViewStyle = vsReport).

See also
Border, Divider, ScollHorz and ScrollVert properties.

10.24.2.2.4 HotCursor

Applies to
acListView component as subproperty of Cursors structure.

Declaration
```
property HotCursor: TCursor;
```

Description
The HotCursor property specifies the image for mouse pointer, when it passes over the tracking list item (when HotTrack = True).

See also
Border, Divider, Header, ScollHorz and ScrollVert properties.

10.24.2.2.5 ScrollHorz

Applies to
acListView and acTreeView components as subproperty of Cursors structure.

Declaration
```
property ScrollHorz: TCursor;
```

Description

© 1998-2006, UtilMind Solutions®

The ScrollHorz property specifies the image for mouse pointer, when it passes over horizontal scrollbar of acListView control.

See also
ScrollVert, Border, Header and Divider properties.

10.24.2.2.6 ScrollVert

Applies to
acListView and acTreeView components as subproperty of Cursors structure.

Declaration
```
property ScrollVert: TCursor;
```

Description
The ScrollVert property specifies the image for mouse pointer, when it passes over vertical scrollbar of acListView control.

See also
ScollHorz, Border, Header and Divider properties.

10.24.2.3EmptyListMessage

Applies to
acListView and acDBListView components.

Declaration
```
property EmptyListMessage: String; // Default text is 'There are no
items in this view.'
```

Description
The EmptyListMessage property specifies the text for label which can be visible if the ListView is empty (there are not items in the view).

To make this label visible when the list is empty — set ShowEmptyListMessage property to True. To change the color of label, use EmptyListMessageColor property.

See also
EmptyListMessageColor and ShowEmptyListMessage properties.

10.24.2.4EmptyListMessageColor

Applies to
acListView and acDBListView components.

Declaration
```
property EmptyListMessageColor: TColor; // clGrayText by default
```

Description
The EmptyListMessageColor property specifies the color for the text label in the center of list view, which can be visible if the ListView is empty (there are not items in the view).

To make this label visible when the list is empty — set ShowEmptyListMessage property to True. To change the text for this label, use EmptyListMessage property.

See also

© 1998-2006, UtilMind Solutions®

EmptyListMessage and ShowEmptyListMessage properties.

10.24.2.5HeaderColor

Applies to
acListView and acDBListView components.

Declaration
```
property HeaderColor: TColor;
```

Description
The HeaderColor property can specify custom color for the column headers of the ListView control (when the ViewStyle is vsReport).

When HeaderColor value is clNone, it will draw the header with default color, pre-configured in system.

See also
HeaderFont and HeaderStyle properties.

10.24.2.6HeaderFont

Applies to
acListView and acDBListView components.

Declaration
```
property HeaderFont: TFont;
```

Description
The HeaderFont property specifies the custom font for the column headers of the ListView control (when the ViewStyle is vsReport). Using this property you can change the font name, color and style (make the text bold or italic).

See also
HeaderStyle property.

10.24.2.7HeaderMenu

Applies to
acListView and acDBListView components.

Declaration
```
type
  TacListViewHeaderMenu = class
  published
    property AlignLeftItem: TMenuItem;
    property AlignRightItem: TMenuItem;
    property AlignCenterItem: TMenuItem;
    property AscendingItem: TMenuItem;
    property DescendingItem: TMenuItem;
    property BestFitItem: TMenuItem;
    property PopupMenu: TPopupMenu;
  end;

property HeaderMenu: TacListViewHeaderMenu;
```

Description

© 1998-2006, UtilMind Solutions®

The HeaderMenu structure specifies custom popup menu for the header sections of the acListView control. You can point custom PopupMenu + "Ascending" and "Descending" items in this menu, which will automatically change sorting direction when user selects these items.

Usually you don't need to specify custom menu, because the acListView already has built-in menu with two default items "Sort Ascending" and "Sort Descending". Use the custom popup menu only if you would like to add additional features to the header menu or wish to translate the text for menu items to another language.

If you don't want to show the popup menu for header sections at all — set ShowHeaderMenu property to False.

See also
ShowHeaderMenu and SortDirection properties.

10.24.2.7.1 AlignLeftItem

Applies to
acListView and acDBListView components as subproperty of HeaderMenu structure.

Declaration
```
property AlignLeftItem: TMenuItem;
```

Description
The AlignLeftItem property points to the menu item, in the custom PopupMenu for header sections. When user clicks this menu item, the alignment of column will be automatically changed to taLeftAlignment.

You don't need to handle the OnClick event for this menu item, all operations will be performed automatically.

See also
PopupMenu, AlignRight and AlignCenter properties.

10.24.2.7.2 AlignRightItem

Applies to
acListView and acDBListView components as subproperty of HeaderMenu structure.

Declaration
```
property AlignRightItem: TMenuItem;
```

Description
The AlignRightItem property points to the menu item, in the custom PopupMenu for header sections. When user clicks this menu item, the alignment of column will be automatically changed to taRightAlignment.

You don't need to handle the OnClick event for this menu item, all operations will be performed automatically.

See also
PopupMenu, AlignLeft and AlignCenter properties.

© 1998-2006, UtilMind Solutions®

10.24.2.7.3 AlignCenterItem

Applies to
acListView and acDBListView components as subproperty of HeaderMenu structure.

Declaration
```
property AlignCenterItem: TMenuItem;
```

Description
The AlignCenterItem property points to the menu item, in the custom PopupMenu for header sections. When user clicks this menu item, the alignment of column will be automatically changed to taCenter.

You don't need to handle the OnClick event for this menu item, all operations will be performed automatically.

See also
PopupMenu, AlignLeft and AlignRight properties.

10.24.2.7.4 AscendingItem

Applies to
acListView and acDBListView components as subproperty of HeaderMenu structure.

Declaration
```
property AscendingItem: TMenuItem;
```

Description
The AscendingItem property points to the menu item, in the custom PopupMenu for header sections, which will automatically change the sorting direction to sdAscending.

You don't need to handle the OnClick event for this menu item, all operations will be performed automatically.

See also
PopupMenu, DescendingItem and SortDirection properties.

10.24.2.7.5 DescendingItem

Applies to
acListView and acDBListView components as subproperty of HeaderMenu structure.

Declaration
```
property DescendingItem: TMenuItem;
```

Description
The DescendingItem property points to the menu item, in the custom PopupMenu for header sections, which will automatically change the sorting direction to sdDescending.

You don't need to handle the OnClick event for this menu item, all operations will be performed automatically.

See also
AscendingItem and PopupMenu properties.

© 1998-2006, UtilMind Solutions®

10.24.2.7.6 BestFitItem

Applies to
acListView and acDBListView components as subproperty of HeaderMenu structure.

Declaration
```
property BestFitItem: TMenuItem;
```

Description
The BestFitItem property points to the menu item, in the custom PopupMenu for header sections. When user clicks this menu item, all columns will be automatically adjusted thus all the columns will fit the text of subitems, so all text in list items will became visible.

You don't need to handle the OnClick event for this menu item, all operations will be performed automatically.

See also
PopupMenu property and BestFit method.

10.24.2.7.7 PopupMenu

Applies to
acListView and acDBListView components as subproperty of HeaderMenu structure.

Declaration
```
property PopupMenu: TPopupMenu;
```

Description
The PopupMenu property points to the custom popup menu, which will be used when user clicks the right mouse button. To point the menu items which will sort the columns when user clicks them — use AscendingItem and DescendingItem properties.

By default, the acListView already has the built-in popup menu with two menu items, "Sort Ascending" and "Sort Descending". Use the custom popup menu only if you would like to add additional features to the header menu or wish to translate the text for menu items to another language.

See also
AscendingItem and DescendingItem properties.

10.24.2.8HeaderStyle

Applies to
acListView and acDBListView components.

Declaration
```
type
  THeaderStyle = (hsButtons, hsFlat);

property HeaderStyle: THeaderStyle;
```

Description
The HeaderStyle property specifies the appearance of the header sections above the ListView control. (Header visible only if the ViewStyle is vsReport).

There are the possible values:

© 1998-2006, UtilMind Solutions®

Value	Meaning
hsButtons	The column sections are buttons;
hsFlat	The column sections are flat buttons.

Note

User can not click the header section (change SortDirection) when HeaderStyle = hsFlat.

See also

HeaderColor and HeaderFont properties.

10.24.2.9 ItemPopupMenu

Applies to

acListView and acTreeView components.

Declaration

```
property ItemPopupMenu: TPopupMenu;
```

Description

The ItemPopupMenu allows to specify the popup menu which could be pop up when user right-clicks the tree nodes, so the popup menu for items could be different of that standard menu for entire TreeView control.

10.24.2.10 RegistrySaver

Applies to

acListView, acHostnameResolver and acAppBar components.

Declaration

```
type
  TacListViewRegistrySaver = class
  published
    property Enabled: Boolean;
    property RegKey: String;
    property RegLocation: TacRegLocation;
  end;

property RegistrySaver: TacListViewRegistrySaver;
```

Description

The RegistrySaver structure used to save and restore the sorting rules, ViewStyle and widths of columns on every exit and startup of your program.

Just specify the RegLocation and RegKey where to store these settings, and set Enabled to True. The SortColumn, SortDirection properties and widths of the list columns will be saved on every exit, to be restored on every program restart.

10.24.2.10.1 Enabled

Applies to

acListView component as subproperty of RegistrySaver structure.

Declaration

```
property Enabled: Boolean;
```

© 1998-2006, UtilMind Solutions®

Description
The Enabled property of the RegistrySaver structure controls whether you would like to save the sorting rules (SortColumn and SortDirection properties), ViewStyle, and widths of list columns on every program exit, to restore them on every program restart. Set Enabled to True to automatically save and restore appearance of the acListView component.

See also
RegKey and RegLocation properties.

10.24.2.10.2 RegKey

Applies to
acListView and acAppBar components as subproperty of RegistrySaver structure.

Declaration
```
property RegKey: String;
```

Description
The RegKey property defines the path to the key in the system registry where you would like to store the ListView's settings (sorting rules and widths of columns). All stored settings will be updated every time when user close the form

Default property value is `'\Software\CompanyName\ProgramName\ListView'`. You should point the RegKey to location where stored the configuration settings of your own application. For example: `'\Software\UtilMind Solutions\The Cool Program\Main ListView'`.

See also
RegLocation and Enabled properties.

10.24.2.10.3 RegLocation

Applies to
acListView and acAppBar components as subproperty of RegistrySaver structure.

Declaration
```
type
  TacRegLocation = (rlCurrentUser, rlLocalMachine);

property RegLocation: TacRegLocation;
```

Description
The RegLocation specifies the root key of the hierarchy of subkeys, where you would like to store the ListView's settings (sorting rules and widths of columns). RegLocation property can point to HKEY_CURRENT_USER (rlCurrentUser) or HKEY_LOCAL_MACHINE (rlLocalMachine) root keys.

When RegLocation is `rlCurrentUser`, acListView stores all settings in the registry section for current logged user. When RegLocation is `rlLocalMachine` settings will be saved for ALL users of current machine.

See also
RegKey and Enabled properties.

© 1998-2006, UtilMind Solutions®

ShowEmptyListMessage
10.24.2.11

Applies to
acListView and acDBListView components.

Declaration
```
property ShowEmptyListMessage: Boolean;
```

Description
The ShowEmptyListMessage property specifies whether the control should show the special label (specified in EmptyListMessage property), when the list is empty.

Set the ShowEmptyListMessage property to True if you wish to show the message when the list is empty, or set it to False otherwise (the label will never visible).

To change the color of label, use EmptyListMessageColor property.

See also
EmptyListMessage and EmptyListMessageColor properties.

ShowHeaderMenu
10.24.2.12

Applies to
acListView and acDBListView components.

Declaration
```
property ShowHeaderMenu: Boolean;
```

Description
The ShowHeaderMenu property specifies whether you would like to use the popup menu for the header sections.

See also
HeaderMenu structure.

ShowScrollTips
10.24.2.13

Applies to
acListView and acDBListView components.

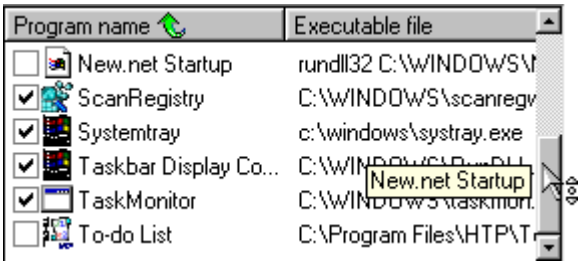
Declaration
```
property ShowScrollTips: Boolean;
```

Description
The ShowScrollTips property controls whether you would like to show the so-called "thumb-tracking" tips. When ShowScrollTips = True and user drags the thumb on the vertical scroll bar, the acListView will display the hint with caption of top visible list item of current sorted column.

Remark
Scrollbar tips appears only when view style is report (ViewStyle = vsReport).

Screenshot

© 1998-2006, UtilMind Solutions®



ShowSortMark
10.24.2.14

Applies to
acListView and acDBListView components.

Declaration
```
property ShowSortMark: Boolean;
```

Description
The ShowSortMark property specifies whether you would like to show the arrow-style sort mark (or) on the header sections, which indicates the sorting direction on the column. Set ShowSortMark to False if you don't want to show these marks.

The sort mark can be "raised" or "sunken". To change the style of sort mark use SortMarkKind property.

Note
The sort mark will be invisible anyway, if SortType property is stNone.

See also
SortMarkKind, SortColumn and SortDirection properties.

ShowToolTips
10.24.2.15

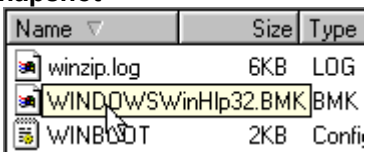
Applies to
acListView and acDBListView components.

Declaration
```
property ShowToolTips: Boolean;
```

Description
The ShowToolTips property controls whether you would like to show tooltips (hints) when mouse pointer moves over partially hidden list item.

To specify behavior of the tooltips — use ToopTipOptions property.

Snapshot



See also
ToolTipOptions property.

© 1998-2006, UtilMind Solutions®

SortColumn
10.24.2.16

Applies to
acListView and acDBListView components.

Declaration
```
property SortColumn: Word;
```

Description
The SortColumn property determines and specifies the column number which should be sorted. When SortColumn = 0, first column will be sorted, SortColumn = 1 will sort second column and so forth..

To specify the sorting direction — use SortDirection property.

See also
SortDirection and ShowSortMark properties.

SortDirection
10.24.2.17

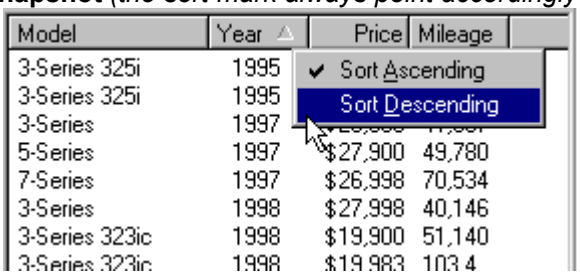
Applies to
acListView and acDBListView components.

Declaration
```
type
  TacListViewSortDirection = (sdAscending, sdDescending);

property SortDirection: TacListViewSortDirection;
```

Description
The SortDirection property controls the sorting direction for currently sorted column. The direction can be ascending (sdAscending) or descending (sdDescending).

User can change the sorting direction at run-time, clicking the column header or selecting "Sort Ascending" or "Sort Descending" menu items in the header popup menu.

Snapshot *(the sort mark always point accordingly to current sorting direction)*



The arrow-style sort mark can be invisible if the list is unsorted (SortType = stNone).

See also
SortColumn, ShowSortMark and HeaderMenu properties.

© 1998-2006, UtilMind Solutions®

SortMarkAlign
10.24.2.18

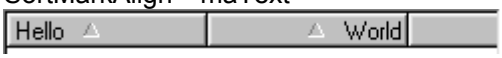
Applies to
acListView and acDBListView components.

Declaration
```
type
  TacListViewSortMarkAlign = (maText, maEdge);

property SortMarkAlign: TacListViewSortMarkAlign;
```

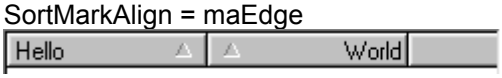
Description
The SortMarkAlign property controls placement of the arrow-style sort mark, which indicates current sorted column and sorting direction.

The sort mark can be displayed near the text (maText) or near the edge (maEdge) of header section.

Snapshots
SortMarkAlign = maText



SortMarkAlign = maEdge



Notes
If you don't want to show the sort mark at all — set ShowSortMark property to False.
The arrow-style sort mark can be invisible if the list is unsorted (SortType = stNone).

See also
SortMarkKind, ShowSortMark, SortColumn and SortDirection properties.

SortMarkImageAsc
10.24.2.19

Applies to
acListView and acDBListView components.

Declaration
```
property SortMarkImageAsc: TPicture;
```

Description
The SortMarkImageAsc property can be used to specify custom picture for sort mark, for column with ascending sorting.

Snapshot

Notes
If you don't want to show the sort mark at all — set ShowSortMark property to False.
The arrow-style sort mark can be invisible if the list is unsorted (SortType = stNone).

© 1998-2006, UtilMind Solutions®

See also
SortMarkImageDesc property.

SortMarkImageDesc
10.24.2.20

Applies to
acListView and acDBListView components.

Declaration
```
property SortMarkImageDesc: TPicture;
```

Description
The SortMarkImageDesc property can be used to specify custom picture for sort mark, for column with descending sorting.

Snapshot

Notes
If you don't want to show the sort mark at all — set ShowSortMark property to False.
The arrow-style sort mark can be invisible if the list is unsorted (SortType = stNone).

See also
SortMarkImageAsc property.

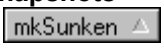
SortMarkKind
10.24.2.21

Applies to
acListView and acDBListView components.

Declaration
```
type
  TacListViewSortMarkKind = (mkSunken, mkRaised);

property SortMarkKind: TacListViewSortMarkKind;
```

Description
The SortMarkKind property controls the appearance of the arrow-style sort mark, which indicates current sorted column and sorting direction.

The sort mark can be "sunken" (mkSunken, by default) or "raised" (mkRaised).

Snapshots


or

Notes
If you don't want to show the sort mark — set ShowSortMark property to False.
The arrow-style sort mark can be invisible if the list is unsorted (SortType = stNone).

See also
SortMarkAlign, ShowSortMark, SortColumn and SortDirection properties.

© 1998-2006, UtilMind Solutions®

ToolTipOptions
10.24.2.22

Applies to
acListView and acDBListView components.

Declaration
```
type
  TacListViewToolTipOption = (ttoInstantShow, ttoLongStay, ttoBelowItem,
ttoOffset);
  TacListViewToolTipOptions = set of TacListViewToolTipOption;

property ToolTipOptions: TacListViewToolTipOptions;
```

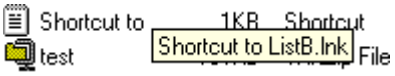
Description
The ToolTipOptions property controls behavior of the tooltips (hints) for partially hidden list items.

There are possible values:

Value	Meaning
ttoInstantShow	show the tooltips instantly when the mouse overs the list item (without a second delay);
ttoLongStay	tooltips does not disappears until user move mouse pointer away from the list item, when ttoLongStay is True. Usually tooltips stays visible only 5 seconds;
ttoBelowItem	the tooltip will be displayed below item (will not cover it by itself);
ttoOffset	offsets the tooltip to display it to the right and below the item.

Snapshots
ttoBelowItem = True

ttoOffset = True



See also
ShowToolTips property.

10.24.3 Methods

10.24.3.1BestFit

Applies to
acListView and acDBListView components.

Declaration
```
procedure BestFit(FitHeaderText: Boolean = False);
```

Description
The BestFit method adjusts the column widths (of report-style list-view) thus the text of all subitems will fit to the widths of columns (maximum length of list items OR maximum length of text on the header).

The FitHeaderText parameter controls whether the columns will be adjusted by length of list items or by length of header items. To make it fit the text of header — set FitHeaderText parameter to True

© 1998-2006, UtilMind Solutions®

(call this method as `BestFit(True)`).

The equivalent of this method exists in every list-view. User can adjust the column widths pressing [Ctrl] + [Plus] keys.

See also
BestFitItem property of HeaderMenu structure.

10.24.3.2Clear

Applies to
acListView and acDBListView components.

Declaration
```
procedure Clear;
```

Description
The Clear method deletes all the listview control. Call Clear to empty the list control so it contains no items.

See also
DeleteSelected method.

10.24.3.3CopySelectionToClipboard

Applies to
acListView and acDBListView components.

Declaration
```
procedure CopySelectionToClipboard;
```

Description
The CopySelectionToClipboard method copies all text from selected rows/columns to the clipboard.

All copied columns separated by Tab character (#9), all rows by CRLF characters (#13#10).

The acListView automatically calls this method to copy selection to the clipboard when user press Ctrl+C or Ctrl+Ins keys.

See also
DeleteSelected method.

10.24.3.4CountPerPage

Applies to
acListView and acDBListView components.

Declaration
```
function CountPerPage: Integer;
```

Description
The CountPerPage function returns the number of items that can fit vertically in the visible area of a list-view control when in list or report view.

Notes
1. Only fully visible items are counted;

© 1998-2006, UtilMind Solutions®

2. If the current view is icon or small icon view, the return value is the total number of items in the list-view control.

10.24.3.5DeleteSelected

Applies to
acListView and acDBListView components.

Declaration
```
procedure DeleteSelected;
```

Description
The DeleteSelected method deletes all the selected items. Call DeleteSelected to remove all selected items from the list control. DeleteSelected deletes only the item or items that are selected.

See also
Clear and SelectAll methods.

10.24.3.6ExportToHTML

Applies to
acListView and acDBListView components.

Declaration
```
procedure ExportToHTML(const FileName: String; ExportColumnCaptions:
Boolean = True; const DocumentName: String = '');
```

Description
The ExportToHTML procedure can export all items and columns (if *ExportColumCaption* parameter is True) to the HTML file specified in *FileName* parameter.

It will create simple HTML file and put all items and columns to the table. The background color of cells with column names will be "silver".

Example
```
procedure TMainForm.AsTXT1Click(Sender: TObject);
const
  EXT_TYPE: Array[0..1] of String = ('TXT', 'HTML');
  ExportConnectionsStr = 'Export Connections to %s file';
begin
  with SaveDialog1, (Sender as TComponent) do
   begin
    DefaultExt := EXT_TYPE[Tag];
    Title := Format(ExportConnectionsStr, [UpperCase(DefaultExt)]);
    FilterIndex := Tag + 1;
    if Execute then
     if Tag = 0 then
       acListView1.ExportToTXT(FileName)
     else
       acListView1.ExportToHTML(FileName);
   end;
end;
```

See also
ExportToHTML method.

© 1998-2006, UtilMind Solutions®

10.24.3.7ExportToTXT

Applies to
acListView and acDBListView components.

Declaration
```
procedure ExportToTXT(const FileName: String; ExportColumnCaptions:
Boolean = True);
```

Description
The ExportToTXT procedure can export all items and columns (if *ExportColumCaption* parameter is True) to the text file specified in *FileName* parameter.

Each row separated by CRLF (#13#10) characters, every column separated by Tab (#9) character.

Example
```
procedure TMainForm.AsTXT1Click(Sender: TObject);
const
  EXT_TYPE: Array[0..1] of String = ('TXT', 'HTML');
  ExportConnectionsStr = 'Export Connections to %s file';
begin
  with SaveDialog1, (Sender as TComponent) do
   begin
    DefaultExt := EXT_TYPE[Tag];
    Title := Format(ExportConnectionsStr, [UpperCase(DefaultExt)]);
    FilterIndex := Tag + 1;
    if Execute then
     if Tag = 0 then
       acListView1.ExportToTXT(FileName)
     else
       acListView1.ExportToHTML(FileName);
   end;
end;
```

See also
ImportFromTXT and ExportToHTML methods.

10.24.3.8FindCaption

Applies to
acListView and acDBListView components.

Declaration
```
function FindCaption(const Caption: String): TListItem;
```

Description
The FindCaption function finds the list item, if any, with specified caption. This is another, overloaded variation of standard FindCaption method, but much simplified.

Notes
The Caption parameter can be case insensitive, so, for example, if you search for "tEsT", the function can return the list item where caption is "Test".
This type of FindCaption method exists only in Delphi 4/C++ Builder 4 and higher.

Original code
```
{$IFDEF D4}
```

© 1998-2006, UtilMind Solutions®

```
function TacCustomListView.FindCaption(const Caption: String):
TListItem;
begin
  Result := inherited FindCaption(0, Caption, False, True, False);
end;
{$ENDIF}
```

See also
GetColumnAt method.

10.24.3.9 GetColumnAt

Applies to
acListView and acDBListView components.

Declaration
```
function GetColumnAt(X: Integer): TListColumn;
```

Description
The GetColumnAt method returns the list column at the specified X-coordinate.

Call GetColumnAt to obtain the TListColumn object for the list item found at the position specified by the X parameter. X specify the coordinate of the position, in pixels, relative to the top left corner of the list view. If there is no item at the location, GetColumnAt returns nil.

Example
```
procedure TForm1.acListView1MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
  Column: TListColumn;
begin
  Column := acListView1.GetColumnAt(X);
  if Column <> nil then
    Caption := Column.Caption;
end;
```

See also
GetItemAt method.

10.24.3.10 GetItemAt

Applies to
acListView and acDBListView components.

Declaration
```
function GetItemAt(X, Y: Integer): TListItem;
```

Description
The GetItemAt method returns the list item at the specified position.

Call GetItemAt to obtain the TListItem object for the list item found at the position specified by the X and Y parameters. X and Y specify the coordinates of the position, in pixels, relative to the top left corner of the list view. If there is no item at the location, GetItemAt returns nil.

Example
```
procedure TForm1.acListView1MouseDown(Sender: TObject;
```

© 1998-2006, UtilMind Solutions®

```
    Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
  var
    Item: TListItem;
  begin
    Item := acListView1.GetItemAt(X, Y);
    if Item <> nil then
      Caption := Item.Caption;
  end;
```

See also
GetColumnAt method.

ImportFromTXT
10.24.3.11

Applies to
acListView and acDBListView components.

Declaration
```
function ImportFromTXT(const FileName: String; SeparatorChar: Char =
#9): Boolean;
```

Description
The ImportFromTXT method imports the list items from the plain text file to the ListView.

When it imports items to ListView, it considers each new string as separate row, and the columns should be separated with character specified in *SeparatorChar* parameter (Tab by default).

Example
```
MyListViewView.ImportFromTXT('MyTemplate.txt');
```

See also
ExportToTXT and ExportToHTML methods.

IsScrollBarsVisible
10.24.3.12

Applies to
acListView and acDBListView components.

Declaration
```
function IsScrollBarsVisible: Boolean;
```

Description
The IsScrollBarsVisible method determines whether the content of the ListView does not fits to the control's bounds and it shows the scrollbars.

IsScrollBarsVisible returns True if horizontal or vertical scrollbar actually are visible, or False if ListView doesn't need the scrollbars.

Example
```
while ListView1.IsScrollBarsVisible do
  ListView1.Height := ListView1.Height + 24;
```

© 1998-2006, UtilMind Solutions®

SelectAll
10.24.3.13

Applies to
acListView and acDBListView components.

Declaration
```
procedure SelectAll;
```

Description
The SelectAll method selects all items or all text in the selected item.

See also
DeleteSelected method.

SetColumnOrders
10.24.3.14

Applies to
acListView and acDBListView components.

Declaration
```
procedure SetColumnOrders(ColumnOrders: Array of Integer);
```

Description
The SetColumnOrders method sets the orders of header items of the listview, or copies the orders from another listview.

Example
```
// demonstrates how to copy the orders from ListView1 to ListView2 when
the order changes
procedure TMainForm.ListView1ColumnDragged(Sender: TObject;
  ColumnOrders: array of Integer);
begin
  ListView2.SetColumnOrders(ColumnOrders);
end;
```

10.24.4 Events

10.24.4.1OnAlignmentChanged

Applies to
acListView and acDBListView components.

Declaration
```
type
  TLVColumnClickEvent = procedure(Sender: TObject; Column: TListColumn)
of object;

property OnAlignmentChanged: TLVColumnClickEvent;
```

Description
The OnAlignmentChanged event occurs when user changes the text alignment for some certain column, clicking the menu item specified in AlignLeftItemClick, AlignRightItemClick or AlignCenterItemClick properties of HeaderMenu structure.

The *Column* parameter determines the column, which alignment has been changed.

© 1998-2006, UtilMind Solutions®

Note

This event works only when user changes the column's alignment using menu items specified in HeaderMenu. It does not works when you programmatically change the alignment using `Columns[ColumnIndex].Alignment` property.

See also

HeaderMenu property.

10.24.4.2OnCheckBoxClick

Applies to

acListView and acDBListView components.

Declaration

```
type
  TacListViewItemEvent = procedure(Sender: TObject; Item: TListItem) of
object;

property OnCheckBoxClick: TacListViewItemEvent;
```

Description

The OnCheckBoxClick event occurs when user clicks the checkbox at the left side of list item (when ViewStyle = vsReport).

See also

CheckOnClick property.

10.24.4.3OnColumnDragged

Applies to

acListView and acDBListView components.

Declaration

```
type
  TacListViewColumnDraggedEvent = procedure(Sender: TObject;
ColumnOrders: array of Integer) of object;

property OnColumnDragged: TacListViewColumnDraggedEvent;
```

Description

The OnColumnDragged event occurs when user have moved the column using a mouse. The *ColumnOrders* parameter determines new orders of the columns.

To make columns of the list-view "draggable" — set FullDrag property to True.

Example

```
procedure TForm1.SearchResultsViewColumnDragged(Sender: TObject;
  ColumnOrders: array of Integer);
var
  I: Integer;
begin
  for I := 0 to ListView1.Columns.Count - 1 do
    Memo1.Lines.Add(IntToStr(ColumnOrders[I]));
end;
```

Note

© 1998-2006, UtilMind Solutions®

This event supported since Delphi 4 and later or C++ Builder 4 and later

See also
OnColumnResize event.

10.24.4.4OnColumnResizeBegin

Applies to
acListView and acDBListView components.

Declaration
```
type
  TacListViewColumnResizeBeginEvent = procedure(Sender: TObject; Column:
TListColumn; var AllowResize: Boolean) of object;

property OnColumnResizeBegin: TacListViewColumnResizeBeginEvent;
```

Description
The OnColumnResizeBegin event occurs when user started to resize the list column, by dragging the divider on its header (when ViewStyle is vsReport).

The *Column* parameter determines the column are about to be resized.
The *AllowResize* parameter can be used to disallow resizing, by specifying False into it in the event handler.

Example
```
procedure TForm1.acListView1ColumnResizeBegin(Sender: TObject;
  Column: TListColumn; var AllowResize: Boolean);
begin
  AllowResize := (Column.Index <> 2) and (Column.Index <> 3);
end;
```

See also
OnColumnResizeEnd and OnColumnDragged events.

10.24.4.5OnColumnResizeEnd

Applies to
acListView and acDBListView components.

Declaration
```
type
  TLVColumnClickEvent = procedure(Sender: TObject; Column: TListColumn)
of object;

property OnColumnResizeEnd: TLVColumnClickEvent;
```

Description
The OnColumnResizeEnd event occurs when user has finished resize the list column, by dragging the divider on its header. The *Column* parameter determines which column of the list view has been resized.

See also
OnColumnResizeBegin and OnColumnDragged events.

© 1998-2006, UtilMind Solutions®

10.24.4.6OnCustomDrawSubItemEx

Applies to
acListView and acDBListView components.

Declaration

```
type
  TacListViewCustomDrawSubItemExEvent = procedure(Sender: TObject;
Canvas: TCanvas; Rect: TRect; Item: TListItem; SubItem: Integer; State:
TCustomDrawState; var DefaultDraw: Boolean) of object;

property OnCustomDrawSubItemEx: TacListViewCustomDrawSubItemExEvent;
```

Description
The OnCustomDrawSubItemEx event occurs when a subitem must be rendered in the list-view. This is a bug-free replacement of standard OnCustomDrawSubItem event existing in usual list-view controls. Additionally it passes to the event handler the *Canvas* and drawing rectangle (*Rect*).

Write code in an OnCustomDrawSubItemEx handler to draw to the subitems that appear in additional columns to the right of each item when ViewStyle is vsReport. Use the Canvas property passed to the handler as a drawing surface. *(Do NOT use the list-view's Canvas!! This is major bug of standard OnCustomDrawSubItem event. The owner's font appears broken on Win95/98/ME after first custom drawing.)*

The *Sender* Parameter specifies the list view that owns the subitems. The *Item* parameter is the current Item being drawn. The *SubItem* parameter is the index of the subitem of that list item in its SubItems property. The *State* indicates various attributes that can affect the way the subitem is drawn. Set *DefaultDraw* to False to prevent the list view from adding the subitem's text after the event handler exits.

Example

```
procedure TForm1.ListView1CustomDrawSubItemEx(Sender: TObject;
  Canvas: TCanvas; Rect: TRect; Item: TListItem; SubItem: Integer;
  State: TCustomDrawState; var DefaultDraw: Boolean);
begin
  if Columns[SubItem].ImageIndex <> -1 then
   begin
    ImageList1.Draw(Canvas, Rect.Left + 5, Rect.Top + 1, 0);
    DefaultDraw := False;
   end;
end;
```

Note
Custom drawing of sub-items of the list-view controls supported in Delphi 4/C++ Builder 4 and higher. This event does not exists in Delphi 2/3 or C++ Builder 3.

10.24.4.7OnDrawHeaderSection

Applies to
acListView and acDBListView components.

Declaration

```
type
  TacListViewDrawHeaderSection = procedure(Sender: TObject; Canvas:
TCanvas; Rect: TRect; ColumnIndex: Integer; Pressed: Boolean) of object;

property OnDrawHeaderSection: TacListViewDrawHeaderSection;
```

© 1998-2006, UtilMind Solutions®

Description
The OnDrawHeaderSection event occurs when the header section needs to be drawn.

Write an OnDrawHeaderSection event handler to draw the contents for all the sections in the ListView. Draw on the header using the methods of the *Canvas* property. The *Rect* parameter indicates the location of the cell on the canvas. The *ColumnIndex* parameter indicate the column of the header that should be drawn. The *Pressed* parameter indicates whether the coulum is selected by mouse (pressed).

If the OnDrawHeaderSection handler is not assigned, the component will paint default content.

10.24.4.8 OnMouseEnter

Applies to
acLabel, acURLLabel, acRichLabel, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox, acTreeComboBox, acHeader, acHeaderPanel, acTreeView and acListView components.

Declaration
```
property OnMouseEnter: TNotifyEvent;
```

Description
The OnMouseEnter occurs when mouse pointer hovers the control.

See also
OnMouseLeave event.

10.24.4.9 OnMouseLeave

Applies to
acLabel, acURLLabel, acRichLabel, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox, acTreeComboBox, acHeader, acHeaderPanel, acTreeView and acListView components.

Declaration
```
property OnMouseLeave: TNotifyEvent;
```

Description
The OnMouseLeave occurs when mouse pointer leaves the control.

See also
OnMouseEnter event.

10.24.4.10 OnSettingsChanged

Applies to
acListView and acDBListView components.

Declaration
```
property OnSettingsChanged: TNotifyEvent;
```

Description
The OnSettingsChanged event occurs when any system settings (system metrics or regional settings) is changed.

Write OnSettingsChanged event handler to refresh the list items when any system settings is

© 1998-2006, UtilMind Solutions®

changed. For example, to refresh items which contains date/time values when the date/time format is changed. (The format settings in your application will be already updated automatically.)

OnToolTip
10.24.4.11

Applies to
acListView and acDBListView components.

Declaration
```
type
  TacListViewToolTipEvent = procedure(Sender: TObject; var ToolTipText:
String; Item: TListItem; LogicalCol: Integer) of object;

property OnToolTip: TacListViewToolTipEvent;
```

Description
The OnToolTip event occurs when the acListView is about to show the tooltip (hint) for some partially hidden list item.

Write OnToolTip event handler to specify custom text for the tooltips. Use *ToolTipText* parameter to get or set the tooltip text for specific list item. To determinate the list item under mouse pointer — use *Item* parameter. To determinate whether this item was a subitem — read the column number from the *LogicalCol* parameter.

Screenshot *(custom tooltips shows ID3 tags of MP3 files)*

Example
```
procedure TformSearch.SearchResultsListViewToolTip(Sender: TObject;
  var ToolTipText: String; Item: TListItem; LogicalCol: Integer);
begin
  with SearchResultsListView do
   if ToolTipText = '' then
    begin
     // Dynamically change the tooltip style
     ToolTipOptions := [ttoLongStay, ttoOffset];
     with Item do
      ToolTipText := 'Name: ' + Item.Caption + #13#10 +
                     'Sex: ' + SubItems[0] + #13#10 +
                     'Age: ' + SubItems[1] + #13#10 +
                     'Location: ' + SubItems[2];
    end
   else
    // Dynamically change the tooltip style
    ToolTipOptions := [];
end;
```

See also

© 1998-2006, UtilMind Solutions®

ShowToolTip and ToolTipOptions properties.

10.25 acMRUComboBox

10.25.1 TacMRUComboBox

Overview

The acMRUComboBox is the edit control with the drop down list which can save and restore the MRU history list (list of most recently used lines) from the registry. When user type the text, the combobox can automatically complete the input with the text taken from the history list. Also it contains the AttachedLabel structure which lets to operate with the label associated with the edit control and specify its behavior and position near the control.

Snapshot

Additional and improved properties

Align	controls placement of the combobox within its container;
AttachedLabel	controls the behavior of the label attached to the control and its position;
AutoFill	whether you would like to auto-complete the input with the string taken from the history list;
AutoFillCaseSensitive	whether the characters case doesn't matter and whether you would like to auto-complete the input disregarding the case of characters;
AutoFillDelay	speicifes the delay, in milliseconds, which should pass between keypress and attempt to auto-complete the input;
MRUEnabled	controls whether the MRU history list should be automatically restored from the registry after creation of the control (on program startup);
MRUMax	specifies the maximum number of items which can contains the MRU history list;
MRURegKey	defines the path to the key in the system registry where you would like to store the MRU history list;
MRURegLocation	specifies the root key of the hierarchy of subkeys, where you would like to store the MRU history list;
MRUSaveOnExit	controls whether the combobox should automatically add the current text from the edit box to the MRU history list, and save the MRU list to the registry, when user moves the focus to another control.

New methods

ClearMRU	clears all items from the history (from drop-down list), and removes them from registry.
Save	moves the current text from the edit box to the MRU history list and saves the MRU list to the system registry.

New events

OnMouseEnter	occurs when mouse pointer passes over the combobox;

© 1998-2006, UtilMind Solutions®

OnMouseLeave occurs when mouse pointer leaves the combobox;

See also
acImagesComboBox component.

10.25.2 Properties

10.25.2.1Align

Applies to
acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acIPEdit and acMRUComboBox components *(in generally to ALL successors of TControl)*.

Declaration
```
type
  TAlign = (alNone, alTop, alBottom, alLeft, alRight, alClient);

property Align: TAlign;
```

Declaration
The Align property determines how the control aligns within its container (or parent control).

Use Align to align a control to the top, bottom, left, or right of a form or panel and have it remain there even if the size of the form, panel, or component that contains the control changes.

For example, to use a panel component with various controls on it as a tool palette, change the panel's Align value to alLeft. The value of alLeft for the Align property of the panel guarantees that the tool palette remains on the left side of the form and always equals the client height of the form.

The default value of Align is alNone, which means a control remains where it is positioned on a form or panel. Specify another value to align the control. Use one of these possible values:

Value	Meaning
alNone	The component remains where it was placed on the form or panel. This is the default value.
alTop	The component moves to the top of the form and resizes to fill the width of the form. The height of the component is not affected.
alBottom	The component moves to the bottom of the form and resizes to fill the width of the form. The height of the component is not affected.
alLeft	The component moves to the left side of the form and resizes to fill the height of the form. The width of the component is not affected.
alRight	The component moves to the right side of the form and resizes to fill the height of the form. The width of the component is not affected.
alClient	The component resizes to fill the client area of a form. If a component already occupies part of the client area, the component resizes to fit within the remaining client area.

Note
If Align is set to alClient, the control fills the entire client area so that it is impossible to select the parent form by clicking on it. In this case, select the parent by selecting the control on the form and pressing Esc, or by using the Object Inspector.

Any number of child components within a single parent can have the same Align value, in which case they stack up along the edge of the parent form or panel. The child controls stack up in z-

© 1998-2006, UtilMind Solutions®

order. To adjust the order in which the controls stack up, drag the controls into their desired positions.

10.25.2.2AttachedLabel

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components.

Unit

acAttachedLabel

Declaration

```
type
  TacAttachedLabelPosition = (lpAbove, lpBelow, lpLeft, lpRight);
  TacAttachedLabelSpacingKind = (skNearPoint, skFarPoint);
  TacAttachedLabel = class
    property Caption: TCaption;
    property Color: TColor default clBtnFace;
    property Cursor: TCursor default crDefault;
    property Enabled: Boolean;
    property FocusOnClick: Boolean default True;
    property Font: TFont;
    property Hint: String;
    property ParentColor: Boolean default True;
    property ParentFont: Boolean default True;
    property ParentShowHint: Boolean default True;
    property PopupMenu: TPopupMenu;
    property Position: TacAttachedLabelPosition default lpLeft;
    property ShowAccelChar: Boolean default True;
    property ShowHint: Boolean default False;
    property Spacing: Integer default 4;
    property SpacingKind: TacAttachedLabelSpacingKind default
skNearPoint;
    property Transparent: Boolean default False;
    property Visible: Boolean;
    property WordWrap: Boolean default False;
  end;
```

Description

The AttachedLabel structure lets you to operate with the label attached to the some side of control (at the left or right side, above or below). You can specify the behaviour of label, its position near the control, and accelerator key for the control (character in caption, after ampersand (&)).

The label always associated with the control. When user press the accelerator key (Alt + Character, specified after ampersand (&)), the input focus will be imediately switched to the labeled control. The input focus also will be moved to the control if user clicks the label with mouse.

Screenshot (the labels at the left side is attached to the edit control):

See also

OnLabelClick and OnLabelDblClick events.

© 1998-2006, UtilMind Solutions®

10.25.2.2.1 Caption

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property Caption: TCaption;
```

Description
The Caption property specified a text string which appers on label attached to control (attached label).

To underline a character in a Caption that labels a component, include an ampersand (&) before the character. This type of character is called an accelerator character. The user can then select the component by pressing Alt while typing the underlined character. To display an ampersand character in the caption, use two ampersands (&&).

The label always associated with the control. When user press the accelerator key (Alt + Character, specified after ampersand (&)), the input focus will be imediately switched to the labeled control.

Screenshot (the labels at the left side is attached to the edit control):

See also
ShowAccelChar, Color, Font, Transparent, Visible and WordWrap properties.

10.25.2.2.2 Color

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property Color: TColor;
```

Description
The Color property specifies the background color for label attached to control. Use Color to read or change the background color of the control.

If a control's ParentColor property is true, then changing the Color property of the control's parent automatically changes the Color property of the control. When the value of the Color property is changed, the control's ParentColor property is automatically set to False.

If you don't want to show background — set Transparent property to True.

See also
ParentColor and Transparent properties.

© 1998-2006, UtilMind Solutions®

10.25.2.2.3 Cursor

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property Cursor: TCursor;
```

Description
The Cursor property specifies the image used to represent the mouse pointer when it passes into the region covered by this label attached to control.

Change the value of Cursor to provide feedback to the user when the mouse pointer enters the control. The value of Cursor is the index of the cursor in the list of cursors maintained by the global variable, Screen. In addition to the built-in cursors provided by TScreen, applications can add custom cursors to the list.

See also
Additional Cursors

10.25.2.2.4 Enabled

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property Enabled: Boolean;
```

Description
The Enabled property controls whether the control responds to mouse, keyboard, and timer events.

Use Enabled to change the availability of the control to the user. To disable a control, set Enabled to false. Disabled controls appear dimmed. If Enabled is false, the control ignores mouse, keyboard, and timer events.

To re-enable a control, set Enabled to true. The control is no longer dimmed, and the user can use the control.

See also
Caption and Color properties.

10.25.2.2.5 FocusOnClick

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property FocusOnClick: Boolean; // True by default
```

Description
The FocusOnClick property controls whether the control, associated with the attached label, should accept focus when user clicks the label with mouse.

© 1998-2006, UtilMind Solutions®

Set FocusOnClick to True if you want to switch focus to the control when user clicks the label, or set it to False otherwise.

Note
Additionally you can use OnLabelClick and OnLabelDblClick events of the control.

10.25.2.2.6 Font

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property Font: TFont;
```

Description
The Font property controls the attributes of text written on the attached label of control.

To change to a new font, specify a new TFont object. To modify a font, change the value of the Charset, Color, Height, Name, Pitch, Size, or Style of the TFont object.

See also
ParentFont property.

10.25.2.2.7 Hint

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property Hint: String;
```

Description
The Hint property contains the text string that can appear when the user moves the mouse over the label attached control.

If you are using the FormHelp component to show the context-sensitive help taken from secondary part of hint AND the Hint for attached label is not specified, the FormHelp will use the hint of main control.

See also
ShowHint and ParentShowHint properties;
acFormHelp and acFormHints components.

10.25.2.2.8 ParentColor

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property ParentColor: Boolean;
```

Description
The ParentColor determines where an attached label looks for its color information.

© 1998-2006, UtilMind Solutions®

To have a control use the same color as its parent control, set ParentColor to true If ParentColor is false the control uses its own Color property.

Set ParentColor to true for all controls in order to ensure that all the controls on a form have a uniform appearance. For example, if ParentColor is true for all controls in a form, changing the background color of the form to gray causes all the controls on the form to also have a gray background.

When the value of a control's Color property changes, ParentColor becomes false automatically.

See also
Color and Transparent properties.

10.25.2.2.9 ParentFont

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property ParentFont: Boolean;
```

Description
The ParentFont property determines where a control looks for its font information.

To have a control use the same font as its parent control, set ParentFont to true If ParentFont is false the control uses its own Font property.

Set ParentFont to true for all controls in order to ensure that all the controls on a form have a uniform appearance. For example, if ParentFont is true for all controls in a form, changing the form's Font property to 12-point Courier causes all controls on the form to use that font.

When the value of a control's Font property changes, ParentFont becomes false automatically.

When ParentFont is true for a form, the form uses the default font.

See also
Font property.

10.25.2.2.10 ParentShowHint

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property ParentShowHint: Boolean;
```

Description
The ParentShowHint property determines where an attached label looks to find out if its Help Hint should be shown.

Use ParentShowHint to ensure that all the controls on a form either uniformly show their Help Hints or uniformly do not show them.

If ParentShowHint is true, the control uses the ShowHint property value of its parent. If ParentShowHint is false, the control uses the value of its own ShowHint property.

© 1998-2006, UtilMind Solutions®

To provide Help Hints for only selected controls on a form, set the ShowHint property for those controls that should have Help Hints to true, and ParentShowHint becomes false automatically.

Note
Enable or disable all Help Hints for the entire application using the ShowHint property of the application object.

See also
ShowHint and Hint properties;
acFormHelp and acFormHints components.

10.25.2.2.11 PopupMenu

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property PopupMenu: TPopupMenu;
```

Description
The PopupMenu property Identifies the pop-up menu associated with the label attached to control.

Assign a value to PopupMenu to make a pop-up menu appear when the user selects the label and clicks the right mouse button. If the TPopupMenu's AutoPopup property is True, the pop-up menu appears automatically. If the menu's AutoPopup property is False, display the menu with a call to its Popup method from the control's OnContextPopup event handler.

10.25.2.2.12 Position

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
type
  TacAttachedLabelPosition = (lpAbove, lpBelow, lpLeft, lpRight);

property Position: TacAttachedLabelPosition;
```

Description
The Position property of the attached label specifies the layout of the label near the control.

You can make the label visible above, below, at the left or the right side of control.

See also
Spacing and SpacingKind properties.

10.25.2.2.13 ShowAccelChar

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property ShowAccelChar: Boolean;
```

© 1998-2006, UtilMind Solutions®

Description

The ShowAccelChar property determines how an ampersand in the label text is displayed.

Set ShowAccelChar to True to allow the label to display an underlined accelerator key value. When ShowAccelChar is True, any character preceded by an ampersand (&) appears underlined. If the FocusControl property is set, the windowed control specified by the FocusControl property receives input focus when the user types that underlined character. To display an ampersand when ShowAccelChar is True, use two ampersands (&&) to stand for the single ampersand that is displayed.

Set ShowAccelChar to False to display the label text with all ampersands appearing as ampersands.

Example

This example uses two labels on a form. The first label has a caption with an accelerator character in it. The second label also includes an ampersand, but it does not appear as an accelerator character.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Label1.ShowAccelChar := True;
  Label1.Caption := 'An &Underlined character appears here';
  Label2.ShowAccelChar := False;
  Label2.Caption := 'An ampersand (&) appears here';
end;
```

See also

Caption property.

10.25.2.2.14 ShowHint

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property ShowHint: Boolean;
```

Description

The ShowHint property determines whether the label attached to control displays a Help Hint when the mouse pointer rests momentarily on the label.

The Help Hint is the value of the Hint property, which is displayed in a box just beneath the control. Use ShowHint to determine whether a Help Hint appears for the control.

To enable Help Hint for a particular control, the application ShowHint property must be true and either the control's own ShowHint property must be true, or the control's ParentShowHint property must be true and its parent's ShowHint property must be true.

For example, imagine a check box within a group box. If the ShowHint property of the group box is true and the ParentShowHint property of the check box is true, but the ShowHint property of the check box is false, the check box still displays its Help Hint.

Changing the ShowHint value automatically sets the ParentShowHint property to false.

See also

© 1998-2006, UtilMind Solutions®

Hint and ParentShowHint properties.

10.25.2.2.15 Spacing

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property Spacing: Integer;
```

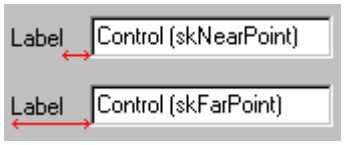
Description
The Spacing property determines the space (in pixels) between the control and attached label.

There is two spacing kinds which have an influence on distance between the control and label (see SpacingKind property).

When SpacingKind is `skNearPoint`, the Spacing property signifies the distance between control and nearest point of attached label.

When SpacingKind is `skFarPoint`, the Spacing property signigies the distance between control and far point of label.

Screenshot



See also
SpacingKind and Position properties.

10.25.2.2.16 SpacingKind

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
type
  TacAttachedLabelSpacingKind = (skNearPoint, skFarPoint);

property SpacingKind: TacAttachedLabelSpacingKind;
```

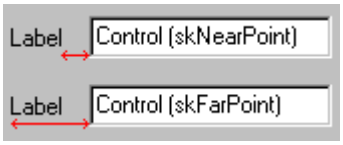
Description
The SpacingKind property specifies how to measure the spacing between the control and attached label.

There is two possible values for SpacingKind property:

Value	Meaning
skNearPoint	the Spacing property signifies the distance between control and nearest point of attached label;
skFarPoint	the Spacing property signigies the distance between control and far point of label.

Screenshot

© 1998-2006, UtilMind Solutions®



See also
Spacing and Position properties.

10.25.2.2.17 Transparent

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property Transparent: Boolean;
```

Description
The Transparent property specifies whether controls that sit below the label on a form can be seen through the label.

Set Transparent to True to prevent the label from obscuring other controls on the form. For example, if the label is used to add text to a graphic, set Transparent to True so that the label does not stand out as a separate object.

Writing text so that the background is transparent is slower than writing text when Transparent is False. If the label is not obscuring a complicated image, performance can be improved by setting the background color of the label to match the object beneath it and setting Transparent to False.

See also
Color property.

10.25.2.2.18 Visible

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property Visible: Boolean;
```

Description
The Visible property determines whether the label attached to the control appears on screen.

Use the Visible property to control the visibility of the control at runtime. If Visible is true, the control appears. If Visible is false, the control is not visible.

10.25.2.2.19 WordWrap

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property WordWrap: Boolean;
```

© 1998-2006, UtilMind Solutions®

Description

The WordWrap property specifies whether the label text wraps when it is too long for the width of the label.

Set WordWrap to True to allow the label to display multiple line of text. When WordWrap is True, text that is too wide for the label control wraps at the right margin and continues in additional lines.

Set WordWrap to False to limit the label to a single line. When WordWrap is False, text that is too wide for the label appears truncated.

See also

Caption property.

10.25.2.3AutoFill

Applies to

acMRUComboBox component.

Declaration

```
property AutoFill: Boolean;
```

Description

The AutoFill property controls whether you would like to auto-complete the input with the string taken from the history list. Set AutoFill to True to allow the auto-completing, or False otherwise.

For example, the list contains the line "Hello world!". At once the user type the "H" character, the combobox will try to put missing part of this line "ello world!".

See also

AutoFillDelay and AutoFillCaseSensitive properties.

10.25.2.4AutoFillCaseSensitive

Applies to

acMRUComboBox component.

Declaration

```
property AutoFillCaseSensitive: Boolean;
```

Description

The AutoFillCaseSensitive property controls whether the characters case does not matter and whether you would like to auto-complete the input disregarding the case of characters.

For example, the drop-down list contains the line "Apple". If the AutoFillCaseSensitive is False and user type the "a", the compobox will try to put missing part of line "pple", disregarding of the charcase of the line in the history list. In case if the AutoFillCaseSensitive is True, the combobox will try to find the line with exactly same case of characters.

See also

AutoFill and AutoFillDelay properties.

© 1998-2006, UtilMind Solutions®

10.25.2.5AutoFillDelay

Applies to
acMRUComboBox component.

Declaration
```
property AutoFillDelay: Integer; // in milliseconds
```

Description
The AutoFillDelay property speicifes the delay, in milliseconds, which should pass between keypress and attempt to auto-complete the input.

By default AutoFillDelay = 500 milliseconds, that means that when user typed the character and delayed on half seconds, the combobox will try to autocomplete the input with the line taken from the MRU history list.

See also
AutoFill and AutoFillCaseSensitive properties.

10.25.2.6MRUEnabled

Applies to
acMRUComboBox components.

Declaration
```
property MRUEnabled: Boolean;
```

Description
The MRUEnabled property controls whether the MRU history list should be automatically restored from the registry after creation of the control (i.e.: on program startup or creation of the form) and could be saved when the user moves the focus to another control.

See also
MRUEnabled, MRUMax, MRURegKey, MRURegLocation and MRUSaveOnExit properties.

10.25.2.7MRUMax

Applies to
acMRUComboBox components.

Declaration
```
property MRUMax: Integer; // default 20
```

Description
The MRUMax property specifies the maximum number of items which can contains the MRU history list.

See also
MRUEnabled, MRURegKey, MRURegLocation and MRUSaveOnExit properties.

10.25.2.8MRURegKey

Applies to
acMRUComboBox components.

Declaration
```
property MRURegKey: String;
```

© 1998-2006, UtilMind Solutions®

Description
The MRURegKey property defines the path to the key in the system registry where you would like to store the MRU history list.

The MRU list will be automatically saved when user moves the focus to another control and MRUSaveOnExit property is True, or after calling the Save method; and restored on program startup, if MRUEnabled is True.

Default value is `'\Software\CompanyName\ProgramName\MRUComboBox'`. You should point the MRURegKey to location where stored the configuration settings of your own application. In example - `'\Software\UtilMind Solutions\The Cool Program\URLs list'`.

See also
MRUEnabled, MRUMax, MRURegLocation and MRUSaveOnExit properties.

10.25.2.9 MRURegLocation

Applies to
acMRUComboBox components.

Declaration
```
property MRURegLocation: TacRegLocation;
```

Description
The MRURegLocation specifies the root key of the hierarchy of subkeys, where you would like to store the MRU history list. RegLocation property can point to HKEY_CURRENT_USER (rlCurrentUser) or HKEY_LOCAL_MACHINE (rlLocalMachine) root keys.

When MRURegLocation is rlCurrentUser, the MRUComboBox stores all settings in the registry section for current logged user. When RegLocation is rlLocalMachine settings will be saved for ALL users of current machine.

See also
MRUEnabled, MRUMax, MRURegKey and MRUSaveOnExit properties.

10.25.2.10 MRUSaveOnExit

Applies to
acMRUComboBox components.

Declaration
```
property MRUSaveOnExit: Boolean;
```

Description
The MRUSaveOnExit property controls whether the combobox should automatically add the current text from the edit box to the MRU history list, and save the MRU list to the registry, when user moves the focus to another control.

See also
MRUEnabled, MRUMax, MRURegKey and MRURegLocation properties.

© 1998-2006, UtilMind Solutions®

10.25.2.11 SubmitButton

Applies to
acMRUComboBox components.

Declaration
```
property SubmitButton: TControl;
```

Description
The SubmitButton property used to point to some control (button), which should automatically trigger the OnSubmit event of combo-box when user clicks it.

This is just optional property, just specify the SubmitButton control and the combo box will automatically save the typed string to the MRU history list when user clicks that button.

See also
OnSubmit event.

10.25.3 Methods

10.25.3.1 ClearMRU

Applies to
acMRUComboBox component.

Declaration
```
procedure ClearMRU;
```

Description
The ClearMRU method removes all items from the history of component (from drop-down list), and clears them from registry (deletes MRURegKey registry key).

See also
MRUEnabled, MRURegKey and MRUSaveOnExit properties.

10.25.3.2 Save

Applies to
acMRUComboBox component.

Declaration
```
procedure Save;
```

Description
The Save method moves the current text from the edit box to the MRU history list and saves the MRU list to the system registry.

If the MRUEnabled is True, the MRU list will be automatically restored from registry on program startup.

See also
MRUEnabled and MRUSaveOnExit properties;
ClearMRU method.

© 1998-2006, UtilMind Solutions®

10.25.3.3SetSelection

Applies to
acEdit, acNumberEdit, acMRUComboBox, acLabeledComboBox and acTreeComboBox components.

Declaration
```
procedure SetSelection(SelStart: Integer = 0; SelEnd: Integer = -1);
```

Description
The SetSelection method used to select a range of characters in an edit control.

SelStart parameter specifies the starting character position of the selection.

SelEnd parameter specifies the ending character position of the selection.

Remarks
If the *SelStart* parameter is 0 and the *SelEnd* parameter is -1, all the text in the edit control is selected. If *SelStart* is -1, any current selection is removed. The caret is placed at the end of the selection indicated by the greater of the two values *SelEnd* and *SelStart*.

10.25.4 Events

10.25.4.1OnMouseEnter

Applies to
acLabel, acURLLabel, acRichLabel, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox, acTreeComboBox, acHeader, acHeaderPanel, acTreeView and acListView components.

Declaration
```
property OnMouseEnter: TNotifyEvent;
```

Description
The OnMouseEnter occurs when mouse pointer hovers the combobox.

See also
OnMouseLeave event.

10.25.4.2OnMouseLeave

Applies to
acLabel, acURLLabel, acRichLabel, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox, acTreeComboBox, acHeader, acHeaderPanel, acTreeView and acListView components.

Declaration
```
property OnMouseLeave: TNotifyEvent;
```

Description
The OnMouseLeave occurs when mouse pointer leaves the combobox.

See also
OnMouseEnter event.

© 1998-2006, UtilMind Solutions®

10.25.4.3OnSubmit

Applies to
acMRUComboBox component.

Declaration

```
type
  TacMRUComboBoxSubmitEvent = procedure(Sender: TObject; var Text:
String; var SaveToMRU: Boolean) of object;

property OnSubmit: TacMRUComboBoxSubmitEvent;
```

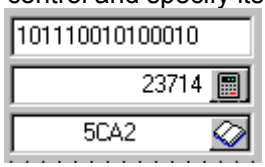
Description
The OnSubmit event occurs when user press the Enter key when the combo box is focused, OR if user clicks the control specified in SubmitButton property.

See also
SubmitButton property.

10.26 acNumberEdit

10.26.1 TacNumberEdit

Overview
The acNumberEdit component is the enchanced edit control control used for input of the numerical values in decimal, hexadecimal and binary format. The acNumberEdit can align the text by center or right edge control and able to have the button with bitmap image, at the right side of edit control. Also it contains the AttachedLabel structure which lets to operate with the label associated with the edit control and specify its behavior and position near the control.



Additionally, the acNumberEdit can automatically change the background color on disabled state (see ColorDisabled property) and supports custom cursor images for non-client area of edit control (see CursorBorder property).

Additional and improved properties

Align	controls placement of edit control within its container or parent control;
Alignment	controls horizontal placement of text within the edit control;
AttachedLabel	controls the behavior of the label attached to the control and its position;
AutoSelect	whether the acEdit should automatically select the line when user moves to the edit control pressing the *Tab* key;
Button	settings for custom button at the right edge of edit control;
ColorDisabled	background color of edit control at disabled state (when Enabled = False);
CursorBorder	cursor image when mouse pointer passes the border of edit control;
Kind	selects the numerical format to edit the number or convert Value to the Text;
MaxValue	specifies the maximum allowed value;
Value	represents the integer value of text typed in the edit control.

New events

OnButtonClick	occurs when user clicks the button at the right side of control;

© 1998-2006, UtilMind Solutions®

OnLabelClick	occurs when user clicks the label attached to the control;
OnLabelDblClick	occurs when user double clicks the label attached to the control;
OnMouseEnter	occurs when mouse pointer passes over the edit control;
OnMouseLeave	occurs when mouse pointer leaves the edit control.

See also
acEdit and acIPEdit components.

10.26.2 Properties

10.26.2.1Align

Applies to
acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acIPEdit and acMRUComboBox components *(in generally to ALL successors of TControl)*.

Declaration

```
type
  TAlign = (alNone, alTop, alBottom, alLeft, alRight, alClient);

property Align: TAlign;
```

Declaration
The Align property determines how the control aligns within its container (or parent control).

Use Align to align a control to the top, bottom, left, or right of a form or panel and have it remain there even if the size of the form, panel, or component that contains the control changes.

For example, to use a panel component with various controls on it as a tool palette, change the panel's Align value to alLeft. The value of alLeft for the Align property of the panel guarantees that the tool palette remains on the left side of the form and always equals the client height of the form.

The default value of Align is alNone, which means a control remains where it is positioned on a form or panel. Specify another value to align the control. Use one of these possible values:

Value	Meaning
alNone	The component remains where it was placed on the form or panel. This is the default value.
alTop	The component moves to the top of the form and resizes to fill the width of the form. The height of the component is not affected.
alBottom	The component moves to the bottom of the form and resizes to fill the width of the form. The height of the component is not affected.
alLeft	The component moves to the left side of the form and resizes to fill the height of the form. The width of the component is not affected.
alRight	The component moves to the right side of the form and resizes to fill the height of the form. The width of the component is not affected.
alClient	The component resizes to fill the client area of a form. If a component already occupies part of the client area, the component resizes to fit within the remaining client area.

Note
If Align is set to alClient, the control fills the entire client area so that it is impossible to select the parent form by clicking on it. In this case, select the parent by selecting the control on the form and

© 1998-2006, UtilMind Solutions®

pressing Esc, or by using the Object Inspector.

Any number of child components within a single parent can have the same Align value, in which case they stack up along the edge of the parent form or panel. The child controls stack up in z-order. To adjust the order in which the controls stack up, drag the controls into their desired positions.

10.26.2.2Alignment

Applies to
acNumberEdit, acEdit and acIPEdit components.

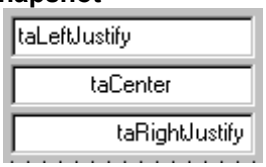
Declaration

```
type
  TAlignment = (taLeftJustify, taRightJustify, taCenter);

property Alignment: TAlignment;
```

Description
The Alignment property controls horizontal placement of the text within the edit control. Set Alignment to specify how the text of the edit control is aligned within the client area (ClientRect property) of this control. Alignment can be one of the following values:

Value	Meaning
taLeftJustify	The text appears lined up along the left edge of the edit control;
taRightJustify	The text appears lined up along the right edge of the edit control;
taCenter	The text is horizontally centered in the edit control.

Snapshot



10.26.2.3AttachedLabel

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox and acImagesComboBox components.

Unit
acAttachedLabel

Declaration

```
type
  TacAttachedLabelPosition = (lpAbove, lpBelow, lpLeft, lpRight);
  TacAttachedLabelSpacingKind = (skNearPoint, skFarPoint);
  TacAttachedLabel = class
    property Caption: TCaption;
    property Color: TColor default clBtnFace;
    property Cursor: TCursor default crDefault;
    property Enabled: Boolean;
    property FocusOnClick: Boolean default True;
    property Font: TFont;
    property Hint: String;
    property ParentColor: Boolean default True;
```

© 1998-2006, UtilMind Solutions®

```
    property ParentFont: Boolean default True;
    property ParentShowHint: Boolean default True;
    property PopupMenu: TPopupMenu;
    property Position: TacAttachedLabelPosition default lpLeft;
    property ShowAccelChar: Boolean default True;
    property ShowHint: Boolean default False;
    property Spacing: Integer default 4;
    property SpacingKind: TacAttachedLabelSpacingKind default
skNearPoint;
    property Transparent: Boolean default False;
    property Visible: Boolean;
    property WordWrap: Boolean default False;
  end;
```

Description

The AttachedLabel structure lets you to operate with the label attached to the some side of control (at the left or right side, above or below). You can specify the behaviour of label, its position near the control, and accelerator key for the control (character in caption, after ampersand (&)).

The label always associated with the control. When user press the accelerator key (Alt + Character, specified after ampersand (&)), the input focus will be imediately switched to the labeled control. The input focus also will be moved to the control if user clicks the label with mouse.

Screenshot (the labels at the left side is attached to the edit control):

See also

OnLabelClick and OnLabelDblClick events.

10.26.2.3.1 Caption

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property Caption: TCaption;
```

Description

The Caption property specified a text string which appers on label attached to control (attached label).

To underline a character in a Caption that labels a component, include an ampersand (&) before the character. This type of character is called an accelerator character. The user can then select the component by pressing Alt while typing the underlined character. To display an ampersand character in the caption, use two ampersands (&&).

The label always associated with the control. When user press the accelerator key (Alt + Character, specified after ampersand (&)), the input focus will be imediately switched to the labeled control.

Screenshot (the labels at the left side is attached to the edit control):

© 1998-2006, UtilMind Solutions®

See also
ShowAccelChar, Color, Font, Transparent, Visible and WordWrap properties.

10.26.2.3.2 Color

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property Color: TColor;
```

Description
The Color property specifies the background color for label attached to control. Use Color to read or change the background color of the control.

If a control's ParentColor property is true, then changing the Color property of the control's parent automatically changes the Color property of the control. When the value of the Color property is changed, the control's ParentColor property is automatically set to False.

If you don't want to show background — set Transparent property to True.

See also
ParentColor and Transparent properties.

10.26.2.3.3 Cursor

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property Cursor: TCursor;
```

Description
The Cursor property specifies the image used to represent the mouse pointer when it passes into the region covered by this label attached to control.

Change the value of Cursor to provide feedback to the user when the mouse pointer enters the control. The value of Cursor is the index of the cursor in the list of cursors maintained by the global variable, Screen. In addition to the built-in cursors provided by TScreen, applications can add custom cursors to the list.

See also
Additional Cursors

10.26.2.3.4 Enabled

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

© 1998-2006, UtilMind Solutions®

Declaration

```
property Enabled: Boolean;
```

Description

The Enabled property controls whether the control responds to mouse, keyboard, and timer events.

Use Enabled to change the availability of the control to the user. To disable a control, set Enabled to false. Disabled controls appear dimmed. If Enabled is false, the control ignores mouse, keyboard, and timer events.

To re-enable a control, set Enabled to true. The control is no longer dimmed, and the user can use the control.

See also

Caption and Color properties.

10.26.2.3.5 FocusOnClick

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property FocusOnClick: Boolean; // True by default
```

Description

The FocusOnClick property controls whether the control, associated with the attached label, should accept focus when user clicks the label with mouse.

Set FocusOnClick to True if you want to switch focus to the control when user clicks the label, or set it to False otherwise.

Note

Additionally you can use OnLabelClick and OnLabelDblClick events of the control.

10.26.2.3.6 Font

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property Font: TFont;
```

Description

The Font property controls the attributes of text written on the attached label of control.

To change to a new font, specify a new TFont object. To modify a font, change the value of the Charset, Color, Height, Name, Pitch, Size, or Style of the TFont object.

See also

ParentFont property.

10.26.2.3.7 Hint

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox

© 1998-2006, UtilMind Solutions®

and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property Hint: String;
```

Description

The Hint property contains the text string that can appear when the user moves the mouse over the label attached control.

If you are using the FormHelp component to show the context-sensitive help taken from secondary part of hint AND the Hint for attached label is not specified, the FormHelp will use the hint of main control.

See also

ShowHint and ParentShowHint properties;
acFormHelp and acFormHints components.

10.26.2.3.8 ParentColor

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property ParentColor: Boolean;
```

Description

The ParentColor determines where an attached label looks for its color information.

To have a control use the same color as its parent control, set ParentColor to true If ParentColor is false the control uses its own Color property.

Set ParentColor to true for all controls in order to ensure that all the controls on a form have a uniform appearance. For example, if ParentColor is true for all controls in a form, changing the background color of the form to gray causes all the controls on the form to also have a gray background.

When the value of a control's Color property changes, ParentColor becomes false automatically.

See also

Color and Transparent properties.

10.26.2.3.9 ParentFont

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property ParentFont: Boolean;
```

Description

The ParentFont property determines where a control looks for its font information.

To have a control use the same font as its parent control, set ParentFont to true If ParentFont is false the control uses its own Font property.

© 1998-2006, UtilMind Solutions®

Set ParentFont to true for all controls in order to ensure that all the controls on a form have a uniform appearance. For example, if ParentFont is true for all controls in a form, changing the form's Font property to 12-point Courier causes all controls on the form to use that font.

When the value of a control's Font property changes, ParentFont becomes false automatically.

When ParentFont is true for a form, the form uses the default font.

See also
Font property.

10.26.2.3.10 ParentShowHint

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property ParentShowHint: Boolean;
```

Description
The ParentShowHint property determines where an attached label looks to find out if its Help Hint should be shown.

Use ParentShowHint to ensure that all the controls on a form either uniformly show their Help Hints or uniformly do not show them.

If ParentShowHint is true, the control uses the ShowHint property value of its parent. If ParentShowHint is false, the control uses the value of its own ShowHint property.

To provide Help Hints for only selected controls on a form, set the ShowHint property for those controls that should have Help Hints to true, and ParentShowHint becomes false automatically.

Note
Enable or disable all Help Hints for the entire application using the ShowHint property of the application object.

See also
ShowHint and Hint properties;
acFormHelp and acFormHints components.

10.26.2.3.11 PopupMenu

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property PopupMenu: TPopupMenu;
```

Description
The PopupMenu property Identifies the pop-up menu associated with the label attached to control.

Assign a value to PopupMenu to make a pop-up menu appear when the user selects the label and clicks the right mouse button. If the TPopupMenu's AutoPopup property is True, the pop-up menu appears automatically. If the menu's AutoPopup property is False, display the menu with a call to its Popup method from the control's OnContextPopup event handler.

© 1998-2006, UtilMind Solutions®

10.26.2.3.12 Position

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
type
  TacAttachedLabelPosition = (lpAbove, lpBelow, lpLeft, lpRight);

property Position: TacAttachedLabelPosition;
```

Description
The Position property of the attached label specifies the layout of the label near the control.

You can make the label visible above, below, at the left or the right side of control.

See also
Spacing and SpacingKind properties.

10.26.2.3.13 ShowAccelChar

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property ShowAccelChar: Boolean;
```

Description
The ShowAccelChar property determines how an ampersand in the label text is displayed.

Set ShowAccelChar to True to allow the label to display an underlined accelerator key value. When ShowAccelChar is True, any character preceded by an ampersand (&) appears underlined. If the FocusControl property is set, the windowed control specified by the FocusControl property receives input focus when the user types that underlined character. To display an ampersand when ShowAccelChar is True, use two ampersands (&&) to stand for the single ampersand that is displayed.

Set ShowAccelChar to False to display the label text with all ampersands appearing as ampersands.

Example
This example uses two labels on a form. The first label has a caption with an accelerator character in it. The second label also includes an ampersand, but it does not appear as an accelerator character.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Label1.ShowAccelChar := True;
  Label1.Caption := 'An &Underlined character appears here';
  Label2.ShowAccelChar := False;
  Label2.Caption := 'An ampersand (&) appears here';
end;
```

See also

© 1998-2006, UtilMind Solutions®

Caption property.

10.26.2.3.14 ShowHint

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property ShowHint: Boolean;
```

Description
The ShowHint property determines whether the label attached to control displays a Help Hint when the mouse pointer rests momentarily on the label.

The Help Hint is the value of the Hint property, which is displayed in a box just beneath the control. Use ShowHint to determine whether a Help Hint appears for the control.

To enable Help Hint for a particular control, the application ShowHint property must be true and either the control's own ShowHint property must be true, or the control's ParentShowHint property must be true and its parent's ShowHint property must be true.

For example, imagine a check box within a group box. If the ShowHint property of the group box is true and the ParentShowHint property of the check box is true, but the ShowHint property of the check box is false, the check box still displays its Help Hint.

Changing the ShowHint value automatically sets the ParentShowHint property to false.

See also
Hint property.

10.26.2.3.15 Spacing

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property Spacing: Integer;
```

Description
The Spacing property determines the space (in pixels) between the control and attached label.

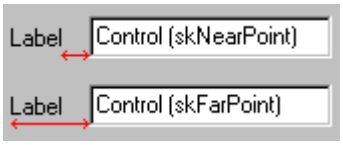
There is two spacing kinds which have an influence on distance between the control and label (see SpacingKind property).

When SpacingKind is `skNearPoint`, the Spacing property signifies the distance between control and nearest point of attached label.

When SpacingKind is `skFarPoint`, the Spacing property signigies the distance between control and far point of label.

Screenshot

© 1998-2006, UtilMind Solutions®



See also
SpacingKind and Position properties.

10.26.2.3.16 SpacingKind

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

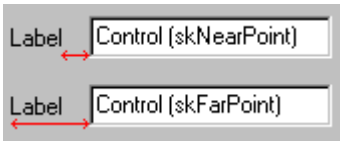
Declaration
```
type
  TacAttachedLabelSpacingKind = (skNearPoint, skFarPoint);

property SpacingKind: TacAttachedLabelSpacingKind;
```

Description
The SpacingKind property specifies how to measure the spacing between the control and attached label.

There is two possible values for SpacingKind property:

Value	Meaning
skNearPoint	the Spacing property signifies the distance between control and nearest point of attached label;
skFarPoint	the Spacing property signigies the distance between control and far point of label.

Screenshot



See also
Spacing and Position properties.

10.26.2.3.17 Transparent

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property Transparent: Boolean;
```

Description
The Transparent property specifies whether controls that sit below the label on a form can be seen through the label.

Set Transparent to True to prevent the label from obscuring other controls on the form. For example, if the label is used to add text to a graphic, set Transparent to True so that the label does not stand

© 1998-2006, UtilMind Solutions®

out as a separate object.

Writing text so that the background is transparent is slower than writing text when Transparent is False. If the label is not obscuring a complicated image, performance can be improved by setting the background color of the label to match the object beneath it and setting Transparent to False.

See also
Color property.

10.26.2.3.18 Visible

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property Visible: Boolean;
```

Description
The Visible property determines whether the label attached to the control appears on screen.

Use the Visible property to control the visibility of the control at runtime. If Visible is true, the control appears. If Visible is false, the control is not visible.

10.26.2.3.19 WordWrap

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property WordWrap: Boolean;
```

Description
The WordWrap property specifies whether the label text wraps when it is too long for the width of the label.

Set WordWrap to True to allow the label to display multiple line of text. When WordWrap is True, text that is too wide for the label control wraps at the right margin and continues in additional lines.

Set WordWrap to False to limit the label to a single line. When WordWrap is False, text that is too wide for the label appears truncated.

See also
Caption property.

10.26.2.4AutoSelect

Applies to
acEdit, acNumberEdit and acIPEdit components.

Declaration
```
property AutoSelect: Boolean; // True by default
```

Description
The AutoSelect property controls whether the text in the edit control should be automatically selected when user moves to the control pressing Tab key, activating input focus of the edit control.

© 1998-2006, UtilMind Solutions®

10.26.2.5Button

Applies to
acEdit, acNumberEdit and acTreeComboBox components.

Declaration
```
type
  TacEditButton = class(TPersistent)
  published
    property Cursor: TCursor;
    property Flat: Boolean; // in Delphi 3 and later !!!
    property Glyph: TBitmap;
    property NumGlyphs: TNumGlyphs; // 1..4
    property Hint: String;
    property Kind: TacEditButtonKind;
    property Width: Word;
    property Visible: Boolean; // False by default
  end;
```

Description
The acEdit component (and all its successors, like acNumberEdit or acTreeComboBox) is able to have a custom button at the right edge of edit control. The Button structure intended for specify settings for this button. To make button visible on edit control — set Button.Visible to True.

You can specify the glyph image and number of images, hint, cursor and width for button.

The button which at the right of edit control is just usual TSpeedButton component. You can even gain direct access to all button's properties using hidden **EditButton** property.
For example, to specify caption for this button you may use following code:
```
acNumberEdit1.EditButton.Caption := 'Test';
```

See also
OnButtonClick event.

10.26.2.5.1 Cursor

Applies to
acEdit, acNumberEdit and acTreeComboBox components as subproperty of Button structure.

Declaration
```
property Cursor: TCursor;
```

Description
The Cursor property specifies the image for mouse pointer when it passes over the custom button at the right side of edit control.

See also
CursorBorder property.

10.26.2.5.2 Flat

Applies to
acEdit, acNumberEdit and acTreeComboBox components as subproperty of Button structure.

Declaration
```
property Flat: Boolean;
```

© 1998-2006, UtilMind Solutions®

Description
The Flat property determines whether the button has a 3D border that provides a raised or lowered look.

Set Flat to True to remove the raised border when the button is unselected and the lowered border when the button is clicked or selected. When Flat is True, use separate bitmaps for the different button states to provide visual feedback to the user about the button state.

Remarks
This property available in Delphi 3 and later.

10.26.2.5.3 Glyph

Applies to
acEdit, acNumberEdit and acTreeComboBox components as subproperty of Button structure.

Declaration
```
property Glyph: TBitmap;
```

Description
The Glyph property specifies the bitmap that appears on the custom button at the right side of edit control.

Set Glyph to a bitmap object that contains the image that should appear on the face of the button. Bring up the Open dialog box from the Object Inspector to choose a bitmap file (with a .BMP extension), or specify a bitmap file at runtime.

Glyph can provide up to four images within a single bitmap. All images must be the same size and next to each other in a horizontal row. Button displays one of these images depending on the state of the button.

Image position	Buttons state	Description
First	Up	This image appears when the button is unselected. If no other images exist in the bitmap, this image is used for all states.
Second	Disabled	This image usually appears dimmed to indicate that the button can't be selected.
Third	Clicked	This image appears when the button is clicked. The Up image reappears when the user releases the mouse button.
Fourth	Down	This image appears when the button stays down indicating that it remains selected.

If only one image is present, Button attempts to represent the other states by altering the image slightly for each state, although the Down state is always the same as the Up state.

If the bitmap contains multiple images, specify the number of images in the bitmap with the NumGlyphs property.

Note
The lower left hand pixel of the bitmap is reserved for the "transparent" color. Any pixel in the bitmap which matches that lower left hand pixel will be transparent.

See also
NumGlyphs property.

© 1998-2006, UtilMind Solutions®

10.26.2.5.4 NumGlyphs

Applies to
acEdit, acNumberEdit and acTreeComboBox components as subproperty of Button structure.

Declaration
```
property NumGlyphs: TNumGlyphs; // 1..4
```

Description
The NumGlyphs is the number of images specified in the Glyph property.

Set NumGlyphs to the number of images provided by the bitmap assigned to the Glyph property. All images must be the same size and next to each other in a row. The Glyph property can provide up to four images.

See also
Glyph property.

10.26.2.5.5 Hint

Applies to
acEdit, acNumberEdit and acTreeComboBox components as subproperty of Button structure.

Declaration
```
property Hint: String;
```

Description
The Hint property contains the text string that can appear when the user moves the mouse pointer over the button at right side of edit control.

10.26.2.5.6 Kind

Applies to
acEdit, acNumberEdit and acTreeComboBox components as subproperty of Button structure.

Declaration
```
type
  TacEditButtonKind = (bkCustom, bkEllipsis, bkDropDown, bkFileOpen,
bkFileSave, bkBrowse, bkCalc);

property Kind: TacEditButtonKind;
```

Description
The Kind property determines the kind of bitmap button.

You can use custom image (Kind = bkCustom) and specify your sign to Glyph property, or use one of following images ( ), just switching the Kind property.

See also
Glyph and NumGlyphs properties.

10.26.2.5.7 Width

Applies to
acEdit, acNumberEdit and acTreeComboBox components as subproperty of Button structure.

Declaration

© 1998-2006, UtilMind Solutions®

```
property Width: Integer;
```

Description
The Width property controls the horizontal size of the button in pixels.

The button's height controlled by height of acNumberEdit control.

10.26.2.5.8 Visible

Applies to
acEdit, acNumberEdit and acTreeComboBox components as subproperty of Button structure.

Declaration
```
property Visible: Boolean;
```

Description
The Visible property determines whether the button appears at the right side of acNumberEdit control.

Use the Visible property to control the visibility of the button at runtime. If Visible is True, the button appears. If Visible is False, the button is not visible.

10.26.2.6Color

Applies to
acEdit, acNumberEdit and acIPEdit components (in general, to all successors of TControl class).

Declaration
```
property Color: TColor;
```

Description
The Color property for acNumberEdit component is the background color of the edit control. Use Color to read or change the background color of the control.

If a control's ParentColor property is True, then changing the Color property of the control's parent automatically changes the Color property of the control. When the value of the Color property is changed, the control's ParentColor property is automatically set to False.

See also
ColorDisabled property.

10.26.2.7ColorDisabled

Applies to
acEdit, acNumberEdit, acIPEdit and acTreeComboBox components.

Declaration
```
property ColorDisabled: TColor; // clBtnFace by default
```

Description
The ColorDisabled property for acNumberEdit component is the background color of the edit control at disabled state (when Enabled is False). Use the ColorDisabled property to specify color for disabled edit control.

See also
Color property.

© 1998-2006, UtilMind Solutions®

10.26.2.8 CursorBorder

Applies to
acEdit, acNumberEdit and acTreeComboBox components.

Declaration
```
property CursorBorder: TCursor;
```

Description
The CursorBorder property specifies the image used to represent the mouse pointer when it passes over the non-client area of edit control.

See also
acAppCursors component and Additional cursors in the IDE.

10.26.2.9 Kind

Applies to
acNumberEdit component.

Declaration
```
type
  TacNumberEditKind = (neDec, neHex, neBin);

property Kind: TacNumberEditKind;
```

Description
The Kind property selects the numerical format for editing in the acNumberEdit control. The format can be Decimal (`neDec`, by default), Hexadecimal (`neHex`) and Binary (`neBin`).

After specifying the Kind property, user will not able to enter characters which not used to represent the value in specified numerical format. For example, accepted chacters for decimal format is 0123456789; for hexadecimal is 0123456789ABCDEF, and 01 for binary format. The text specified in the Text property will be converted to according format as well.

See also
MaxValue and Value properties.
Numerical Conversion routines.

10.26.2.10 MaxValue

Applies to
acNumberEdit component.

Declaration
```
property MaxValue: LongInt;
```

Description
The MaxValue property specifies the maximum value which can be accepted by acNumberEdit control and assigned to the Value property. User will NOT able to enter value higher than specified in the MaxValue property.

See also
MaxValue and Value properties.

© 1998-2006, UtilMind Solutions®

Value

10.26.2.11

Applies to
acNumberEdit component.

Declaration
```
property Value: LongInt; // must be positive
```

Description
The Value property represents the integer value of text typed or specified in the Text property of acNumberEdit control. Use Value property to specify the the value to the edit control and the control will automatically convert it to the Text, in format specified by Kind property (decimal, hexadecimal or binary).

Remarks
The Value should be unsigned number only. If you'll try to specify value less than 0, the Value will be 0. If Value is higher than MaxValue, the Value will be put equal to MaxValue.

See also
Kind and MaxValue properties.

10.26.3 Methods

10.26.3.1 SetSelection

Applies to
acEdit, acNumberEdit, acMRUComboBox, acLabeledComboBox and acTreeComboBox components.

Declaration
```
procedure SetSelection(SelStart: Integer = 0; SelEnd: Integer = -1);
```

Description
The SetSelection method used to select a range of characters in an edit control.

SelStart parameter specifies the starting character position of the selection.

SelEnd parameter specifies the ending character position of the selection.

Remarks
If the *SelStart* parameter is 0 and the *SelEnd* parameter is -1, all the text in the edit control is selected. If *SelStart* is -1, any current selection is removed. The caret is placed at the end of the selection indicated by the greater of the two values *SelEnd* and *SelStart*.

10.26.4 Events

10.26.4.1 OnButtonClick

Applies to
acEdit, acNumberEdit and acTreeComboBox components.

Declaration
```
property OnButtonClick: TNotifyEvent;
```

Description
The OnButtonClick event occurs when user clicks the custom button at the right side of edit control.

© 1998-2006, UtilMind Solutions®

To make the button appear on edit control — set Button.Visible to True.

See also
Button structure.

10.26.4.2OnButtonLeftMouseDown

Applies to
acEdit, acNumberEdit and acTreeComboBox components.

Declaration
```
property OnButtonLeftMouseDown: TNotifyEvent;
```

Description
The OnButtonLeftMouseDown event occurs when user presses the left mouse button over the button at the right side of edit control.

To make the button appear on edit control — set Button.Visible to True.

See also
Button structure.

10.26.4.3OnChange

Applies to
acNumberEdit and acIPEdit components.

Declaration
```
property OnChange: TNotifyEvent;
```

Description
The OnChange event occurs whenever the value has been changed.

Write an OnChange event handler to take specific action after changing of the Value property. The Text property will already be updated to reflect any changes. This event provides the first opportunity to respond to modifications that the user types into the edit control.

See also
Value property.

10.26.4.4OnLabelClick

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and acTreeComboBox components.

Declaration
```
property OnLabelClick: TNotifyEvent;
```

Description
The OnLabelClick event occurs when user clicks the label attached to the control with a mouse.

To make the label appear near the edit control — set AttachedLabel.Visible to True, or to False if you'd like to hide the attached label.

See also

© 1998-2006, UtilMind Solutions®

AttachedLabel structure.

10.26.4.5OnLabelDblClick

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and acTreeComboBox components.

Declaration
```
property OnLabelDblClick: TNotifyEvent;
```

Description
The OnLabelDblClick event occurs when user double clicks the label attached to the control with a mouse.

To make the label appear near the edit control — set AttachedLabel.Visible to True, or to False if you'd like to hide the attached label.

See also
AttachedLabel structure.

10.26.4.6OnMouseEnter

Applies to
acLabel, acURLLabel, acRichLabel, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox, acTreeComboBox, acHeader, acHeaderPanel, acTreeView and acListView components.

Declaration
```
property OnMouseEnter: TNotifyEvent;
```

Description
The OnMouseEnter occurs when mouse pointer hover the edit control.

See also
OnMouseLeave event.

10.26.4.7OnMouseLeave

Applies to
acLabel, acURLLabel, acRichLabel, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox, acTreeComboBox, acHeader, acHeaderPanel, acTreeView and acListView components.

Declaration
```
property OnMouseLeave: TNotifyEvent;
```

Description
The OnMouseLeave occurs when mouse pointer leave the edit control.

See also
OnMouseEnter event.

© 1998-2006, UtilMind Solutions®

10.27 acPicturePanel

10.27.1 TacPicturePanel

Overview

The acPicturePanel is the hightly customizeable panel with caption and icon near the panel contents. This sort of panels very commonly used in the Microsoft-style dialogs.

Screenshot

How to use?

First and foremost, drop the acPicturePanel to your dialog box. Specify the text for title in Caption property (the text can contain rich-style tags, making the text partially colored or with another font style). The Caption text will appear on the special bevel-line

See also

10.27.2 Properties

10.27.2.1Background

Unit

acGraphics

Applies to

acPicturePanel, acHeader, acHeaderPanel and acSplitterPanel components.

Declaration

```
type
  TacBackground = class(TPersistent)
  published
    property Active: Boolean default False;
    property GradientColor1: TColor default clBlue;
    property GradientColor2: TColor default clBlack;
    property GradientDirection: TacGradientDirection default
gdHorizontal;
    property Kind: TacBackgroundKind default bkGradient;
    property Picture: TPicture;
    property PictureStyle: TacBackgroundPictureStyle default bsTile;
  end;
```

Description

© 1998-2006, UtilMind Solutions®

The Background structure used to specify custom background and its behaviors for the panel.

If you're tired of single-tone color panels and want to show the picture as "wallpaper", or smooth gradient effect under the controls of panel — simply activate it by specifying True to Active property and choose the Kind of background (specify which background you want: picture or gradient effect).

If you want to show the picture (wallpaper) — set Kind property to `bkPicture`, specify the image in Picture property and choose the style of wallpaper using PictureStyle property.

If you want to show the gradient effect instead of usual background — set Kind property to `bkGradient`, choose the style of effect in GradientDirection property (it can be vertical, horizontal or central), then specify the colors in GradientColor1 and GradientColor2 properties.

Alternatively, if you want to show your very own background / effect, you can use OnPaint event. Just write the OnPaint event handler and draw on the canvas of panel before all controls located on this panel will be displayed.

See also
OnPaint event.
acFormBackground component.

10.27.2.1.1 Active

Applies to
acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components as sub-property of Background structure.

Declaration
```
property Active: Boolean;
```

Description
The Active property of Background structure specifies whether the control should have custom background (gradient or picture).

Set Active to True to show the custom background (with smooth gradient effect or wallpaper specified in Picture property). Otherwise, set it to False to show standard single-tone background with color specified in Color property of control.

To specify the kind of background (gradient or picture) — use Kind property.

Alternatively, for better customization, you can use OnPaint event.

See also
Kind property;
OnPaint event.

10.27.2.1.2 GradientColor1

Applies to
acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components as sub-property of Background structure.

Declaration
```
property GradientColor1: TColor;
```

Description

© 1998-2006, UtilMind Solutions®

Specified the first (left if GradientDirection is horizontal or top if GradientDirection is vertical) gradient color of your form.

See also
GradientColor2, GradientDirection and Kind properties.

10.27.2.1.3 GradientColor2

Applies to
acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components as sub-property of Background structure.

Declaration
```
property GradientColor2: TColor;
```

Description
Specified the second (right if GradientDirection is horizontal or bottom if GradientDirection is vertical) gradient color of your form.

See also
GradientColor1, GradientDirection and Kind properties.

10.27.2.1.4 GradientDirection

Applies to
acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components as sub-property of Background structure.

Declaration
```
type
  TacGradientDirection = (gdVertical, gdHorizontal, gdCentral);

property GradientDirection: TacGradientDirection;
```

Description
Specifies the direction of the gradient effect. Direction can be vertical, horizontal or central. To specify the colors for gradient effect — use GradientColor1 and GradientColor2 properties.

Snapshots
```
GradientDirection = gdVertical
```

```
GradientDirection = gdHorizontal
```

```
GradientDirection = gdCentral
```

© 1998-2006, UtilMind Solutions®

See also
GradientColor1, GradientColor2 and Kind properties.

10.27.2.1.5 Kind

Applies to
acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components as sub-property of Background structure.

Declaration
```
type
  TacFormBackgroundKind = (bkPicture, bkGradient);

property Kind: TacFormBackgroundKind;
```

Description
The Kind property specifies how to paint the form background: using the picture (bkPicture), or drawing the gradient effect (bkGradient).

If you would like to paint background using the **picture**:
1. Set Kind property to `bkPicture`;
2. Specify the bitmap image in Picture property;
3. Select the style of drawing in the PictureStyle property.

If you wish to show the **gradient effect**:
1. Set Kind property to `bkGradient`;
2. Specify the direction of the effect in GradientDirection property;
3. Choose two colors to draw the effect in GradientColor1 and GradientColor2 properties.

See also
Active, GradientDirection and PictureStyle properties;
OnPaint event.

10.27.2.1.6 Picture

Applies to
acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components as sub-property of Background structure.

Declaration
```
property Picture: TPicture;
```

Description
The Picture property specifies the image that appears as form background. The Picture can contains an icon, betafile, bitmap graphic, or user-defined graphic to be displayed as the background (if Kind = bkPicture).

Setting Picture at design time brings up the Picture Editor, which can be used to specify the file that contains the image.

To specify the style of drawing — use PictureStyle property.

See also
PictureStyle and Kind properties.

© 1998-2006, UtilMind Solutions®

10.27.2.1.7 PictureStyle

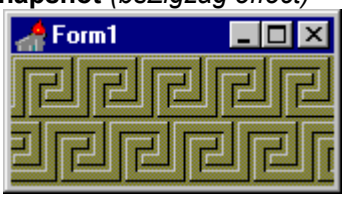
Applies to
acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components as sub-property of Background structure.

Declaration
```
property PictureStyle: TacBackgroundPictureStyle;
```

Description
The PictureStyle property specifies the effect used to draw the Picture. There are possible values:

Value	Meaning
bsTile	tiles the bitmap image on form's background;
bsStretch	stretches the image so that it exactly fits the bounds of the form. When the form resized, the image will be resized as well;
bsZigzag	almost the same as tile effect, but every next row of tile is shifted relatively to previous row, like bricks in the wall;
bsCenter	draws the image in the center of form.

Snapshot *(bsZigzag effect)*



See also
Picture and Kind properties.

10.27.2.2BevelStyle

Applies to
acPicturePanel component.

Declaration
```
type
  TBevelStyle = (bsLowered, bsRaised);

property BevelStyle: TBevelStyle;
```

Description
The BevelStyle property specifies the style for the line near the caption text. This line can be "lowered" (appears sunken) or "raised" (looks like light line with shadow).

See also
ShowBevel property.

10.27.2.3Image

Applies to
acPicturePanel component.

Declaration

© 1998-2006, UtilMind Solutions®

Description

See also

10.27.2.4ImageAlignment

Applies to
acPicturePanel component.

Declaration

```
type
  TacImageAlignment = (iaLeft, iaRight);

property ImageAlignment: TacImageAlignment;
```

Description

See also

10.27.2.5ImageIndex

Applies to
acPicturePanel component.

Declaration

Description

See also

10.27.2.6ImageList

Applies to
acPicturePanel component.

Declaration

```
property ImageList: TImageList;
```

Description

See also
Image, ImageAlignment, ImageIndex, ImageMargin and ImageTransparent properties.

10.27.2.7ImageMargin

Enter topic text here.

10.27.2.8ImageTransparent

Enter topic text here.

10.27.2.9Margin

Enter topic text here.

© 1998-2006, UtilMind Solutions®

ShowBevel
10.27.2.10

Applies to
acPicturePanel component.

Declaration
```
property ShowBevel: Boolean; // True by default
```

Description
The ShowBevel property specifies whether the line near the caption text should be visible on PicturePanel.

Set ShowBevel to False to hide the line from panel, or set it True to show the line near caption.

See also
BevelStyle property.

Text
10.27.2.11

Applies to
acPicturePanel component.

Declaration
```
property Text: String;
```

Description
The Text property specifies the text which

See also
TextAlignment and TextStyle properties.

TextAlignment
10.27.2.12

Applies to
acPicturePanel component.

Declaration
```
type
  TAlignment = (taLeftJustify, taRightJustify, taCenter);

property TextAlignment: TAlignment;
```

Description

See also
TextAlignment and TextStyle properties.

TextStyle
10.27.2.13

Applies to
acRichLabel, acFormHelp, acInfoBox, acHeader, acHeaderPanel and acPicturePanel components.

Declaration

© 1998-2006, UtilMind Solutions®

```
type
  TacTextStyle = class(TPersistent)
  published
    property PlainText: Boolean default False;
    property TagOpen: Char default '[';
    property TagClose: Char default ']';
  end;
```

Description

The TextStyle structure used to specify the style of text displayed by this component.

The PlainText property specifies whether you would like to use right-formatted text or just flat, plain text. TagOpen and TagClose properties used to specify prefered signs for brackets which identifies begin and end of tags used for rich formatting. By default every tag beginning with '**[**' sign and ending with '**]**' sign.

For example, when PlainText is False, TagOpen is '[' and TagClose is ']', following text:

```
[i]Hello[] [red][b]World[def]!
```

will looks like:

Hello **World**!

Otherwise, if PlainText is True, the tags will not be used and users will see just flat text, including all tags.

See also

Text and TextAlignment properties;
acRichLabel component and Rich-formatting tags overview.

10.27.2.13.1 PlainText

Applies to

acRichLabel, acInfoBox, acFormHelp, acHeader, acHeaderPanel and acPicturePanel components as subproperty of TextStyle structure.

Declaration

```
property PlainText: Boolean;
```

Description

The PlainText property controls whether the text should be displayed as rich text (if PlainText is False) or as plain, non-formatted text (if PlainText is True). When PlainText is True, the text will be displayed as usual flat text and nor tags will be used.

See also

TagOpen and TagClose properties;
Using rich-formatting tags topic.

10.27.2.13.2 TagClose

Applies to

acRichLabel, acInfoBox, acFormHelp, acHeader, acHeaderPanel and acPicturePanel components as subproperty of TextStyle structure.

Declaration

```
property TagClose: Char;
```

© 1998-2006, UtilMind Solutions®

Description
The TagClose property specifies the sign which marks the ending of formatting area of rich-text. The rich formatting tags of AppControls pack are similar to brackets "<>" of tags in HTML format. Default TagClose value is "**]**" sign.

Examples of standard tags
`[b]` - makes text after this tag **bold**.
`[i]` - makes text after this tag *italic*
`[biu]` - makes text after this tag ***bold, italic and underlined***.
`[]` - clears text formatting.
`[red]` - makes text after this tag red.
`[lime]` - makes text after this tag light green (lime).
`[def]` or `[default]` - returns the text formatting to default state as specified in Font property.

Note
If PlainText property is True, nor rich text will be displayed. All formating will be shown as usual flat text.

See also
TagOpen and PlainText properties and example of using rich-formatting tags.

10.27.2.13.3 TagOpen

Applies to
acRichLabel, acInfoBox, acFormHelp, acHeader, acHeaderPanel and acPicturePanel components as subproperty of TextStyle structure.

Declaration
```
property TagOpen: Char;
```

Description
The TagOpen property specifies the sign which marks the beginning of rich-text formatting area. The rich formatting tags of AppControls pack are similar to brackets "<>" of tags in HTML format. Default TagOpen value is "**[**" sign.

Examples of standard tags
`[b]` - makes text after this tag **bold**.
`[i]` - makes text after this tag *italic*
`[biu]` - makes text after this tag ***bold, italic and underlined***.
`[]` - clears text formatting.
`[red]` - makes text after this tag red.
`[lime]` - makes text after this tag light green (lime).
`[def]` or `[default]` - returns the text formatting to default state as specified in Font property.

See also
TagClose and PlainText properties and example of using rich-formatting tags.

10.27.3 Events

10.27.3.1 OnMouseEnter

Enter topic text here.

© 1998-2006, UtilMind Solutions®

10.27.3.2OnMouseLeave

Enter topic text here.

10.27.3.3OnPaint

Applies to
acHeader, acHeaderPanel, acSplitterPanel and acPicturePanel components.

Declaration
```
property OnPaint: TNotifyEvent;
```

Description
The OnPaint event occurs when Windows requires the panel to paint, such as when the form or panel receives focus or becomes visible when it wasn't previously.

Your application can use this event to draw on the canvas of the panel. For example, to draw custom background.

See also
Background structure.

10.28 acProportionalImage

10.28.1 TacProportionalImage

Overview
The acProportionalImage used to show the images with correct aspect ratio. With acProportionalImage you can be sure that the image will be fully displayed in the image control without any distortions.

Also this component allows to align the image within control (Alignment property), determinate the bounds of image wihin control (DrawRect property), can stretch image correctly (unlike standard TImage which contains a bug and break big images on stretching. See FixedStretching property), gives an access to properties of JPEG image at design-time (JPEGProperties structure).

How to use?
The same way as standard TImage component. Use Picture property to specify the actual bitmap, icon, metafile or other graphical object displayed by TImage. Additionally, TacProportionalImage introduces several new properties to determinate how the image is displayed within the boundaries of the control.

10.28.2 Properties

10.28.2.1Alignment

Applies to
acProportionalImage property.

Declaration
```
type
  TacImageAlignment = (iaCenter, iaLeft, iaTop, iaRight, iaBottom,
iaTopLeft, iaTopRight, iaBottomLeft, iaBottomRight);

property Alignment: TacImageAlignment;
```

Description

© 1998-2006, UtilMind Solutions®

The Alignment property controls how the image should be aligned within control (if it's not stretched).

There are possible values for Alignment property of acProportionalImage component:

Values	Meanings
iaCenter	aligns the image by center of control;
iaLeft	puts the image to the left side of control and aligns it by center vertically;
iaTop	puts the image to the top side of control and aligns it by center horizontally;
iaRight	puts the image to the right side of control and aligns it by center vertically;
iaBottom	puts the image to the bottom side of control and aligns it by center horizontally;
iaTopLeft	puts the image to top-left corner of control;
iaTopRight	puts the image to top-right corner of control;
iaBottomLeft	puts the image to bottom-left corner of control;
iaBottomRight	puts the image to bottom-right corner of control.

See also
DrawRect property.

10.28.2.2ControlCanvas

Applies to
acProportionalImage property.

Declaration
```
property ControlCanvas: TCanvas;
```

Description
The ControlCanvas is the read-only property which gives access to the canvas of Image control at run-time. Use ControlCanvas to draw something over image.

See also
DrawRect property.

10.28.2.3DrawRect

Applies to
acProportionalImage property.

Declaration
```
property DrawRect: TRect;
```

Description
The DrawRect is the *read-only* property used to determinate the bounds of rectangle where the picture has been drawn.

Example
```
// this code makes the client area size of the form equal to the
// size of picture, so the picture will exactly fit to the form size
with acProportionalImage1.DrawRect do
 begin
  ClientWidth := Right - Left;
  ClientHeight := Bottom - Top;
 end;
```

© 1998-2006, UtilMind Solutions®

See also
Proportional and ControlCanvas properties.

10.28.2.4FixedStretching

Applies to
acProportionalImage property.

Declaration
```
property FixedStretching: Boolean;
```

Description
When you use standard Stretch property (to display the image so it exactly fits the bounds of the image control), the image can be displayed incorrectly (part of image not shown, some pixels shifted etc). This is a bug of standard TImage component used in Delphi and C++ Builder *(or bug of StretchDraw routine in Win98, I'm not sure where the bug is, but I get tired of this)*.

When FixedStretching is True, the stretched image will be displayed correctly, using another routines for stretching image. If you want to display the *stretched* image the same way as it does the TImage component — set FixedStretching to False.

You can play with this feature at design-time to decide what do you like more:
1. Drop TacProportionalImage component onto your form and specify some image from disk in the Picture property;
2. Set Proportional property to False;
3. Set Stretched property to True;
*. Enable and disable FixedStretching property several times and make decision, use or not to use this property.

See also
Proportional property.

10.28.2.5JPEGProperties

Applies to
acProportionalImage property.

Declaration
```
type
  TacJPEGProperties = class(TPersistent)
  published
    property CompressionQuality: TJPEGQualityRange default 90;
    property Grayscale: Boolean default False;
    property Performance: TJPEGPerformance default jpBestQuality;
    property PixelFormat: TJPEGPixelFormat default jf24Bit;
    property ProgressiveDisplay: Boolean default False;
    property ProgressiveEncoding: Boolean default False;
    property Scale: TJPEGScale default jsFullSize;
    property Smoothing: Boolean default True;
  end;
```

Description
Use JPEGProperties structure to specify properties of JPEG images at design-time. These properties will work only if you have specified TJPEGImage object as Picture.

© 1998-2006, UtilMind Solutions®

10.28.2.5.1 CompressionQuality

Applies to
acProportionalImage as subproperty of JPEGProperties structure.

Declaration
```
type
  TJPEGQualityRange = 1..100;

property CompressionQuality: TJPEGQualityRange;
```

Description
Indicates the trade-off ratio between the image quality and the file size.

Use CompressionQuality to set the compression quality of the JPEG image when writing out a JPEG image. Higher compression results in a poorer picture quality, but a smaller file size. This property is not used for reading in files.

TJPEGQualityRange is the type of the CompressionQuality property. The higher the TJPEGQualityRange value (up to a maximum of 100), the better the image quality, but the larger the file size. The lower the TJPEGQualityRange value (to a minimum of 1), the smaller the resulting file size, but at the expense of picture quality.

See also
Grayscale, Performance, PixelFormat, ProgressiveDisplay, ProgressiveEncoding, Scale and Smoothing properties.

10.28.2.5.2 Grayscale

Applies to
acProportionalImage as subproperty of JPEGProperties structure.

Declaration
```
property Grayscale: Boolean;
```

Description
The Grayscale property determines whether the image output of a JPEG image is black and white, or color.

Use Grayscale for reading in and writing out, that is, for decompressing and compressing, a JPEG image. This property affects how the image is displayed.

Grayscale is used for speed in output optimization. When Grayscale is set to True, the color is separated from luminosity. Color takes the most time to decompress, so for previewing the image, Grayscale can be set to True for speed. The output then contains 255 shades of gray.

See also
CompressionQuality, Performance, PixelFormat, ProgressiveDisplay, ProgressiveEncoding, Scale and Smoothing properties.

10.28.2.5.3 Performance

Applies to
acProportionalImage as subproperty of JPEGProperties structure.

Declaration
```
type
```

© 1998-2006, UtilMind Solutions®

```
  TJPEGPerformance = (jpBestQuality, jpBestSpeed);

property Performance: TJPEGPerformance;
```

Description

Controls the trade-off between color quality and speed of decompression.

Use Performance to set the performance of the JPEG image when reading, that is, for decompressing files. This property is not used for writing out files.

Performance affects the display of the internal bitmap image. This image is a 256 color bitmap. Setting Performance to jpBestSpeed may cause some dithering in the bitmap image.

See also

CompressionQuality, Grayscale, PixelFormat, ProgressiveDisplay, ProgressiveEncoding, Scale and Smoothing properties.

10.28.2.5.4 PixelFormat

Applies to

acProportionalImage as subproperty of JPEGProperties structure.

Declaration

```
type
  TJPEGPixelFormat = (jf24Bit, jf8Bit);

property PixelFormat: TJPEGPixelFormat;
```

Description

Determines whether the JPEG image is displayed in 24 or 8-bit format.

Use PixelFormat to set the pixel format of the JPEG image to 8-bit for video drivers that cannot display 24-bit. the native format of a JPEG image is 24-bit. PixelFormat is used for decompression, that is, for reading in files. PixelFormat will also apply to a bitmap if the image is copied to it.

See also

CompressionQuality, Grayscale, Performance, ProgressiveDisplay, ProgressiveEncoding, Scale and Smoothing properties.

10.28.2.5.5 ProgressiveDisplay

Applies to

acProportionalImage as subproperty of JPEGProperties structure.

Declaration

```
property ProgressiveDisplay: Boolean;
```

Description

Controls how the image is displayed when decompressing files.

Use ProgressiveDisplay to turn on or off the incremental display of an image when reading in files. For example, if the JPEG data source file is on the hard drive, progressively displaying it slows down the display time: in this case setting ProgressiveDisplay to False speeds up the displaying of the image. If the file must be downloaded, set ProgressiveDisplay to True if the downloading process will take as long as the progressive displaying of the image.

© 1998-2006, UtilMind Solutions®

ProgressiveDisplay is used only for decompression, but is affected by the manner in which the image was compressed. Images that were compressed with ProgressiveEncoding set to False cannot take advantage of ProgressiveDisplay. Setting ProgressiveDisplay to True for these images has no effect.

See also
CompressionQuality, Grayscale, Performance, PixelFormat, ProgressiveEncoding, Scale and Smoothing properties.

10.28.2.5.6 ProgressiveEncoding

Applies to
acProportionalImage as subproperty of JPEGProperties structure.

Declaration
```
property ProgressiveEncoding: Boolean;
```

Description
Determines whether an image can be progressively displayed when it is decompressed.

Use ProgressiveEncoding to compress an image when writing out files, so that it can be progressively displayed when it is decompressed. Setting the ProgressiveDisplay property to True has no effect if ProgressiveEncoding was not set to True when the image was compressed.

See also
CompressionQuality, Grayscale, Performance, PixelFormat, ProgressiveDisplay, Scale and Smoothing properties.

10.28.2.5.7 Scale

Applies to
acProportionalImage as subproperty of JPEGProperties structure.

Declaration
```
type
  TJPEGScale = (jsFullSize, jsHalf, jsQuarter, jsEighth);

property Scale: TJPEGScale;
```

Description
Determines the size of the JPEG image when it is displayed.

Use Scale to optimize performance for previewing an image. jsFullSize displays the image as maximum size, whereas jsEight displays an eighth of the image size, but 8 times faster.

TJPEGScale is the type of the Scale property. The following table lists the values of TJPEGScale:

Value	Meaning
jsFullSize	Displays a full-size image.
jsHalf	Displays a half-size image in half the time it takes to display a full-size image.
jsQuarter	Displays a quarter-size image in a quarter of the time it takes to display a full-size image.
jsEighth	Displays an eighth-size image in an eighth of the time it takes to display a full-size image.

Scale it used for reading in files, that is, for decompression.

© 1998-2006, UtilMind Solutions®

See also
CompressionQuality, Grayscale, Performance, PixelFormat, ProgressiveDisplay, ProgressiveEncoding and Smoothing properties.

10.28.2.5.8 Smoothing

Applies to
acProportionalImage as subproperty of JPEGProperties structure.

Declaration
```
property Smoothing: Boolean;
```

Description
Determines whether the JPEG image displays in blocks or with blurred edges.

Smoothing is used if ProgressiveDisplay is True, to determine whether the JPEG image is displayed in small blocks or with blurred edges. If Smoothing is True, the edges are blurred during progressive displaying of the image. If Smoothing is False, raw blocks of color are used to display the image.

See also
CompressionQuality, Grayscale, Performance, PixelFormat, ProgressiveDisplay, ProgressiveEncoding and Scale properties.

10.28.2.6Proportional

Applies to
acProportionalImage property.

Declaration
```
property Proportional: Boolean;
```

Description
The Proportional property indicates whether the image should be changed, without distortion, so that it fits the bounds of the image control.

Set Proportional to True to ensure that the image can be fully displayed in the image control without any distortion such as occurs with the Stretch property. When Proportional is True, images that are too large to fit in the image control are scaled down (while maintaining the same aspect ratio) until they fit in the image control. Images that are too small are displayed normally. That is, Proportional can reduce the magnification of the image, but does not increase it.

When the image control resizes, the image resizes also.

To resize the image so that it fits exactly in the image control, even if that causes distortion, use the Stretch property instead.

To resize the control to the image rather than resizing the image to the control, use the AutoSize property instead.

Note: Proportional has no effect if the Picture property contains an icon.

Use DrawRect property to determinate the bounds of drawn image (not image control).

See also
AutoSize, Stretch and DrawRect properties.

© 1998-2006, UtilMind Solutions®

10.29 acQuickAboutBox

10.29.1 TacQuickAboutBox

Overview

With acQuickAboutBox component you can quickly add simple but cool looking "about" box for your freeware and shareware application. Just drop component onto your main form, set several properties such like program name, short description, copyright notice, email address and URL to your website and About box is done. To show it — just call Execute method. You can see this dialog box as About property in any component of AppControls pack (click right mouse button on component image and select "About…" menu item).

Features

help is under construction...

You may doubleclick the component image at design-time to test the appearance of about box.

Snapshot

10.29.2 Properties

10.29.2.1AnimationDelay

Applies to

acQuickAboutBox component.

Declaration

```
property AnimationDelay: Word;
```

Description

10.29.2.2AnimationEffect

Applies to

acQuickAboutBox component.

Declaration

```
property AnimationEffect: Boolean;
```

Description

© 1998-2006, UtilMind Solutions®

10.29.2.3Caption

Applies to
acQuickAboutBox component.

Declaration
```
property Caption: String;
```

Description

10.29.2.4Copyright

Applies to
acQuickAboutBox component.

Declaration
```
property Copyright: TStrings;
```

Description

See also
Description property.

10.29.2.5Cursors

Applies to
acQuickAboutBox component.

Declaration
```
type
  TacQuickAboutBoxFormCursors = class
  published
    property Caption: TCursor;
    property Client: TCursor;
    property Border: TCursor;
    property BtnClose: TCursor;
    property Buttons: TCursor;
    property Links: TCursor;
  end;
```

Description

10.29.2.5.1 Caption

Applies to
acQuickAboutBox component.

Declaration
```
property Caption: TCursor;
```

Description

See also
Caption, Client, Border, BtnClose, Buttons, Links cursors properties.

© 1998-2006, UtilMind Solutions®

10.29.2.5.2 Client

Applies to
acQuickAboutBox component.

Declaration
```
property Client: TCursor;
```

Description

See also
Caption, Client, Border, BtnClose, Buttons, Links cursors properties.

10.29.2.5.3 Border

Applies to
acQuickAboutBox component.

Declaration
```
property Border: TCursor;
```

Description

See also
Caption, Client, Border, BtnClose, Buttons, Links cursors properties.

10.29.2.5.4 BtnClose

Applies to
acQuickAboutBox component.

Declaration
```
property BtnClose: TCursor;
```

Description

See also
Caption, Client, Border, BtnClose, Buttons, Links cursors properties.

10.29.2.5.5 Buttons

Applies to
acQuickAboutBox component.

Declaration
```
property Buttons: TCursor;
```

Description

See also
Caption, Client, Border, BtnClose, Buttons, Links cursors properties.

10.29.2.5.6 Links

Applies to
acQuickAboutBox component.

© 1998-2006, UtilMind Solutions®

Declaration
```
property Links: TCursor;
```

Description

See also
Caption, Client, Border, BtnClose, Buttons, Links cursors properties.

10.29.2.6 Description

Applies to
acQuickAboutBox component.

Declaration
```
property Description: TStrings;
```

Description

See also
Copyright property.

10.29.2.7 Email

Applies to
acQuickAboutBox component.

Declaration
```
property Email: String;
```

Description

See also
WebSite property.

10.29.2.8 GradientBackground

Applies to
acQuickAboutBox component.

Declaration
```
type
  TacQuickAboutBoxGradientBackground = class
  published
    property Color1: TColor;
    property Color2: TColor;
    property Direction: TacGradientDirection;
    property Enabled: Boolean;
  end;
```

Description

10.29.2.8.1 Color1

Applies to
acQuickAboutBox component.

© 1998-2006, UtilMind Solutions®

Declaration
```
property Color1: TColor;
```

Description

See also
Color2 property.

10.29.2.8.2 Color2

Applies to
acQuickAboutBox component.

Declaration
```
property Color2: TColor;
```

Description

See also
Color1 property.

10.29.2.8.3 Direction

Applies to
acQuickAboutBox component.

Declaration
```
type
  TacGradientDirection = (gdVertical, gdHorizontal);

property Direction: TacGradientDirection;
```

Description

10.29.2.8.4 Enabled

Applies to
acQuickAboutBox component.

Declaration
```
property Enabled: Boolean;
```

Description

10.29.2.9Icon

Applies to
acQuickAboutBox component.

Declaration
```
property Icon: TIcon;
```

Description

© 1998-2006, UtilMind Solutions®

Localization
10.29.2.10

Applies to
acQuickAboutBox component.

Declaration
```
type
  TacQuickAboutBoxLocalization = class
  published
    property stAbout: String;
    property stOK: String;
    property stRegister: String;
    property stRegisteredTo: String;
    property stUnregistered: String;
  end;
```

Description

10.29.2.10.1 stAbout

Applies to
acQuickAboutBox component as subproperty of Localization.

Declaration
```
property stAbout: String;
```

Description

See also
stAbout, stOK, stRegister, stRegisteredTo, stUnregistered properties.

10.29.2.10.2 stOK

Applies to
acQuickAboutBox component as subproperty of Localization.

Declaration
```
property stOK: String;
```

Description

See also
stAbout, stOK, stRegister, stRegisteredTo, stUnregistered properties.

10.29.2.10.3 stRegister

Applies to
acQuickAboutBox component as subproperty of Localization.

Declaration
```
property stRegister: String;
```

Description

© 1998-2006, UtilMind Solutions®

See also
stAbout, stOK, stRegister, stRegisteredTo, stUnregistered properties.

10.29.2.10.4 stRegisteredTo

Applies to
acQuickAboutBox component as subproperty of Localization.

Declaration
```
property stRegisteredTo: String;
```

Description

See also
stAbout, stOK, stRegister, stRegisteredTo, stUnregistered properties.

10.29.2.10.5 stUnregistered

Applies to
acQuickAboutBox component as subproperty of Localization.

Declaration
```
property stUnregistered: String;
```

Description

See also
stAbout, stOK, stRegister, stRegisteredTo, stUnregistered properties.

Registered
10.29.2.11

Applies to
acQuickAboutBox component.

Declaration
```
property Registered: Boolean;
```

Description

See also
RegisteredTo, RegistrationURL, Shareware properties.

RegisteredTo
10.29.2.12

Applies to
acQuickAboutBox component.

Declaration
```
property RegisteredTo: String;
```

Description

See also

© 1998-2006, UtilMind Solutions®

Registered, RegistrationURL, Shareware properties.

RegistrationURL
10.29.2.13

Applies to
acQuickAboutBox component.

Declaration
```
property RegistrationURL: String;
```

Description

See also
Registered, RegisteredTo, Shareware properties.

Shareware
10.29.2.14

Applies to
acQuickAboutBox component.

Declaration
```
property Shareware: Boolean;
```

Description

See also
Registered, RegisteredTo, RegistrationURL properties.

ShowBevels
10.29.2.15

Applies to
acQuickAboutBox component.

Declaration
```
property ShowBevels: Boolean;
```

Description

ShowHints
10.29.2.16

Applies to
acQuickAboutBox component.

Declaration
```
property ShowHints: Boolean;
```

Description

© 1998-2006, UtilMind Solutions®

WebSite
10.29.2.17

Applies to
acQuickAboutBox component.

Declaration
```
property WebSite: String;
```

Description

See also
Email property.

Width
10.29.2.18

Applies to
acQuickAboutBox component.

Declaration
```
property Width: Integer; // default 260
```

Description

10.29.3 Methods

10.29.3.1 Execute

Applies to
acQuickAboutBox component.

Declaration
```
procedure Execute;
```

Description

10.30 acRadioButton

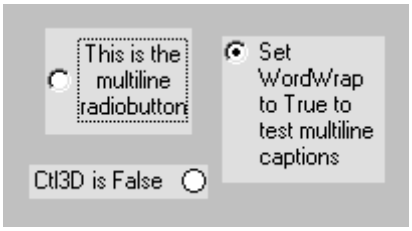
10.30.1 TacRadioButton

Overview
The acRadioButton component is the improved TRadioButton, which supports multiline captions and able to have non-3D (flat) border for the radio button. Contains all needed properties for flexible adjustment of multiline text within client area of control and aligment withing its container.

Since this is just enchancement of standard TRadioButton, you can find out about its basic features in the Delphi/BCB documentation (but you probably already know how to use RadioButtons. If not — select necessary property in Object Inspector and press F1). This documentation descibes only additional features and improvements.

Snapshot

© 1998-2006, UtilMind Solutions®



To specify multiline caption — set WordWrap property to True. To make the flat checkbox — use Ctl3D property.

Remark

The acRadioButton is incompatible with standard TRadioButton. Do not place the TRadioButton's together with TacRadioButtons to one group.

Additional and improved properties

Align	controls placement of button within its container;
AlignmentHorz	specifies horizontal alignment of text within client area of control;
AlignmentVert	specifies vertical alignment of text within client area of control;
BoxAlignment	controls placement of the checkbox relatively to the caption;
Ctl3D	determines whether a checkbox has a three-dimensional (3-D) or two-dimensional look;
ParentCtl3D	determines whether a checkbox has same 3-D look as its container;
WordWrap	specifies whether the text on checkbox wraps when it is too long for the width of the control. When WordWrap is True, caption can be multiline.

Events

OnMouseEnter	occurs when mouse pointer hovers the radiobutton;
OnMouseLeave	occurs when mouse pointer leaves the radiobutton.

See also

acCheckBox component.

10.30.2 Properties

10.30.2.1Align

Applies to

acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acIPEdit and acMRUComboBox components *(in generally to ALL successors of TControl)*.

Declaration

```
type
  TAlign = (alNone, alTop, alBottom, alLeft, alRight, alClient);

property Align: TAlign;
```

Declaration

The Align property determines how the control aligns within its container (or parent control).

Use Align to align a control to the top, bottom, left, or right of a form or panel and have it remain there even if the size of the form, panel, or component that contains the control changes.

For example, to use a panel component with various controls on it as a tool palette, change the

© 1998-2006, UtilMind Solutions®

panel's Align value to alLeft. The value of alLeft for the Align property of the panel guarantees that the tool palette remains on the left side of the form and always equals the client height of the form.

The default value of Align is alNone, which means a control remains where it is positioned on a form or panel. Specify another value to align the control. Use one of these possible values:

Value	Meaning
`alNone`	The component remains where it was placed on the form or panel. This is the default value.
`alTop`	The component moves to the top of the form and resizes to fill the width of the form. The height of the component is not affected.
`alBottom`	The component moves to the bottom of the form and resizes to fill the width of the form. The height of the component is not affected.
`alLeft`	The component moves to the left side of the form and resizes to fill the height of the form. The width of the component is not affected.
`alRight`	The component moves to the right side of the form and resizes to fill the height of the form. The width of the component is not affected.
`alClient`	The component resizes to fill the client area of a form. If a component already occupies part of the client area, the component resizes to fit within the remaining client area.

Note

If Align is set to alClient, the control fills the entire client area so that it is impossible to select the parent form by clicking on it. In this case, select the parent by selecting the control on the form and pressing Esc, or by using the Object Inspector.

Any number of child components within a single parent can have the same Align value, in which case they stack up along the edge of the parent form or panel. The child controls stack up in z-order. To adjust the order in which the controls stack up, drag the controls into their desired positions.

10.30.2.2AlignmentHorz

Applies to

acRadioButton and acCheckBox components.

Declaration

```
type
  TAlignment = (taLeftJustify, taRightJustify, taCenter);

property AlignmentHorz: TAlignment;
```

Description

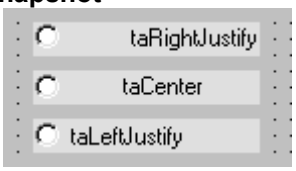
The AlignmentHorz property controls horizontal placement of text string that appears as a label for radiobutton control.

By default, text is lined up along the left edge of control (near the check box, if BoxAlignment is `taLeftJustify` as well). To make text lined up along the right edge — set AlignmentHorz to `taRightJustify`. For centering the text horziontally within a client area of control — set AlignmentHorz to `taCenter`.

The effect of the AlignmentHorz property is more obvious if the WordWrap property is True and the caption includes more than one line of text.

© 1998-2006, UtilMind Solutions®

Snapshot



See also

AlignmentVert property.

10.30.2.3AlignmentVert

Applies to

acRadioButton and acCheckBox components.

Declaration

```
type
  TVerticalAlignment = (vaTopJustify, vaBottomJustify, vaCenter);

property AlignmentVert: TVerticalAlignment;
```

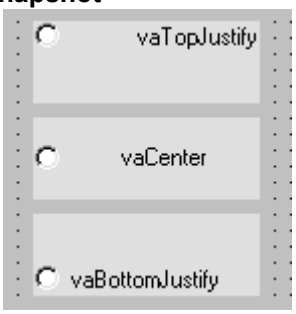
Description

The AlignmentVert property controls vertical placement of text string that appears as a label for radiobutton control.

By default, text is lined up along the top edge of control (`AlignmentVert = vaTopJustify`). To align text along the bottom edge — set AlignmentVert to `vaRightJustify`. For centering the text vertically within a client area — set AlignmentVert to `vaCenter`.

The effect of the AlignmentVert property is more obvious if the WordWrap property is True and the caption includes more than one line of text.

Snapshot



See also

AlignmentHorz property.

10.30.2.4BoxAlignment

Applies to

acRadioButton, acCheckBox and acCheckGroupBox components.

Declaration

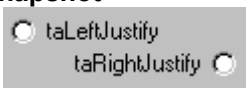
```
type
  TAlignment = (taLeftJustify, taRightJustify, taCenter);
  TLeftRight = taLeftJustify..taRightJustify;
```

© 1998-2006, UtilMind Solutions®

```
property BoxAlignment: TLeftRight;
```

Description
The BoxAlignment property specifies placement of the radio button, relatively to the caption on this control.

By default, radio button is left-aligned. When the BoxAlignment is `taRightJustify`, the button will be displayed to the right of text.

Snapshot


See also
AlignmentHorz and AlignmentVert properties.

10.30.2.5Ctl3D

Applies to
acRadioButton, acCheckBox and acCheckGroupBox components *(in generally, to ALL successors of TControl)*.

Declaration
```
property Ctl3D: Boolean;
```

Description
The Ctl3D property determines whether a control has a three-dimensional (3-D) or two-dimensional look.

Use Ctl3D to control the appearance of the control. If Ctl3D is True, the control has a 3-D appearance. If Ctl3D is False, the control appears flat. The default value of Ctl3D is True.

If a control's ParentCtl3D property is True, then changes in the Ctl3D property of the control's parent automatically changes the Ctl3D property of the control. When assigning a value directly to a control's Ctl3D property, the control's ParentCtl3D property is automatically set to False.

See also
ParentCtl3D property.

10.30.2.6ParentCtl3D

Applies to
acRadioButton and acCheckBox components *(in generally, to ALL successors of TControl)*.

Declaration
```
property ParentCtl3D: Boolean;
```

Description
The ParentCtl3D property determines where a radiobutton looks to determine if it should appear three dimensional.

Use ParentCtl3D to ensure that all the components on a form have a uniform appearance. For example, to make all components on a form appear three dimensional, set the form's Ctl3D property to True and each control's ParentCtl3D property to True. Not only will all components have a three-dimensional appearance, but to change to a uniform two-dimensional appearance, you only have to

© 1998-2006, UtilMind Solutions®

change the Ctl3D property of the form and all the components will become two dimensional.

If ParentCtl3D is True, the component uses the value of its parent component's Ctl3D property. If ParentCtl3D is False, the control uses the value of its own Ctl3D property. The default value is True.

To specify either a 2D or 3D appearance for a particular component, set Ctl3D to True (for 3D) or False (for 2D). When setting the Ctl3D property of a control, ParentCtl3D automatically gets set to False.

See also
Ctl3D property.

10.30.2.7WordWrap

Applies to
acRadioButton, acCheckBox, acLabel, acURLLabel, acRichLabel, acButton and acBitBtn components.

Declaration
```
property WordWrap: Boolean;
```

Description
The WordWrap property controls whether the text on radiobutton wraps when it is too long for the width of the control. When WordWrap is True, caption can be multiline, and single line only when WordWrap is False.

10.30.3 Events

10.30.3.1OnMouseEnter

Applies to
acLabel, acURLLabel, acRichLabel, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acMRUComboBox, acHeader, acHeaderPanel, acTreeView, acListView components.

Declaration
```
property OnMouseEnter: TNotifyEvent;
```

Description
The OnMouseEnter occurs when mouse pointer hovers the radiobutton control.

See also
OnMouseLeave event.

10.30.3.2OnMouseLeave

Applies to
acLabel, acURLLabel, acRichLabel, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acMRUComboBox, acHeader, acHeaderPanel, acTreeView, acListView components.

Declaration
```
property OnMouseLeave: TNotifyEvent;
```

Description
The OnMouseLeave occurs when mouse pointer leaves the radiobutton control.

© 1998-2006, UtilMind Solutions®

See also

OnMouseEnter event.

10.31 acRichLabel

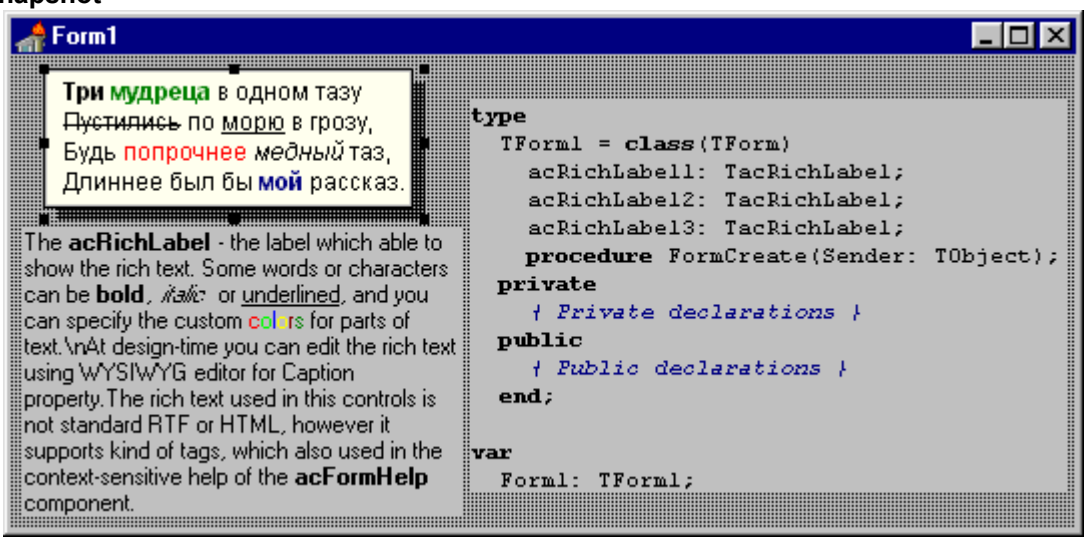
10.31.1 TacRichLabel

Overview

The acRichLabel component is the label which able to show the rich text. Some words or characters can be bold, italic or underlined, and you can specify the custom colors for various parts of text. At design-time you can edit the rich text using WYSIWYG editor for Caption property.

The rich text used in this controls is not standard RTF or HTML, however it supports kind of tags, which also used in the context-sensitive help of the acFormHelp component.

Snapshot



Additional / improved properties

Caption	specifies the text for label. You can use rich-style tags;
MarginHorz	;
MarginVert	;
ShadowColor	;
ShowFrame	;
ShowShadow	;
Transparent	;
WordWrap	whether the text should wraps to the next line if it doesn't fits to the control's bounds.

New events

OnMouseEnter	occurs when mouse pointer hovers the label;
OnMouseLeave	occurs when mouse pointer leaves the label.

See also

acFormHelp, acLabel, acURLLabel and acHTMLViewer components.

© 1998-2006, UtilMind Solutions®

10.31.2 Properties

10.31.2.1Caption

Applies to
acRichLabel, acCheckGroupBox, acButton and acBitBtn components *(to all visible controls, in general)*

Declaration
```
property Caption: String;
```

Description

See also

10.31.2.2MarginHorz

Applies to
acRichLabel component.

Declaration
```
property MarginHorz: Word;
```

Description

See also
MarginVert property.

10.31.2.3MarginVert

Applies to
acRichLabel component.

Declaration
```
property MarginHorz: Word;
```

Description

See also
MarginHorz property.

10.31.2.4ShadowColor

Applies to
acRichLabel and acFormHelp components.

Declaration
```
property ShadowColor: TColor;
```

Description
The ShadowColor property specifies the color of shadow (dots that imitate the window shadow) for popup windows with context-sensitive help. ShadowColor used only if ShowShadow property is True.

See also
ShowShadow property.

© 1998-2006, UtilMind Solutions®

10.31.2.5ShowFrame

Applies to
acRichLabel components.

Declaration
```
property ShowFrame: Boolean;
```

Description

See also
ShadowColor and Transparent properties.

10.31.2.6ShowShadow

Applies to
acRichLabel and acFormHelp components.

Declaration
```
property ShowShadow: Boolean;
```

Description

See also
ShadowColor, ShowFrame and Transparent properties.

10.31.2.7TextStyle

Applies to
acRichLabel, acFormHelp, acInfoBox, acHeader, acHeaderPanel and acPicturePanel components.

Declaration
```
type
  TacTextStyle = class(TPersistent)
  published
    property PlainText: Boolean default False;
    property TagOpen: Char default '[';
    property TagClose: Char default ']';
  end;
```

Description
The TextStyle structure used to specify the style of text displayed by this component.

The PlainText property specifies whether you would like to use right-formatted text or just flat, plain text. TagOpen and TagClose properties used to specify prefered signs for brackets which identifies begin and end of tags used for rich formatting. By default every tag beginning with '**[**' sign and ending with '**]**' sign.

For example, when PlainText is False, TagOpen is '[' and TagClose is ']', following text:

```
[i]Hello[] [red][b]World[def]!
```

will looks like:

Hello **World**!

Otherwise, if PlainText is True, the tags will not be used and users will see just flat text, including all tags.

© 1998-2006, UtilMind Solutions®

See also
Rich-formatting tags overview.

10.31.2.7.1 PlainText

Applies to
acRichLabel, acInfoBox, acFormHelp, acHeader, acHeaderPanel and acPicturePanel components as subproperty of TextStyle structure.

Declaration
```
property PlainText: Boolean;
```

Description
The PlainText property controls whether the text should be displayed as rich text (if PlainText is False) or as plain, non-formatted text (if PlainText is True). When PlainText is True, the text will be displayed as usual flat text and nor tags will be used.

See also
TagOpen and TagClose properties;
Using rich-formatting tags topic.

10.31.2.7.2 TagClose

Applies to
acRichLabel, acInfoBox, acFormHelp, acHeader, acHeaderPanel and acPicturePanel components as subproperty of TextStyle structure.

Declaration
```
property TagClose: Char;
```

Description
The TagClose property specifies the sign which marks the ending of formatting area of rich-text. The rich formatting tags of AppControls pack are similar to brackets "<>" of tags in HTML format. Default TagClose value is "**]**" sign.

Examples of standard tags
`[b]` - makes text after this tag **bold**.
`[i]` - makes text after this tag *italic*
`[biu]` - makes text after this tag ***bold, italic and underlined***.
`[]` - clears text formatting.
`[red]` - makes text after this tag red.
`[lime]` - makes text after this tag light green (lime).
`[def]` or `[default]` - returns the text formatting to default state as specified in Font property.

Note
If PlainText property is True, nor rich text will be displayed. All formating will be shown as usual flat text.

See also
TagOpen and PlainText properties and example of using rich-formatting tags.

10.31.2.7.3 TagOpen

Applies to
acRichLabel, acInfoBox, acFormHelp, acHeader, acHeaderPanel and acPicturePanel components as subproperty of TextStyle structure.

© 1998-2006, UtilMind Solutions®

Declaration
```
property TagOpen: Char;
```

Description
The TagOpen property specifies the sign which marks the beginning of rich-text formatting area. The rich formatting tags of AppControls pack are similar to brackets "<>" of tags in HTML format. Default TagOpen value is "**[**" sign.

Examples of standard tags
`[b]` - makes text after this tag **bold**.
`[i]` - makes text after this tag *italic*
`[biu]` - makes text after this tag ***bold, italic and underlined***.
`[]` - clears text formatting.
`[red]` - makes text after this tag red.
`[lime]` - makes text after this tag light green (lime).
`[def]` or `[default]` - returns the text formatting to default state as specified in Font property.

See also
TagClose and PlainText properties and example of using rich-formatting tags.

10.31.2.8Transparent

Applies to
acRichLabel property.

Declaration
```
property Transparent: Boolean;
```

Description

See also
ShowFrame and ShowShadow properties.

10.31.2.9WordWrap

Applies to
acRichLabel, acLabel, acURLLabel, acButton, acBitBtn, acCheckBox and acRadioButton components.

Declaration
```
property WordWrap: Boolean;
```

Description
The WordWrap property specifies whether the text string wraps when it is too long for the width of the label. When WordWrap is True, caption can be multiline, and single line when WordWrap is False.

10.31.3 Events

10.31.3.1OnMouseEnter

Applies to
acLabel, acURLLabel, acRichLabel, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acMRUComboBox, acHeader, acHeaderPanel, acTreeView, acListView components.

© 1998-2006, UtilMind Solutions®

Declaration
```
property OnMouseEnter: TNotifyEvent;
```

Description
The OnMouseEnter occurs when mouse pointer hovers the RichLabel.

See also
OnMouseLeave event.

10.31.3.2 OnMouseLeave

Applies to
acLabel, acURLLabel, acRichLabel, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acMRUComboBox, acHeader, acHeaderPanel, acTreeView, acListView components.

Declaration
```
property OnMouseLeave: TNotifyEvent;
```

Description
The OnMouseLeave occurs when mouse pointer leaves the RichLabel.

See also
OnMouseEnter event.

10.32 acSimpleTCPClient

10.32.1 TacSimpleTCPClient

Overview
description coming soon...

See also
acSimpleTCPServer component.

10.32.2 Properties

10.32.2.1 AllowChangeHostAndPortOnConnection

Applies to
acSimpleTCPClient and acSimpleTCPServer components.

Declaration
```
property AllowChangeHostAndPortOnConnection: Boolean;
```

Description

See also
Host, Port and Connected properties.

10.32.2.2 AutoReconnect

Enter topic text here.

10.32.2.3 AutoReconnectInterval

Enter topic text here.

© 1998-2006, UtilMind Solutions®

10.32.2.4Connected

Applies to
acSimpleTCPClient component.

Declaration
```
property Connected: Boolean;
```

Description

See also
Host and Port properties.

10.32.2.5Host

Applies to
acSimpleTCPClient component.

Declaration
```
property Host: String;
```

Description

See also
Port, Connected and AllowChangeHostAndPortOnConnection properties;
OnConnected and OnDisconnected events.

10.32.2.6Port

Applies to
acSimpleTCPClient and acSimpleTCPServer components.

Declaration
```
property Port: Word;
```

Description

See also
Host, Connected and AllowChangeHostAndPortOnConnection properties;
OnConnected and OnDisconnected events.

10.32.3 Methods

10.32.3.1Send

Applies to
acSimpleTCPClient and acSimpleTCPServer components.

Declaration
```
function Send(Buffer: PChar; BufLength: Integer): Integer;
```

Description

See also
SendStream method; OnRead event.

© 1998-2006, UtilMind Solutions®

10.32.3.2SendStream

Applies to
acSimpleTCPClient and acSimpleTCPServer components.

Declaration
```
function SendStream(Stream: TStream): Integer;
```

Description

See also
Send method; OnRead event.

10.32.3.3Receive

Enter topic text here.

10.32.3.4ReceiveStream

Enter topic text here.

10.32.4 Events

10.32.4.1OnConnected

Applies to
acSimpleTCPClient and acSimpleTCPServer components.

Declaration
```
property OnConnected: TNotifyEvent;
```

Description

See also
OnDisconnected event.

10.32.4.2OnDataAvailable

Applies to
acSimpleTCPClient component.

Declaration
```
type
  TacSimpleTCPClientDataAvailEvent = procedure(Sender: TObject;
DataSize: Integer) of object;

property OnDataAvailable:  TacSimpleTCPClientDataAvailEvent;
```

Description

See also

10.32.4.3OnDisconnected

Applies to
acSimpleTCPClient and acSimpleTCPServer components.

© 1998-2006, UtilMind Solutions®

Declaration

```
property OnDisconnected: TNotifyEvent;
```

Description

See also

OnConnected event.

10.32.4.4 OnError

Applies to

acSimpleTCPClient and acSimpleTCPServer components.

Declaration

```
type
  TacSimpleTCPErrorEvent = procedure(Sender: TObject; Socket: TSocket;
ErrorCode: Integer; ErrorMsg: String) of object;

property OnError: TacSimpleTCPErrorEvent;
```

Description

10.32.4.5 OnRead

Applies to

acSimpleTCPClient component.

Declaration

```
type
  TacSimpleTCPClientIOEvent = procedure(Sender: TObject; Stream:
TStream) of object;

property OnRead: TacSimpleTCPClientIOEvent;
```

Description

10.32.4.6 OnSelfDisconnected

Applies to

acSimpleTCPClient component.

Declaration

```
property OnSelfDisconnected: TNotifyEvent;
```

Description

See also

10.33 acSimpleTCPServer

10.33.1 TacSimpleTCPServer

Overview
description coming soon...

© 1998-2006, UtilMind Solutions®

See also
acSimpleTCPClient component.

10.33.2 Properties

10.33.2.1AllowChangeHostAndPortOnConnection

Applies to
acSimpleTCPServer and acSimpleTCPClient components.

Declaration
```
property AllowChangeHostAndPortOnConnection: Boolean;
```

Description

See also
Port and Listen properties.

10.33.2.2Connections

Applies to
acSimpleTCPServer component.

Declaration
```
property Connections: TList; // list of TacSimpleTCPClient objects
```

Description

See also

10.33.2.3Listen

Applies to
acSimpleTCPServer component.

Declaration
```
property Listen: Boolean;
```

Description

See also
Port and AllowChangeHostAndPortOnConenction properties;
OnClientConnected and OnClientDisconnected events.

10.33.2.4LocalHostName

Applies to
acSimpleTCPServer component.

Declaration
```
property LocalHostName: String;
```

Description

See also
LocalIP property.

© 1998-2006, UtilMind Solutions®

10.33.2.5LocalIP

Applies to
acSimpleTCPServer component.

Declaration
```
property LocalIP: String;
```

Description

See also
LocalHostName property.

10.33.2.6Port

Applies to
acSimpleTCPServer and acSimpleTCPClient components.

Declaration
```
property Port: Word;
```

Description

See also
Listen and AllowChangeHostAndPortOnConnection properties;
OnClientConnected and OnClientDisconnected events.

10.33.3 Methods

10.33.3.1Send

Applies to
acSimpleTCPServer and acSimpleTCPClient components.

Declaration
```
function Send(Client: TacSimpleTCPClient; Buffer: PChar; BufLength:
Integer): Integer;
```

Description

See also
SendStream, Broadcast and BroadcastStream methods.

10.33.3.2SendStream

Applies to
acSimpleTCPServer and acSimpleTCPClient components.

Declaration
```
function SendStream(Client: TacSimpleTCPClient; Stream: TStream):
Integer;
```

Description

See also
Send, Broadcast and BroadcastStream methods.

© 1998-2006, UtilMind Solutions®

10.33.3.3Broadcast

Applies to
acSimpleTCPServer component.

Declaration
```
function Broadcast(Buffer: PChar; BufLength: Integer): Integer;
```

Description
The Broadcast

This is the same as
```
var
  I: Integer;
  TheString: String;
  tmpClient: TacSimpleTCPClient;
begin
  TheString := 'test';
  I := acSimpleTCPServer1.Connections.Count;
  if I <> 0 then
   for I := 0 to I - 1 do
    begin
     tmpClient := Server.Connections[I];
     Server.Send(tmpClient, PChar(TheString), Length(TheString));
    end;
end;
```

See also
Send, SendStream and BroadcastStream methods.

10.33.3.4BroadcastStream

Applies to
acSimpleTCPServer component.

Declaration
```
function BroadcastStream(Stream: TStream): Integer;
```

Description

See also
Send, SendStream and Broadcast methods.

10.33.3.5Receive

Enter topic text here.

10.33.3.6ReceiveStream

Enter topic text here.

10.33.4 Events

10.33.4.1OnAccept

Applies to
acSimpleTCPServer component.

© 1998-2006, UtilMind Solutions®

Declaraton

```
type
  TacSimpleTCPAcceptEvent = procedure(Sender: TObject; Client:
TacSimpleTCPClient; var Accept: Boolean) of object;

property OnAccept: TacSimpleTCPAcceptEvent;
```

Description

See also

OnAccept, OnClientConnected and OnClientDisconnected events.

10.33.4.2OnClientConnected

Applies to

acSimpleTCPServer component.

Declaraton

```
type
  TacSimpleTCPServerEvent = procedure(Sender: TObject; Client:
TacSimpleTCPClient) of object;

property OnClientConnected: TacSimpleTCPServerEvent;
```

Description

See also

OnAccept and OnClientDisconnected events.

10.33.4.3OnClientDataAvailable

Applies to

acSimpleTCPServer component.

Declaraton

```
type
  TacSimpleTCPServerDataAvailEvent = procedure(Sender: TObject; Client:
TacSimpleTCPClient; DataSize: Integer) of object;

property OnDataAvailable: TacSimpleTCPClientDataAvailEvent
```

Description

See also

OnAccept and OnClientDisconnected events.

10.33.4.4OnClientDisconnected

Applies to

acSimpleTCPServer component.

Declaraton

```
type
  TacSimpleTCPServerEvent = procedure(Sender: TObject; Client:
TacSimpleTCPClient) of object;

property OnClientDisconnected: TacSimpleTCPServerEvent;
```

© 1998-2006, UtilMind Solutions®

Description

See also
OnAccept and OnClientConnected events.

10.33.4.5OnClientRead

Applies to
acSimpleTCPServer component.

Declaraton
```
type
  TacSimpleTCPServerIOEvent = procedure(Sender: TObject; Client:
TacSimpleTCPClient; Stream: TStream) of object;

property OnClientRead: TacSimpleTCPServerIOEvent;
```

Description

See also
OnError event.

10.33.4.6OnError

Applies to
acSimpleTCPServer and acSimpleTCPClient components.

Declaration
```
type
  TacSimpleTCPErrorEvent = procedure(Sender: TObject; Socket: TSocket;
ErrorCode: Integer; ErrorMsg: String) of object;

property OnError: TacSimpleTCPErrorEvent;
```

Description

10.34 acSplitterPanel

10.34.1 TacSplitterPanel

Overview
The acSplitterPanel control is specialized panel designed to split itself into two areas, or panes, either horizontally or vertically. Each pane can be easily customized with number of properties (i.e: you can specify minimal size for pane, specify the splitter behavior and appearence, make the panes to be resized proportionally when user resize the window etc). The acSplitterPanels can be nested to create multi-paned splitter regions. Also, it can automatically save and restore the splitter position from the registry.

How to use?

See also

© 1998-2006, UtilMind Solutions®

10.34.2 Properties

10.34.2.1Align

Applies to

10.34.2.2Background

Unit
acGraphics

Applies to
acSplitterPanel, acHeader, acHeaderPanel and acPicturePanel components.

Declaration

```
type
  TacBackground = class(TPersistent)
  published
    property Active: Boolean default False;
    property GradientColor1: TColor default clBlue;
    property GradientColor2: TColor default clBlack;
    property GradientDirection: TacGradientDirection default
gdHorizontal;
    property Kind: TacBackgroundKind default bkGradient;
    property Picture: TPicture;
    property PictureStyle: TacBackgroundPictureStyle default bsTile;
  end;
```

Description
The Background structure used to specify custom background and its behaviors for the panel.

If you're tired of single-tone color panels and want to show the picture as "wallpaper", or smooth gradient effect under the controls of panel — simply activate it by specifying True to Active property and choose the Kind of background (specify which background you want: picture or gradient effect).

If you want to show the picture (wallpaper) — set Kind property to `bkPicture`, specify the image in Picture property and choose the style of wallpaper using PictureStyle property.

If you want to show the gradient effect instead of usual background — set Kind property to `bkGradient`, choose the style of effect in GradientDirection property (it can be vertical, horizontal or central), then specify the colors in GradientColor1 and GradientColor2 properties.

Alternatively, if you want to show your very own background / effect, you can use OnPaint event. Just write the OnPaint event handler and draw on the canvas of panel before all controls located on this panel will be displayed.

See also
OnPaint event.
acFormBackground component.

10.34.2.2.1 Active

Applies to
acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components as sub-property of Background structure.

© 1998-2006, UtilMind Solutions®

Declaration

```
property Active: Boolean;
```

Description

The Active property of Background structure specifies whether the control should have custom background (gradient or picture).

Set Active to True to show the custom background (with smooth gradient effect or wallpaper specified in Picture property). Otherwise, set it to False to show standard single-tone background with color specified in Color property of control.

To specify the kind of background (gradient or picture) — use Kind property.

Alternatively, for better customization, you can use OnPaint event.

See also

Kind property;
OnPaint event.

10.34.2.2.2 GradientColor1

Applies to

acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components as sub-property of Background structure.

Declaration

```
property GradientColor1: TColor;
```

Description

Specified the first (left if GradientDirection is horizontal or top if GradientDirection is vertical) gradient color of your form.

See also

GradientColor2, GradientDirection and Kind properties.

10.34.2.2.3 GradientColor2

Applies to

acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components as sub-property of Background structure.

Declaration

```
property GradientColor2: TColor;
```

Description

Specified the second (right if GradientDirection is horizontal or bottom if GradientDirection is vertical) gradient color of your form.

See also

GradientColor1, GradientDirection and Kind properties.

10.34.2.2.4 GradientDirection

Applies to

acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components as sub-property of Background structure.

© 1998-2006, UtilMind Solutions®

Declaration

```
type
  TacGradientDirection = (gdVertical, gdHorizontal, gdCentral);

property GradientDirection: TacGradientDirection;
```

Description

Specifies the direction of the gradient effect. Direction can be vertical, horizontal or central. To specify the colors for gradient effect — use GradientColor1 and GradientColor2 properties.

Snapshots

`GradientDirection = gdVertical`

`GradientDirection = gdHorizontal`

`GradientDirection = gdCentral`

See also

GradientColor1, GradientColor2 and Kind properties.

10.34.2.2.5 Kind

Applies to

acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components as sub-property of Background structure.

Declaration

```
type
  TacFormBackgroundKind = (bkPicture, bkGradient);

property Kind: TacFormBackgroundKind;
```

Description

The Kind property specifies how to paint the form background: using the picture (bkPicture), or drawing the gradient effect (bkGradient).

If you would like to paint background using the **picture**:

1. Set Kind property to `bkPicture`;
2. Specify the bitmap image in Picture property;
3. Select the style of drawing in the PictureStyle property.

If you wish to show the **gradient effect**:

1. Set Kind property to `bkGradient`;
2. Specify the direction of the effect in GradientDirection property;

© 1998-2006, UtilMind Solutions®

3. Choose two colors to draw the effect in GradientColor1 and GradientColor2 properties.

See also
Active, GradientDirection and PictureStyle properties;
OnPaint event.

10.34.2.2.6 Picture

Applies to
acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components as sub-property of Background structure.

Declaration
```
property Picture: TPicture;
```

Description
The Picture property specifies the image that appears as form background. The Picture can contains an icon, betafile, bitmap graphic, or user-defined graphic to be displayed as the background (if Kind = bkPicture).

Setting Picture at design time brings up the Picture Editor, which can be used to specify the file that contains the image.

To specify the style of drawing — use PictureStyle property.

See also
PictureStyle and Kind properties.

10.34.2.2.7 PictureStyle

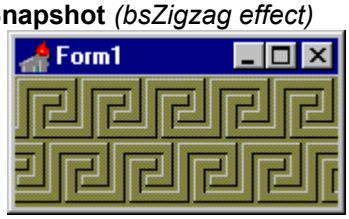
Applies to
acHeader, acHeaderPanel, acPicturePanel and acSplitterPanel components as sub-property of Background structure.

Declaration
```
property PictureStyle: TacBackgroundPictureStyle;
```

Description
The PictureStyle property specifies the effect used to draw the Picture. There are possible values:

Value	Meaning
bsTile	tiles the bitmap image on form's background;
bsStretch	stretches the image so that it exactly fits the bounds of the form. When the form resized, the image will be resized as well;
bsZigzag	almost the same as tile effect, but every next row of tile is shifted relatively to previous row, like bricks in the wall;
bsCenter	draws the image in the center of form.

Snapshot *(bsZigzag effect)*



© 1998-2006, UtilMind Solutions®

See also
Picture and Kind properties.

10.34.2.3ClipSplitRect

Applies to
acSplitterPanel component.

Declaration
```
property ClipSplitRect: Boolean;
```

Description

See also

10.34.2.4CursorHorz

Applies to
acSplitterPanel component.

Declaration
```
property CursorHorz: TCursor;
```

Description
The CursorHorz properties specifies the image for the mouse pointer, when mouse hovers the horizontal divider line between two panes.

See also
CursorVert and SplitterStyle properties.

10.34.2.5CursorVert

Applies to
acSplitterPanel component.

Declaration
```
property CursorVert: TCursor;
```

Description
The CursorVert properties specifies the image for the mouse pointer, when mouse hovers the vertical divider line between two panes.

See also
CursorHorz and SplitterStyle properties.

10.34.2.6MinSize

Applies to
acSplitterPanel component.

Declaration
```
property MinSize: Word;
```

Description

See also

© 1998-2006, UtilMind Solutions®

ProportionalPanes, SplitterPos, SplitterWidth and SplitterStyle properties.

10.34.2.7 ProportionalPanes

Applies to
acSplitterPanel component.

Declaration
```
property ProportionalPanes: Boolean;
```

Description

See also

10.34.2.8 RegistrySaver

Enter topic text here.

10.34.2.8.1 Enabled

Enter topic text here.

10.34.2.8.2 IniFileName

Enter topic text here.

10.34.2.8.3 IniSection

Enter topic text here.

10.34.2.8.4 Options

Enter topic text here.

10.34.2.8.5 RegKey

Enter topic text here.

10.34.2.8.6 RegLocation

Enter topic text here.

10.34.2.8.7 Storage

Enter topic text here.

10.34.2.9 SplitterStyle

Applies to
acSplitterPanel component.

Declaration
```
type
  TacSplitterStyle = (ssUpdate, ssLine, ssPattern);

property SplitterStyle: TacSplitterStyle;
```

Description
The SplitterStyle specifies the look of the dividing line between to panes of SplitterPanel.

© 1998-2006, UtilMind Solutions®

See also

SplitterPos
10.34.2.10

Applies to
acSplitterPanel component.

Declaration
```
property SplitterPos: Integer;
```

Description

See also
MinSize, SplitterWidth and SplitterStyle properties;
OnSplitterPosChanged event.

SplitterWidth
10.34.2.11

Applies to
acSplitterPanel component.

Declaration
```
property SplitterWidth: Byte;
```

Description
The SplitterWidth property specifies the width for splitter bar between two splitted panes. When user drags the splitter, he will see

See also
SplitterPos and SplitterStyle properties.

10.34.3 Events

10.34.3.1 OnPaint

Applies to
acHeader, acHeaderPanel, acSplitterPanel and acPicturePanel components.

Declaration
```
property OnPaint: TNotifyEvent;
```

Description
The OnPaint event occurs when Windows requires the panel to paint, such as when the form or panel receives focus or becomes visible when it wasn't previously.

Your application can use this event to draw on the canvas of the panel. For example, to draw custom background.

See also
Background structure.

© 1998-2006, UtilMind Solutions®

10.34.3.2OnSplitterPosChanged

Aplies to
acSplitterPanel component.

Declaration
```
type
  TacSplitterPanelSplitterPosChanged = procedure(Sender: TObject;
SplitterPos: Integer) of object;

property OnSplitterPosChanged: TacSplitterPanelSplitterPosChanged;
```

Description
The OnSplitterPosChanged event occurs when the value of SplitterPos property changes (in example, when user moves the dividing line between two panes of splitter panel with mouse).

The *SplitterPos* parameter determines the current position of divider between panes.

See also
SplitterPos, SplitterWidth and SplitterStyle properties.

10.35 acSQLQueryThread

10.35.1 TacSQLQueryThread

Overview
The TacSQLQueryThread used to *Open* the SQL queries in separate process, avoiding suspendings of the main application thread.

How to use?
Just specify TDataSet object any its descendent (i.e: standard TQuery component, or TmySQLQuery by MicroOLAP) in Query property, and call RunSQL method.

After calling RunSQL method, the component opens a dataset in serparate thread (calls *Open* method of Query), and allows to retrieve results in OnDone method, if succeed. It opens an SQL query without suspending other parts of your application (i.e: user interface). Also you can specify Priority of this thread in a schedule of system processes.

To get results of SQL query — use OnDone event. If you wish to terminate thread imediately — use Abort method.

See also
acThread component.

10.35.2 Properties

10.35.2.1Priority

Applies to
acSQLQueryThread and acThread components.

Declaration
```
property Priority: TThreadPriority;
```

Description
Priority indicates the priority used when scheduling the thread. Adjust the priority higher or lower as

© 1998-2006, UtilMind Solutions®

needed.

TThreadPriority type defines the possible values for the Priority property of the acThread component, as defined in the following table. The system schedules CPU cycles to each thread based on a priority scale; the Priority property adjusts a thread's priority higher or lower on the scale.

Values	Meaning
tpIdle	The thread executes only when the system is idle. The system will not interrupt other threads to execute a thread with tpIdle priority.
tpLowest	The thread's priority is two points below normal.
tpLower	The thread's priority is one point below normal.
tpNormal	The thread has normal priority.
tpHigher	The thread's priority is one point above normal.
tpHighest	The thread's priority is two points above normal.
tpTimeCritical	The thread gets highest priority.

Warning
Boosting the thread priority of a CPU intensive operation may "starve" the other threads in the application. Only apply priority boosts to threads that spend most of their time waiting for external events.

10.35.2.2Query

Applies to
acSQLQueryThread component.

Declaration
```
property Query: TDataSet;
```

Description
The Query property specifies the DataSet which you would like to *Open* in separate thread. You can specify ANY descendant of TDataSet class, for example standard TQuery component, or TmySQLQuery by MicroOLAP.

After calling RunSQL method, the component opens a dataset in serparate thread (calls *Open* method of Query), and returns results in OnDone method, if succeed.

See also
Query, Priority and WaitThread properties;
OnDone and OnException events;
Abort method.

10.35.2.3WaitThread

Applies to
acThread, acSQLQueryThread, acHTTP and acSendMail components.

Declaration
```
property WaitThread: Boolean;
```

Description
The WaitThread property controls whether the procedure that calls the RunSQL method should be suspended and wait until the scanning process will be done.

Set the WaitThread to True, if you would like to process SQL query in separate thread so that the

© 1998-2006, UtilMind Solutions®

application does not continue with next lines of code after calling the RunSQL method. Your application will done execution of the thread *before* continuing to next step.

If your application can wait, until the the SQL query will be completed, only for some limited time interval — specify WaitTimeout property.

See also
RunSQL and Abort methods.

10.35.2.4WaitTimeout

Applies to
acSQLQueryThread, acHTTP, acSendMail and acThread components.

Declaration
```
property WaitTimeout: Integer;
```

Description
The WaitTimeout property specifies the time interval (limit), in milliseconds unit, which application able to wait until the SQL query will be completed.

For example, if the maximum time which you can allow to complete SQL query is 5 seconds, set this value to 5000 (milliseconds). If application can wait infinitely, set WaitTimeout to 0.

When the timeout is expired, the component automatically terminates the SQL query. To be notified when the WaitTimeout is expired — write OnWaitTimeoutExpired event handler.

Notes
The WaitTimeout only works together with WaitThread property, only when it set to True.

See also
WaitThread and Priority properties;
RunSQL and Abort methods;
OnWaitTimeoutExpired event.

10.35.3 Methods

10.35.3.1RunSQL

Applies to
acSQLQueryThread component.

Declaration
```
function RunSQL: Boolean; // returns False if busy or WaitTimeout expired
```

Description
The RunSQL method *opens* the dataset (or any its descendant) specified in Query property. It just calls **Open** method of specified Query in the separate thread, without suspending other processes of your application.

To retrieve results after opening the dataset — use OnDone event.

RunSQL method returns True if successful or False if thread is busy (already processing another query) OR WaitTimeout is expired.

© 1998-2006, UtilMind Solutions®

See also
Query, Priority, WaitThread and WaitTimeout properties;
OnDone and OnException events;
Abort method.

10.35.3.2Abort

Applies to
acSQLQueryThread, acHTTP, acAutoUpgrader, acProportionalImage and acSendMail components.

Declaration
```
procedure Abort;
```

Description
The Abort method imediately terminates execution of thread which *opens* the SQL Query.

See also
RunSQL method.

10.35.4 Events

10.35.4.1OnDone

Applies to
acSQLQueryThread, acHTTP and acProportionalImage components.

Declaration
```
type
  TacSQLQueryDoneEvent = procedure(Sender: TObject; DataSet: TDataSet)
of object;

property OnDone: TacSQLQueryDoneEvent;
```

Description
The OnDone method occurs when the dataset (specified in Query property) has been successfully *opened* and can be used to retrieve results of SQL query.

Example
```
procedure TForm1.LogQueryDone(Sender: TObject;
  DataSet: TDataSet);
var
  Field1: Integer;
  Field2: String;
begin
  with DataSet do
   try
     Field1 := Fields[0].AsInteger;
     Field2 := Fields[1].AsString;
   except
   end;
end;
```

See also
RunSQL and Abort methods;
OnException event.

© 1998-2006, UtilMind Solutions®

10.35.4.2OnException

Applies to
acThread, acSQLQueryThread and acAppEvents components.

Declaration
```
property OnException: TNotifyEvent;
```

Description
The OnException event occurs if something has been failed on execution of thread, when the component *opens* the Query.

See also
OnDone event.

10.35.4.3OnWaitTimeoutExpired

Applies to
acSQLQueryThread, acHTTP, acSendMail and acThread components.

Declaration
```
property OnWaitTimeoutExpired: TNotifyEvent;
```

Description
The OnWaitTimeoutExpired event occurs when the component could not complete the SQL query for the time interval specified in WaitTimeout property (when WaitTimeout is expired).

The *TherminateThread* is optional parameter (True by default), which allows to terminate, or prevent termination of thread in case if specified time interval is expired before completion of execution of the thread.

Notes
The WaitTimeout only works together with WaitThread property, only when it set to True.

See also
WaitThread and WaitTimeout properties;
Abort method.

10.36 acStatusControl

10.36.1 TacStatusControl

Overview
The acStatusControl component used to insert any visible control (button, panel, checkbox etc) to the section of the status bar.

Snapshot

Properties

Control	specifies the control to insert;
PanelIndex	specifies the section of the status bar
StatusBar	specifies the status bar to insert the control.

© 1998-2006, UtilMind Solutions®

10.36.2 Properties

10.36.2.1Control

Applies to
acStatusControl and acCursorLocker components.

Declaration
```
property Control: TControl;
```

Description
The Control property specifies the visible control to insert it to the panel of TStatusBar control. You can point the Control property to any visible component (i.e: panel, button, check box, and so on).

See also
StatusBar and PanelIndex property.

10.36.2.2PanelIndex

Applies to
acStatusControl component.

Declaration
```
property PanelIndex: Integer;
```

Description
The PanelIndex property specifies the section in the StatusBar to insert the Control.

Note
When you place some control to panel of the status bar, this will change the panel's draw style to *psOwnerDraw* (see Style property of TStatusPanel object).

When you change the PanelIndex property to move the control to another panel, make sure that panel where you'd like to display text has *psText* style.

See also
Control and StatusBar properties.

10.36.2.3StatusBar

Applies to
acStatusControl component.

Declaration
```
property StatusBar: TStatusBar;
```

Description
The StatusBar property is the pointer to current status bar on the form. The Control will be inserted to this status bar, in section specified by PanelIndex property.

See also
Control and PanelIndex properties.

© 1998-2006, UtilMind Solutions®

10.37 acStringListContainer

10.37.1 acStringListContainer

Overview

The acStringListContainer is the invisible component which can be used to specify the string lists at design-time and hold the sets of strings right in the form.

Contains some additional features which allows to work with the lists of "Names=Values" (strings separated on two parts, where first part is some "name" and second is "value", separated with some character). Can disallow adding of the duplicate string independently of whether the list is sorted or not (unlike usual TStringList). Contains the methods to put and retrieve the strings in FIFO manner (allows to specify the list capacity (maximum number of strings), so if the list is full and the new string is about to be added, the first will be deleted previously, plus has the method which automatically delete the first string upon retreiving). Also it can store and retrieve the strings from registry.

10.37.2 Properties

10.37.2.1Count

Applies to

acStringListContainer component.

Declaration

```
property Count: Integer; // read-only!
```

Description

The Count is the read-only property which determines the number of strings which currently are in the list.

See also

MaxSize property.

10.37.2.2Duplicates

Applies to

acStringListContainer component.

Declaration

```
type
  TDuplicates = (dupIgnore, dupAccept, dupError);

property Duplicates: TDuplicates;
```

Description

The Duplicates property determines whether duplicate strings are allowed in the list of strings.

Unlike of the usual TStringList, where the Duplicates property only takes the effect in sorted lists only, here the Duplicates works independently, whether the list is sorted or not. It works always and don't let to add the same string if `Duplicates = dupIgnore`.

These are the possible values for Duplicates property:

Value	Meaning
dupIgnore	Attempts to add a duplicate string to a list are ignored

© 1998-2006, UtilMind Solutions®

dupAccept	Duplicate strings CAN be added to a string list
dupError	Adding a duplicate string results in an EListError exception

See also
Sorted property.

10.37.2.3List

Applies to
acStringListContainer component.

Declaration
```
property List: TacStringList; // read-only!
```

Description
The List property gives an access to TacStringList object (successor of TStringList), which holds the strings for this container component.

The StringListContainer is just the component which contains the TacStringList object. By using the List property you can get access to pure TacStringList.

10.37.2.4MaxSize

Applies to
acStringListContainer component.

Declaration
```
property MaxSize: Integer;
```

Description
The MaxSize specifies the capacity of string list (the maximum number of strings which it can hold).

Warning!
If the list is full (already contains the maximum number of strings), and you are trying to add a new string into list, the FIRST string will be deleted! This feature was added to let you to operate with "FIFO" (first-in-first-out) lists, so if you adding more lines than it can hold, the first line will be automatically thrown out. You can use GetFIFO method to retreive the first line and automatically delete it.

For example, if MaxSize = 100 and you're trying to add 101st string, the first string will be deleted from the list to allow the new string to be added. Anyway, by default the MaxSize set to MaxInt, so it can hold up to 2,147,483,647 lines of text (but of course you can decrease this value if your program requires so).

See also
Count property;
GetFIFO, GetAndDeleteItem, Add, Insert, StoreToRegistry and LoadFromRegistry methods.

10.37.2.5NVSeparator

Applies to
acStringListContainer component.

Declaration
```
property NVSeparator: String; // name/value separator, '=' by default
```

© 1998-2006, UtilMind Solutions®

Description

The NVSeparator is the string or character which separates each string on two parts, where first part is some "name" and second is some "value". It only used for the list where each string presented in the **name=value** manner.

This feature allows to quickly find required Names and Values by their indexes, read or modify the names/values without changing whole structure of string list.

See also

Names, Values, Strings and NVSeparatorMainSide properties.

10.37.2.6NVSeparatorMainSide

Applies to

acStringListContainer component.

Declaration

```
type
  TacStringListNVSeparatorSide = (nvsLeft, nvsRight);

property NVSeparatorMainSide: TacStringListNVSeparatorSide;
```

Description

The NVSeparatorMainSide property used to specify the priority side of string separated by NVSeparator character.

For example, if you know that Names can contain the characters which also used by separator, you can set NVSeparatorMainSide to nvsRight, and the TacStringList object will separate the strings to Names and Values using LAST separator character found in the string.

For another example, the NVSeparator is the space character (" "), and our list contain a lines like `"hello world 12345"`. In this case, if NVSeparatorMainSide is `nvsLeft`, the Names property will return `"hello"` and Values will return `"world 12345"`. However, if you set NVSeparatorMainSide to `nvsRight`, the Names property will return `"hello world"` and Values — `"12345"`.

See also

Names, Values, Strings and NVSeparator properties.

10.37.2.7Sorted

Applies to

acStringListContainer component.

Declaration

```
property Sorted: Boolean; // False by default
```

Description

The Sorted property specifies whether the strings in the list should be automatically sorted.

Set Sorted to True to cause the strings in the list to be automatically sorted in ascending order. Set Sorted to False to allow strings to remain where they are inserted. When Sorted is False, the strings in the list can be put in ascending order at any time by calling the Sort method.

When Sorted is True, do not use Insert to add strings to the list. Instead, use Add, which will insert the new strings in the appropriate position. When Sorted is False, use Insert to add strings to an

© 1998-2006, UtilMind Solutions®

arbitrary position in the list, or Add to add strings to the end of the list.

See also
Duplicates, Strings, Names and Values properties;
Add, AddStrings and Sort methods.

10.37.2.8Names

Applies to
acStringListContainer component.

Declaration
```
property Names[Index: Integer]: String;
```

Description
The Names is the *run-time only* property, which can be used to read or modify the first part of string, separated on 2 parts with some character of string specified in NVSeparator property. The Names property should be used only if each string of the list is organized in the manner of "**name=value**" and allows to retreive and modify the "names" (as well as Values property gives an access to "values").

With the Names you can access a *first part* of a specific string ("name" part, *before* separator), to read or modify the "name" at a particular position. *Index* gives the position of the string, where 0 is the position of the first string, 1 is the position of the second string, and so on ("Count - 1" is the position of last string in the list).

To locate a particular string in the list, call IndexOf method, or use **IndexOfName** and IndexOfValue methods if the string list is the list of "names=values" (where each string is some "name" and "value", separated with some character).

For example, if the fist string in the list is "hello=world", and the separator sign is "=", then reading of the Names[0] at run-time will returns string "hello".

Example *(demonstrates how to retrieve the first part of string before the separator specified in the NVSeparator property)*
```
procedure TForm1.Button1Click(Sender: TObject);
begin
  // retreives the "name", by index entered in some Edit box, and
displays it in some Label
  Label1.Caption := acStringListContainer1.Names[StrToIntDef(Edit1.Text,
0)];

  // modifies the "name" part of the first string
  acStringListContainer1.Names[0] := 'Hello';
  // modifies the "value" part of the first string
  acStringListContainer1.Values[0] := 'World!';

  // the example above is the same as example below, if NVSeparator
property is '='
  acStringListContainer1.Strings[0] := 'Hello=World!';
end;
```

See also
Values, Strings and NVSeparator properties;
LoadFromRegistry and StoreToRegistry methods.

© 1998-2006, UtilMind Solutions®

10.37.2.9 Strings

Applies to
acStringListContainer component.

Declaration
```
property Strings[Index: Integer]: String; default;
```

Description
The Strings is the *run-time only* property. With the Strings you can access a specific string, to read or modify the string at a particular position. *Index* gives the position of the string, where 0 is the position of the first string, 1 is the position of the second string, and so on ("Count - 1" is the position of last string in the list).

To locate a particular string in the list, call **IndexOf** method, or use IndexOfName and IndexOfValue methods if the string list is the list of "names=values" (where each string is some "name" and "value", separated with some character).

Strings is the default property of string list objects. The Strings identifier can be omitted when accessing the Strings property of a string list object. For example, the following two lines of code are both acceptable and do the same thing:
```
acStringListContainer1.Strings[0] := 'This is the first string';
acStringListContainer1[0] := 'This is the first string';
```

To determinate the number of strings which currently are available in the list — read Count property.

See also
Names and Values properties.

Values 10.37.2.10

Applies to
acStringListContainer component.

Declaration
```
property Values[Index: Integer]: String;
```

Description
The Values is the *run-time only* property, which can be used to read or modify the second part of string, separated on 2 parts with some character of string specified in NVSeparator property. The Values property should be used only if each string of the list is organized in the manner of `"name=value"` and allows to retreive and modify the "values" (as well as Names property gives an access to "names").

With the Values you can access a *second part* of a specific string ("value" part, *after* separator), to read or modify the "value" at a particular position. *Index* gives the position of the string, where 0 is the position of the first string, 1 is the position of the second string, and so on ("Count - 1" is the position of last string in the list).

To locate a particular string in the list, call IndexOf method, or use IndexOfName and **IndexOfValue** methods if the string list is the list of "names=values" (where each string is some "name" and "value", separated with some character).

For example, if the fist string in the list is `"hello=world"`, and the separator sign is `"="`, then reading of the `Values[0]` at run-time will returns string `"world"`.

© 1998-2006, UtilMind Solutions®

Example *(demonstrates how to retrieve the first part of string before the separator specified in the NVSeparator property)*

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  // retreives the "value", by index entered in some Edit box, and
displays it in some Label
  Label1.Caption :=
acStringListContainer1.Values[StrToIntDef(Edit1.Text, 0)];

  // modifies the "name" part of the first string
  acStringListContainer1.Names[0] := 'Hello';
  // modifies the "value" part of the first string
  acStringListContainer1.Values[0] := 'World!';

  // the example above is the same as example below, if NVSeparator
property is '='
  acStringListContainer1.Strings[0] := 'Hello=World!';
end;
```

See also

Names, Strings and NVSeparator properties;
LoadFromRegistry and StoreToRegistry methods.

10.37.3 Methods

10.37.3.1Add

Applies to

acStringListContainer component.

Declaration

```
function Add(const S: String): Integer;
```

Description

The Add method adds a new string to the list.

Call Add to add the string *S* to the end of the list. If the list is not sorted, S is added to the end of the list. Add returns the position of the item in the list, where the first item in the list has a value of 0.

The Add adds the string to the end of the list. If you would like to insert the string to some particular position of the list — use Insert method instead.

Notes

Add will raise an EListError exception if the string *S* already appears in the list and Duplicates is set to dupError. If Duplicates is set to dupIgnore, trying to add a duplicate string does nothing.

Note that the list can't contain more strings than the number specified in the MaxSize property. For example, if MaxSize = 100 and you're trying to add 101st string, the first string will be deleted from the list to allow the new string to be added.

Example

```
procedure TForm1.FormCreate(Sender: TObject);
var
  MyList: TacStringList;
  Index: Integer;
begin
```

© 1998-2006, UtilMind Solutions®

```
  MyList := TacStringList.Create;
  try
    MyList.Add('Animals');
    MyList.Add('Flowers');
    MyList.Add('Cars');

    if MyList.Find('Flowers', Index) then
     begin
      ListBox1.Items.AddStrings(MyList);
      Label1.Caption := 'Flowers has an index value of ' +
IntToStr(Index);
     end;
  finally
    MyList.Free;
  end;
end;
```

See also

Insert and AddStrings methods;
Duplicates and MaxSize properties;
OnChange event.

10.37.3.2AddStrings

Applies to

acStringListContainer component.

Declaration

```
procedure AddStrings(Strings: TStrings);
```

Description

The AddStrings method adds a group of strings to the list.

Call AddStrings to add the strings from another TStrings object (or its successor) to the list. If both the source and destination TStrings objects support objects associated with their strings, references to the associated objects will be added as well.

Example

```
procedure TForm1.FormCreate(Sender: TObject);
var
  MyList: TacStringList;
  Index: Integer;
begin
  MyList := TacStringList.Create;
  try
    MyList.Add('Animals');
    MyList.Add('Flowers');
    MyList.Add('Cars');

    if MyList.Find('Flowers', Index) then
     begin
      ListBox1.Items.AddStrings(MyList);
      Label1.Caption := 'Flowers has an index value of ' +
IntToStr(Index);
     end;
  finally
    MyList.Free;
```

© 1998-2006, UtilMind Solutions®

```
  end;
end;
```

See also
Add and Insert methods;
Duplicates property.

10.37.3.3Clear

Applies to
acStringListContainer component.

Declaration
```
procedure Clear;
```

Description
The Clear method deletes all the strings from the list.

See also
Delete and DeleteFirstItem methods.

10.37.3.4Delete

Applies to
acStringListContainer component.

Declaration
```
procedure Delete(Index: Integer);
```

Description
The Delete method removes the string specified by the *Index* parameter.

Call Delete to remove a single string from the list. If an object is associated with the string, the reference to the object is removed as well. *Index* gives the position of the string, where 0 is the first string, 1 is the second string, and so on ("Count - 1" is the position of last string in the list).

See also
Clear and DeleteFirstItem methods.

10.37.3.5DeleteFirstItem

Applies to
acStringListContainer component.

Declaration
```
procedure DeleteFirstItem;
```

Description
The DeleteFirstItem method removes the first string from the list.

See also
GetAndDeleteItem, GetFIFO, Delete and Clear methods.

10.37.3.6Exchange

Applies to
acStringListContainer component.

© 1998-2006, UtilMind Solutions®

Declaration

```
procedure Exchange(Index1, Index2: Integer);
```

Description

The Exchange method swaps the position of two strings in the list.

Call Exchange to rearrange the strings in the list. The strings are specified by their index values in the Index1 and Index2 parameters. Indexes are zero-based, so the first string in the list has an index value of 0, the second has an index value of 1, and so on.

If either string has an associated object, Exchange changes the index of the object as well.

See also

Add, Insert, Delete and GetFIFO methods;
Strings, Names, Values and MaxSize properties;
OnChange event.

10.37.3.7GetAndDeleteItem

Applies to

acStringListContainer component.

Declaration

```
function GetAndDeleteItem(Index: Integer): String;
```

Description

The GetAndDeleteItem method retreives the string in particular position of the list, then removes it from the list after retreiving. *Index* gives the position of the string, where 0 is the position of the first string, 1 is the position of the second string, and so on ("Count - 1" is the position of last string in the list).

See also

Delete, DeleteFirstItem and GetFIFO methods;
Strings, Names, Values and MaxSize properties;
OnChange event.

10.37.3.8GetFIFO

Applies to

acStringListContainer component.

Declaration

```
function GetFIFO: String;
```

Description

The GetFIFO method retreives fist string, then removes it from the list.

See also

Delete, DeleteFirstItem and GetAndDeleteItem methods;
Strings, Names, Values and MaxSize properties;
OnChange event.

10.37.3.9IndexOf

Applies to

acStringListContainer component.

© 1998-2006, UtilMind Solutions®

Declaration

```
function IndexOf(const S: String): Integer;
```

Description

The IndexOf method returns the position of a string in the list.

Call IndexOf to obtain the position of the first occurrence of the string S. IndexOf returns the 0-based index of the string. Thus, if S matches the first string in the list, IndexOf returns 0, if S is the second string, IndexOf returns 1, and so on. If the string is not in the string list, IndexOf returns -1.

Note

If the string appears in the list more than once, IndexOf returns the position of the first occurrence.

See also

IndexOf, IndexOfName and IndexOfValue methods;
Strings, Names, Values, MaxSize and Count properties.

IndexOfName
10.37.3.10

Applies to

acStringListContainer component.

Declaration

```
function IndexOfName(const S: String; IgnoreCase: Boolean = True):
Integer;
```

Description

The IndexOfName method returns the position of a string in the list, where the first part of the string (before the separator specified in NVSeparator property) is the string specified in *S* parameter.

The *IgnoreCase* parameter specifies whether the method should find the Name by specified string ignoring the case of characters (without case sensitivety, so it will find the string independently of whether you specified "sTrInG" or "String" or *S* parameter). However, you should set IgnoreCase parameter to False, if the case of character is essential.

To retreive or modify the first part of string before some separator ("name" part), in some particular position of the list — use **Names** property.

Note

If the specified first part of string ("name" part) appears in the list more than once, IndexOfName returns the position of the first occurrence.

See also

IndexOf and IndexOfValue methods;
Strings, Names, Values, MaxSize, Count and NVSeparator properties.

IndexOfValue
10.37.3.11

Applies to

acStringListContainer component.

Declaration

© 1998-2006, UtilMind Solutions®

```
function IndexOfValue(const S: String; IgnoreCase: Boolean = True):
Integer;
```

Description

The IndexOfValue method returns the position of a string in the list, where the second part of the string (after the separator specified in NVSeparator property) is the string specified in *S* parameter.

The *IgnoreCase* parameter specifies whether the method should find the Value by specified string ignoring the case of characters (without case sensitivety, so it will find the string independently of whether you specified "sTrInG" or "String" or *S* parameter). However, you should set IgnoreCase parameter to False, if the case of character is essential.

To retreive or modify the second part of string after some separator ("value" part), in some particular position of the list — use **Values** property.

Note

If the specified second part of string ("value" part) appears in the list more than once, IndexOfValue returns the position of the first occurrence.

See also

IndexOf and IndexOfName methods;
Strings, Names, Values, MaxSize, Count and NVSeparator properties.

Insert
10.37.3.12

Applies to

acStringListContainer component.

Declaration

```
procedure Insert(Index: Integer; const S: String);
```

Description

The Insert method inserts a string to the list at the position specified by Index.

Call Insert to add the string S to the list at the position specified by Index. If Index is 0, the string is inserted at the beginning of the list. If Index is 1, the string is put in the second position of the list, and so on ("Count - 1" is the position of last string in the list).

See also

Add and AddStrings methods;
Duplicates and MaxSize properties;
OnChange event.

LoadFromFile
10.37.3.13

Applies to

acStringListContainer component.

Declaration

```
procedure LoadFromFile(const FileName: String);
```

Description

The LoadFromFile method fills the list with the lines of text in a specified file.

© 1998-2006, UtilMind Solutions®

Call LoadFromFile to fill the list from the file specified by FileName. Each line in the file, as indicated by carriage return or linefeed characters, is appended as a string in the list.

Note
LoadFromFile uses the Add method to add the strings that are read from the file.

See also
SaveToFile, LoadFromRegistry, StoreToRegistry and Add methods;
Strings, Names, Values, MaxSize and Duplicates properties.

LoadFromRegistry
10.37.3.14

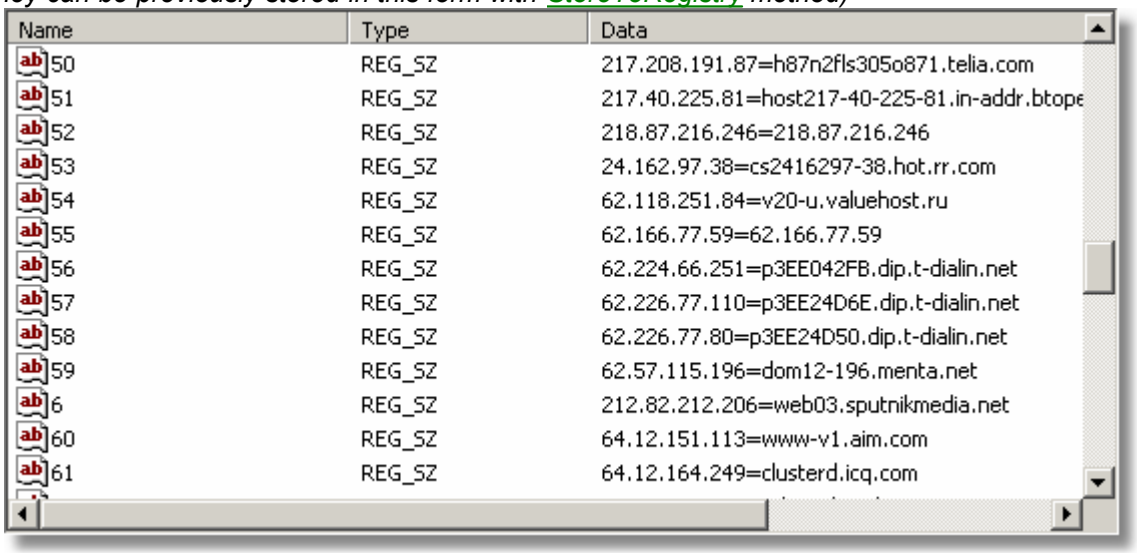
Applies to
acStringListContainer component.

Declaration
```
procedure LoadToRegistry(Reg: TRegistry); overload;
procedure LoadToRegistry(const KeyName: String; const RootKey: hKey =
HKEY_CURRENT_USER); overload;
```

Description
The LoadToRegistry method loads the strings to the from the registry.

There is 2 overloaded methods of calling the LoadFromRegisty:
1. The registry key must be previously opened and handle to TRegistry object must be passed with *Reg* parameter (see first example below);
2. You must pass the *KeyName* and the *RootKey* (HKEY_CURRENT_USER or HKEY_LOCAL_MACHINE) as parameters (see second example below).

Screenshot from RegEdit *(demonstrates how the values should be stored in registry to load them. They can be previously stored in this form with StoreToRegistry method)*



Example 1 *(if the TRegistry object already prepared before saving of the strings)*
```
uses Registry;

procedure TForm1.Button1Click(Sender: TObject);
var
```

© 1998-2006, UtilMind Solutions®

```
    MyKey: String;
    Reg: TRegistry;
  begin
    MyKey := 'SOFTWARE\My Company Name\My Program Name\StringList';

    Reg := TRegistry.Create;
    with Reg do
     try
       RootKey := HKEY_CURRENT_USER;
       if OpenKey(MyKey, False) then
         acStringListContainer1.LoadToRegistry(Reg);
     finally
       Free;
     end;
  end;
```

Example 2 *(if we want to pass the KeyName and RootKey as parameters without preparing of the TRegistry object)*

```
  procedure TForm1.Button1Click(Sender: TObject);
  begin
     acStringListContainer1.LoadToRegistry('SOFTWARE\My Company Name\My
  Program Name\StringList');
end;
```

See also

StoreToRegistry, SaveToFile and LoadFromFile methods;
Strings, Names, Values, MaxSize and Duplicates properties.

SaveToFile
10.37.3.15

Applies to

acStringListContainer component.

Declaration

```
  procedure SaveToFile(const FileName: String);
```

Description

The SaveToFile method saves the strings in the list to the specified file.

Call SaveToFile to save the strings in the list to the file specified by FileName. Each string in the list is written to a separate line in the file.

See also

LoadFromFile, LoadFromRegistry and StoreToRegistry methods;
Strings, Names, Values, MaxSize and Duplicates properties.

Sort
10.37.3.16

Applies to

acStringListContainer component.

Declaration

```
  procedure Sort;
```

Description

© 1998-2006, UtilMind Solutions®

The Sort method sorts the strings in the list in ascending order.

Call Sort to sort the strings in a list that has the Sorted property set to False. String lists with the Sorted property set to True are automatically sorted.

See also
Duplicates, Sorted and Strings properties;
Add and AddStrings methods.

StoreToRegistry
10.37.3.17

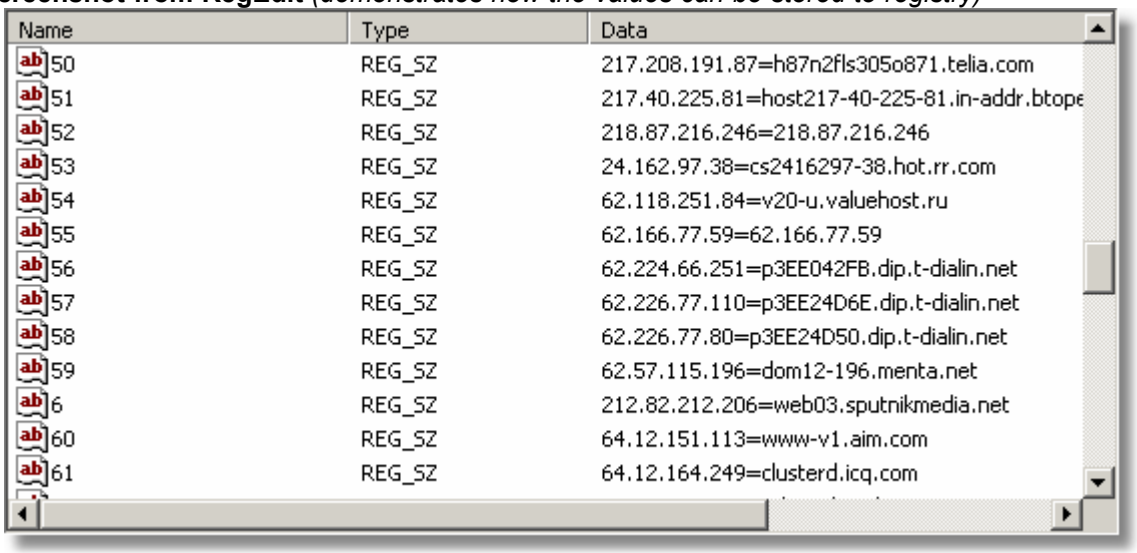
Applies to
acStringListContainer component.

Declaration
```
procedure StoreToRegistry(Reg: TRegistry); overload;
procedure StoreToRegistry(const KeyName: String; const RootKey: hKey =
HKEY_CURRENT_USER); overload;
```

Description
The StoreToRegistry method saves all content of the string list to the registry.

There is 2 overloaded methods of calling the StoreToRegisty:
1. The registry key must be previously opened and handle to TRegistry object must be passed with *Reg* parameter (see first example below);
2. You must pass the *KeyName* and the *RootKey* (HKEY_CURRENT_USER or HKEY_LOCAL_MACHINE) as parameters (see second example below).

Screenshot from RegEdit *(demonstrates how the values can be stored to registry)*



Name	Type	Data
50	REG_SZ	217.208.191.87=h87n2fls305o871.telia.com
51	REG_SZ	217.40.225.81=host217-40-225-81.in-addr.btope
52	REG_SZ	218.87.216.246=218.87.216.246
53	REG_SZ	24.162.97.38=cs2416297-38.hot.rr.com
54	REG_SZ	62.118.251.84=v20-u.valuehost.ru
55	REG_SZ	62.166.77.59=62.166.77.59
56	REG_SZ	62.224.66.251=p3EE042FB.dip.t-dialin.net
57	REG_SZ	62.226.77.110=p3EE24D6E.dip.t-dialin.net
58	REG_SZ	62.226.77.80=p3EE24D50.dip.t-dialin.net
59	REG_SZ	62.57.115.196=dom12-196.menta.net
6	REG_SZ	212.82.212.206=web03.sputnikmedia.net
60	REG_SZ	64.12.151.113=www-v1.aim.com
61	REG_SZ	64.12.164.249=clusterd.icq.com

Example 1 *(if the TRegistry object already prepared before saving of the strings)*
```
uses Registry;

procedure TForm1.Button1Click(Sender: TObject);
var
  MyKey: String;
  Reg: TRegistry;
```

© 1998-2006, UtilMind Solutions®

```
begin
  MyKey := 'SOFTWARE\My Company Name\My Program Name\StringList';

  Reg := TRegistry.Create;
  with Reg do
   try
     RootKey := HKEY_CURRENT_USER;
     DeleteKey(MyKey); // delete old values from that registry key
before store new
     if OpenKey(MyKey, True) then
       acStringListContainer1.StoreToRegistry(Reg);
   finally
     Free;
   end;
end;
```

Example 2 *(if we want to pass the KeyName and RootKey as parameters without preparing of the TRegistry object)*

```
procedure TForm1.Button1Click(Sender: TObject);
begin
   acStringListContainer1.StoreToRegistry('SOFTWARE\My Company Name\My
Program Name\StringList');
end;
```

See also

LoadFromRegistry, SaveToFile and LoadFromFile methods;
Strings, Names, Values, MaxSize and Duplicates properties.

10.37.4 Events

10.37.4.1OnChange

Applies to

acStringListContainer component.

Declaration

```
property OnChange: TNotifyEvent;
```

Description

The OnChange event specifies which event handler should execute when the contents of the string list changes.

Write OnChange event handler to take some specific actions when the string list has been changed, or use OnChanging event to be notified before the list will be changed (when it just about to be changed).

See also

OnChanging event.

10.37.4.2OnChanging

Applies to

acStringListContainer component.

Declaration

```
property OnChanging: TNotifyEvent;
```

© 1998-2006, UtilMind Solutions®

Description

The OnChanging event occurs when the content in the is about to be changed (before the actual changes).

Write OnChanging event handler to take some specific actions when the string list is about to be changed, or use OnChange to be notified when the list already has been changed.

See also

OnChanging event.

10.38 acSystemImageList

10.38.1 TacSystemImageList

Overview

The acSystemImageList gives access to the system image list. That's the image list which all icons (32x32 or 16x16) for various file types that Explorer uses. All these icons represents the files within the Windows shell.

On Win9x-family machines the list are automatically filled, however the NT operates with separate system icon lists for each process. The icon list are populated (new icons being added to it) every time when you access some object which has an icon (file, folder, drive etc). You can see how that list populated even at design-time. Drop the acSystemImageList on your form, connect it with some ListView as LargeIcons property and create some fake list items. Then try to browse the disk with "File | Open", and you will see how the list grows.

How to use ?

Just point the *Images* property of any component, which uses the ImageList's to display images, to the acSystemImageList. The icons will be taken from the list of system images.

For example, if you have the ListView which contains the list of files, you can use the acSystemImageList as source for file icons.

To retrieve *small* icons (16x16) — set the ImageSize property to `is16x16`. To retrieve *large* icons (32x32) — set the ImageSize property to `is32x32`.

To retrieve the index which corresponds to some file name, use **GetSystemIconIndex** function from acUtils unit.

Note

You can use/retrieve the icons of this list, but can NOT modify (add/delete) them.

See also

acListView and acTreeView components.

10.38.2 Properties

10.38.2.1ImageSize

Applies to

acSystemImageList component.

Declaration

```
type
  TacSystemImageSize = (is16x16, is32x32);
```

© 1998-2006, UtilMind Solutions®

```
property ImageSize: TacSystemImageSize;
```

Description

The ImageSize property specifies the size of icons in the system image list. The size of the icons can be 16x16 or 32x32.

If you are using acSystemImageList as source for images in the ListView, then point SmallImages property of the ListView to the acSystemImageList with ImageSize equal to `is16x16`, and LargeImages to the acSystemImageList where ImageSize = `is32x32`.

10.39 acSystemInfo

10.39.1 TacSystemInfo

Overview

The acSystemInfo component provides detailed system information such like machine name, current user name and organization, CPU info and frequency of main processor, serial number of hard disk, DirectX info, pathes of system directories, executable file for default browser, current screen saver, screen resolution and color mode and so forth.

Property groups

Computer / User / Network Information
ComputerName, ComputerWorkgroup, ComputerDescription - Network Information about local machine.
CurrentUserName - logon name of current user.
OwnerName, OwnerCompany - Information about machine owner.
InternetStatus - current Internet status. Online or Offline.
DefaultBrowser - path to executable of default Web browser.

CPU Information
CPUFrequency - frequency of main (first) processor in megaherzes.
CPUArchitecture, CPUIdentifier, CPUType, CPUVendor - some information about main processor.
CPUsNumber - number of processors.

Display / Screen Information
ScreenColorMode, ScreenResolution - information about current screen resolution and color mode. You may change the screen resolution changing values of these properties.
ScreenSaver - path current screen saver.

Windows Version Information
WinNT - returns True if current OS is Windows NT or 2000. Same as IsNT routine.
OSVersion, WinVersion, WinVersionBuild, WinVersionMajor, WinVersionMinor, WinVersionNumber, WInVersionSubNumber - Windows version information.
WinProductID, WinProductKey, WinProductName - product information.

System Directories
DirAppData, DirCache, DirCommonFiles, DirCookies, DirDesktop, DirFavorites, DirFonts, DirHistory, DirMedia, DirMyPictures, DirNetHood, DirPersonal, DirPrintHood, DirProgramFiles, DirPrograms, DirRecent, DirSendTo, DirStartMenu, DirStartup, DirSystem, DirTemp, DirTemplates, DirWindows, SystemPathes.

RAM Information
MemAvailPhisical, MemAvailVirtual, MemTotalPhisical, MemTotalVirtual - amount of total and avialable memory.
MemLoad - memory load in percents.

© 1998-2006, UtilMind Solutions®

Disk Serial # Information
DiskCSerialNumber - integer value of serial number of disk C:.
DiskCSerialNumberStr - string value of serial numbr of disk C: in "XXXX-XXXX" format.

Note: If you need to get much more information your disks, files and file system, shell and so on, please check out DiskControls pack at http://www.appcontrols.com/diskcontrols.html

10.39.2 Properties

10.39.2.1Allow_Edit

Applies to
acSystemInfo component.

Declaration
```
property ALLOW_EDIT: Boolean;
```

Description
The ALLOW_EDIT property included by security issues, to disallow you to make accidental changes. If you would like to edit values of other properties, make sure that ALLOW_EDIT property is True.

10.39.2.2ComputerDescription

Applies to
acSystemInfo component.

Declaration
```
property ComputerDescription: String;
```

Description
The ComputerDescription property determines the network description of computer. The description of computer usually can be specified at the Control Panel / Network properties, and means an optional comment that other people see when they look at your computer on the network. Use it to determineate or change the description of computer (for example, your name, department, or location) or the type of information on it that you're sharing.

Remarks
This property available on Windows 95/98/ME only!

See also
ComputerName and ComputerWorkgroup properties.

10.39.2.3ComputerName

Applies to
acSystemInfo component.

Declaration
```
property ComputerName: String;
```

Description
The ComputerName property determines the network name of computer. The name of computer usually specified at the Control Panel / Network properties and identifies user's computer to other people on the network. You can give your computer a unique name of up to 15 characters. The name cannot include blank spaces.

© 1998-2006, UtilMind Solutions®

See also
ComputerDescription and ComputerWorkgroup properties.

10.39.2.4ComputerWorkgroup

Applies to
acSystemInfo component.

Declaration
```
property ComputerWorkgroup: String;
```

Description
The ComputerWorkgroup property identifies the group of computers that user's computer is in. The computer's workgroup usually specified at the Control Panel / Network properties and can be 15 characters length. A workgroup is generally composed of the computers you are most likely to communicate with, containing most of the network resources (such as files and printers) you will want to use.

Remarks
This property available on Windows 95/98/ME only!

See also
ComputerDescription and ComputerName properties.

10.39.2.5CPUArchitecture

Applies to
acSystemInfo component.

Declaration
```
type
  TacCPUArchitecture = (paINTEL, paMIPS, paALPHA, paPPC, paUNKNOWN);

property CPUArchitecture: TacCPUArchitecture; // Read only!
```

Description
The CPUArchitecture property determines the system's processor archytecture. This value can be INTEL on any Windows versions and INTEL, MIPS, ALPHA or PPC on Windows NT machines.

See also
CPUFrequency, CPUIdentifier, CPUType, CPUVendor properties.

10.39.2.6CPUFrequency

Applies to
acSystemInfo component.

Declaration
```
property CPUFrequency: Integer; // Read only!
```

Description
The CPUFrequency determines the frequency of first processor available on current machine. The return value is the processor speed in megaherzes (MHz).

See also
CPUArchitecture, CPUIdentifier, CPUType, CPUVendor properties.

© 1998-2006, UtilMind Solutions®

10.39.2.7CPUIdentifier

Applies to
acSystemInfo component.

Declaration
```
property CPUIdentifier: String; // Read only!
```

Description
The CPUIdentifier property determines the string which identifies the processor. For example, "Pentium(r) III Processor".

See also
CPUArchitecture, CPUFrequency, CPUType, CPUVendor properties.

10.39.2.8CPUsNumber

Applies to
acSystemInfo component.

Declaration
```
property CPUsNumber: Integer; // Read only!
```

Description
The CPUsNumber property determines the number of processors available on current machine.

See also
CPUArchitecture, CPUFrequency, CPUIdentifier, CPUType, CPUVendor properties.

10.39.2.9CPUType

Applies to
acSystemInfo component.

Declaration
```
type
  TacCPUType = (ptIntel386, ptIntel486, ptPentium, ptMIPS_R4000,
ptALPHA_21064, ptUNKNOWN);

property CPUType: TacCPUType; // Read only!
```

Description
The CPUType determines the processor type. See also CPUArchitecture.

See also
CPUArchitecture, CPUFrequency, CPUIdentifier, CPUVendor properties.

CPUVendor 10.39.2.10

Applies to
acSystemInfo component.

Declaration
```
property CPUVendor: String; // Read only!
```

Description

© 1998-2006, UtilMind Solutions®

The CPUVendor property determines the name of processor vendor.

See also
CPUArchitecture, CPUFrequency, CPUIdentifier, CPUType properties.

CurrentUserName
10.39.2.11

Applies to
acSystemInfo component.

Declaration
```
property CurrentUserName: String;
```

Description
The CurrentUserName determines or changes (if Allow_Edit is True) the name of current user.

See also
OwnerName and OwnerCompany properties.

DefaultBrowser
10.39.2.12

Applies to
acSystemInfo component.

Declaration
```
property DefaultBrowser: String; // Read only!
```

Description
The DefaultBrowser property determines the location of file which executes default Web browser. For example, "C:\Program Files\Internet Explorer\iexplore.exe".

DirAppData
10.39.2.13

Applies to
acSystemInfo component.

Declaration
```
property DirAppData: String;
```

Description
The DirAppData determines location of the system directory used for application data, for current user. For example, "C:\WINDOWS\Application Data".

See also
DirAppDataAll property.

DirAppDataAll
10.39.2.14

Applies to
acSystemInfo component.

Declaration
```
property DirAppDataAll: String;
```

© 1998-2006, UtilMind Solutions®

Description
The DirAppDataAll determines location of the system directory used for application data, for all users of local PC. For example, "C:\WINDOWS\All Users\Application Data".

See also
DirAppData property.

DirCache
10.39.2.15

Applies to
acSystemInfo component.

Declaration
```
property DirCache: String;
```

Description
The DirCache determines location of the cache directory for temporary internet files. For example, "C:\WINDOWS\Temporary Internet Files".

See also
DirCookies property.

DirCommonFiles
10.39.2.16

Applies to
acSystemInfo component.

Declaration
```
property DirCommonFiles: String;
```

Description
The DirCommonFiles determines the path to the system directory where applications can store their commonly used files. For example "C:\Program Files\Common Files".

DirCookies
10.39.2.17

Applies to
acSystemInfo component.

Declaration
```
property DirCookies: String;
```

Description
The DirCookies property determines location of directory where the standard Internet application stores the cookies received from the Web. For example, "C:\WINDOWS\Cookies".

See also
DirCache property.

DirDesktop
10.39.2.18

Applies to
acSystemInfo component.

© 1998-2006, UtilMind Solutions®

Declaration

```
property DirDesktop: String;
```

Description

The DirDesktop property determines the path to the desktop folder for current user. For example, "C:\WINDOWS\Desktop".

See also

DirStartMenu and DirPrograms properties.

DirFavorites
10.39.2.19

Applies to

acSystemInfo component.

Declaration

```
property DirFavorites: String;
```

Description

The DirFavorites retrieves the folder which contains the Internet bookmarks.

DirFonts
10.39.2.20

Applies to

acSystemInfo component.

Declaration

```
property DirFonts: String;
```

Description

The DirFonts property retrieves the path to the directory which contains the system fonts. For example, "C:\WINDOWS\Fonts".

DirHistory
10.39.2.21

Applies to

acSystemInfo component.

Declaration

```
property DirHistory: String;
```

Description

The DirHistory property retrieves the path to directory which contains the information about recently visited Web sites.

DirMedia
10.39.2.22

Applies to

acSystemInfo component.

Declaration

```
property DirMedia: String;
```

© 1998-2006, UtilMind Solutions®

Description
The DirMedia property retrieves path to the directory which contains Windows media files (i.e. sounds for various Windows themes, some MIDI-samples etc).

DirMyPictures
10.39.2.23

Applies to
acSystemInfo component.

Declaration
```
property DirMyPictures: String;
```

Description
The DirMyPictures determines the path to the directory used to store picture files. For example, "C:\WINDOWS\Personal\My Pictures".

DirNetHood
10.39.2.24

Applies to
acSystemInfo component.

Declaration
```
property DirNetHood: String;
```

Description
The DirNetHood property is the path to the system directory containing objects that appear in the network neighborhood.

DirPersonal
10.39.2.25

Applies to
acSystemInfo component.

Declaration
```
property DirPersonal: String;
```

Description
The DirPersonal is the file system directory that serves as a common respository for documents.

DirPrintHood
10.39.2.26

Applies to
acSystemInfo component.

Declaration
```
property DirPrintHood: String;
```

Description
The DirPrintHood retrieves the path to the to the system directory containing objects that appear in the printhood.

© 1998-2006, UtilMind Solutions®

DirProgramFiles
10.39.2.27

Applies to
acSystemInfo component.

Declaration
```
property DirProgramFiles: String;
```

Description
The DirProgramFiles property retrieves the path to th directory used for program files. For example, "C:\Program Files".

DirPrograms
10.39.2.28

Applies to
acSystemInfo component.

Declaration
```
property DirPrograms: String;
```

Description
The DirPrograms is the file system directory that contains the user's program groups (which are also file system directories).

Press "Start" button and select "Programs" menu item to check out contents of this directory.

See also
DirStartMenu property.

DirRecent
10.39.2.29

Applies to
acSystemInfo component.

Declaration
```
property DirRecent: String;
```

Description
The DirRecent property determines the directory that contains the user's most recently used documents. For example, "C:\WINDOWS\Recent".

DirSendTo
10.39.2.30

Applies to
acSystemInfo component.

Declaration
```
property DirSendTo: String;
```

Description
The DirSendTo is the file system directory that contains "Send To" menu items.

© 1998-2006, UtilMind Solutions®

DirStartMenu
10.39.2.31

Applies to
acSystemInfo component.

Declaration
```
property DirStartMenu: String;
```

Description
The DirStartMenu is the file system directory containing Start menu items.

Press "Start" button to check out contents of this directory.

See also
DirPrograms and DirDesktop properties.

DirStartup
10.39.2.32

Applies to
acSystemInfo component.

Declaration
```
property DirStartup: String;
```

Description
The DirStartup is the file system directory that corresponds to the user's Startup program group. All files in this folder will be automatically executed on every Windows startup for current user.

Startup folder in system menu may also contain entries for all PC users. To determinate folder with files for all users of local machine — use DirStartupAll property.

See also
DirStartupAll property.

DirStartupAll
10.39.2.33

Applies to
acSystemInfo component.

Declaration
```
property DirStartup: String;
```

Description
The DirStartup is the file system directory that corresponds to the Startup program group for all users of current PC. All files in this folder will be automatically executed on every Windows startup for *every* user of current PC.

See also
DirStartup property.

© 1998-2006, UtilMind Solutions®

DirSystem
10.39.2.34

Applies to
acSystemInfo component.

Declaration
```
property DirSystem: String;
```

Description
The DirSystem property retrieves the path of the Windows system directory. For example, "C:\WINDOWS\SYSTEM".

The system directory contains such files as Windows libraries, drivers, and font files.

See also
GetSystemDir function;
DirWindows and DirTemp properties.

DirTemp
10.39.2.35

Applies to
acSystemInfo component.

Declaration
```
property DirTemp: String;
```

Description
The DirTemp property retrieves the path of the directory designated for temporary files. For example, "C:\WINDOWS\TEMP".

See also
GetTempDir function;
DirWindows and DirSystem properties.

DirTemplates
10.39.2.36

Applies to
acSystemInfo component.

Declaration
```
property DirTemplates: String;
```

Description
The DirTemplates is the file system directory that serves as a common repository for document templates.

DirWindows
10.39.2.37

Applies to
acSystemInfo component.

Declaration
```
property DirWindows: String;
```

© 1998-2006, UtilMind Solutions®

Description
The DirWindows retrieves the path to the Windows directory. For example, "C:\WINDOWS".

The Windows directory contains such files as Windows-based applications, initialization files, and Help files.

See also
GetWindowsDir function;
DirSystem and DirTemp properties.

DiskCSerialNumber
10.39.2.38

Applies to
acSystemInfo component.

Declaration
```
property DiskCSerialNumber: Integer; // Read only!
```

Description
The DiskCSerialNumber determines the serial number of disk C: in integer format. Serial number is unique for every disk and can be changed only after re-formatting of the disk.

See also
DiskCSerialNumberStr property.

DiskCSerialNumberStr
10.39.2.39

Applies to
acSystemInfo component.

Declaration
```
property DiskCSerialNumberStr: String; // Read only!
```

Description
The DiskCSerialNumber determines the serial number of disk C: in string format. For example, "1647-08F4".

Serial number is unique for every disk and can be changed only after re-formatting of the disk.

See also
DiskCSerialNumber property.
Try also the DiskControls pack.

InternetStatus
10.39.2.40

Applies to
acSystemInfo component.

Declaration
```
type
  TacInternetStatus = (iOffline, iOnline);

property InternetStatus: TacInternetStatus; // Read only!
```

© 1998-2006, UtilMind Solutions®

Description
The InternetStatus procedure returns current online status of the computer. It returns `iOnline` when user currently connected to the Internet and `iOffline` otherwise.

See also
IsOnline function.

MemAvailPhisical
10.39.2.41

Applies to
acSystemInfo component.

Declaration
```
property MemAvailPhisical: Integer; // read-only!
```

Description
The MemAvailPhisical property indicates the number of bytes of physical memory available.

See also
MemAvailPhisical, MemAvailVirtual, MemTotalPhisical and MemTotalVirtual properties.

MemAvailVirtual
10.39.2.42

Applies to
acSystemInfo component.

Declaration
```
property MemAvailVirtual: Integer; // read-only!
```

Description
The MemAvailVirtual property indicates the number of bytes of virtual memory available.

See also
MemAvailPhisical, MemAvailVirtual, MemTotalPhisical and MemTotalVirtual properties.

MemLoad
10.39.2.43

Applies to
acSystemInfo component.

Declaration
```
property MemLoad: Integer; // read-only!
```

Description
The MemLoad property determines a number between 0 and 100 that gives a general idea of current memory utilization, in which 0 indicates no memory use and 100 indicates full memory use.

MemTotalPhisical
10.39.2.44

Applies to
acSystemInfo component.

Declaration

© 1998-2006, UtilMind Solutions®

```
property MemTotalPhisical: Integer; // read-only!
```

Description
The MemTotalPhisical property indicates the total number of bytes of physical memory.

See also
MemAvailPhisical, MemAvailVirtual, MemTotalPhisical and MemTotalVirtual properties.

MemTotalVirtual
10.39.2.45

Applies to
acSystemInfo component.

Declaration
```
property MemTotalVirtual: Integer; // read-only!
```

Description
The MemTotalVirtual property indicates the total number of bytes of virtual memory.

See also
MemAvailPhisical, MemAvailVirtual, MemTotalPhisical and MemTotalVirtual properties.

OSVersion
10.39.2.46

Applies to
acSystemInfo component.

Declaration
```
type
  TacOSVersion = (osUnknown, os95, os95OSR2, os98, os98SE, osNT3, osNT4,
os2K, osME, osXP, os2003Server);

property OSVersion: TacOSVersion; // read-only!
```

Description
The OSVersion property determines the operation system on user's machine.

Value	Meaning
os95	Windows 95;
os95OSR2	Windows 95, OSR 2 edition;
os98	Windows 98;
os98SE	Windows 98, SE edition;
osNT3	Windows NT3;
osNT4	Windows NT4;
os2K	Windows 2000;
osME	Windows ME (Millenium);
osXP	Windows XP
os2003Server	Windows 2003 Server **or better!**;
osUnknown	Unknown system.

See also
WinNT property and GetOS routine.

© 1998-2006, UtilMind Solutions®

OwnerCompany
10.39.2.47

Applies to
acSystemInfo component.

Declaration
```
property OwnerCompany: String;
```

Description
The OwnerCompany property determines and changes (if Allow_Edit is True) the company name of machine owner.

See also
OwnerName property.

OwnerName
10.39.2.48

Applies to
acSystemInfo component.

Declaration
```
property OwnerName: String;
```

Description
The OwnerName property determines and changes (if Allow_Edit is True) the name of machine owner.

See also
OwnerCompany property.

ScreenColorMode
10.39.2.49

Applies to
acSystemInfo component.

Declaration
```
type
  TacScreenColorMode = (sm4bit, sm8bit, sm16bit, sm24bit, sm32bit);

property ScreenColorMode: TacScreenColorMode;
```

Description
The ScreenColorMode property determines and changes (if Allow_Edit is True) the screen color mode. Screen modes can be 4bit, 8bit, 16bit, 24bit and 32bit.

The ScreenColorMode works both at run- and design-time. You can play with it changing the property value in the Delphi/BCB IDE.

See also
ScreenResolution property.

© 1998-2006, UtilMind Solutions®

ScreenResolution
10.39.2.50

Applies to
acSystemInfo component.

Declaration
```
type
  TacScreenResolution = (sr640x480, sr800x600, sr1024x768,
                         sr1152x864, sr1280x960, sr1280x1024,
                         sr1600x1200, sr1920x1080, sr1920x1200,
                         // pivot
                         sr480x640, sr600x800, sr768x1024,
                         sr864x1152, sr960x1280, sr1024x1280,
                         sr1200x1600, sr1080x1920, sr1200x1920);

property ScreenResolution: TacScreenResolution;
```

Description
The ScreenResolution property determines and changes (if Allow_Edit is True) the screen resolution. Screen resolution can be 640x480, 800x600, 1024x768, 1152x864, 1280x960, 1280x1024 and higher. In case if used the Pivot monitor (with portrait display), the resolutions will be 480x640, 600x800, 768x1024, 864x1152, 960x1280, 1024x1280 and higher.

The ScreenResolution works both at run- and design-time. You can play with it changing the property value in the Delphi/BCB IDE.

However, please remember that some old video cards may not support high resolutions.

See also
ScreenColorMode and PivotMonitor properties.

ScreenSaver
10.39.2.51

Applies to
acSystemInfo component.

Declaration
```
property ScreenSaver: String;
```

Description
The ScreenSaver property determines the path and file name of current screen saver.

See also
StartScreenSaver procedure.

SystemPathes
10.39.2.52

Applies to
acSystemInfo component.

Declaration
```
property SystemPathes: String;
```

© 1998-2006, UtilMind Solutions®

Description
The SystemPathes property retrieves the string value of the evnironment variable, which identifies the searching pathes of Windows.

See also
GetEnvironmentString function.

SystemUpTime
10.39.2.53

Applies to
acSystemInfo component.

Declaration
```
property SystemUpTime: Integer;
```

Description
The SystemUpTime property determines the number of seconds that passes after system startup.

WinDirectXVersion
10.39.2.54

Applies to
acSystemInfo component.

Declaration
```
property WinDirectXVersion: String;
```

Description
The WinDirectXVersion determines the version of the DirectX drivers. DirectX is a set of technologies that assist in multimedia and games programming.

WinNT
10.39.2.55

Applies to
acSystemInfo component.

Declaration
```
property WinNT: Boolean; // Read only!
```

Description
The WinNT property returns True if the program running on Windows NT-based system, and False if user's system is Windows 95/98/ME.

See also
OSVersion property;
IsNT and IsWin2k functions.

WinProductID
10.39.2.56

Applies to
acSystemInfo component.

Declaration
```
property WinProductID: String;
```

© 1998-2006, UtilMind Solutions®

Description
The WinProductID property determines the serial number of current Windows version.

This value located in the "HKLM\Software\Microsoft\Windows\CurrentVersion" registry key. Alternatively, you can determinate it reading the ProductId data value.

WinProductKey
10.39.2.57

Applies to
acSystemInfo component.

Declaration
```
property WinProductKey: String;
```

Description
The WinProductID property determines the product key of current Windows version.

This value located in the "HKLM\Software\Microsoft\Windows\CurrentVersion" registry key. Alternatively, you can determinate it reading the ProductKey data value.

WinProductName
10.39.2.58

Applies to
acSystemInfo component.

Declaration
```
property WinProductName: String;
```

Description
The WinProductName property determines the product name of current Windows version.

This value located in the "HKLM\Software\Microsoft\Windows\CurrentVersion" registry key. Alternatively, you can determinate it reading the ProductName data value.

WinVersion
10.39.2.59

Applies to
acSystemInfo component.

Declaration
```
property WinVersion: String;
```

Description
The WinVersion property determines the string which identifies the Windows version.

For example, in Windows 95 this property will be "Windows 95", in Windows 98 it will be "Windows 98" and so forth...

This value located in the "HKLM\Software\Microsoft\Windows\CurrentVersion" registry key. Alternatively, you can determinate it reading the Version data value.

© 1998-2006, UtilMind Solutions®

WinVersionBuild
10.39.2.60

Applies to
acSystemInfo component.

Declaration
```
property WinVersionBuild: Integer;
```

Description

WinVersionMajor
10.39.2.61

Applies to
acSystemInfo component.

Declaration
```
property WinVersionMajor: Integer;
```

Description

WinVersionMinor
10.39.2.62

Applies to
acSystemInfo component.

Declaration
```
property WinVersionMinor: Integer;
```

Description

WinVersionNumber
10.39.2.63

Applies to
acSystemInfo component.

Declaration
```
property WinVersionNumber: String;
```

Description

WinVersionSubNumber
10.39.2.64

Applies to
acSystemInfo component.

Declaration
```
property WinVersionSubNumber: String;
```

© 1998-2006, UtilMind Solutions®

Description

10.40 acTextTemplateConverter

10.40.1 TacTextTemplateConverter

Overview

The acTextTemplateConverter is the utility which translates some specified %keywords% inside the text into some specified values. It can be used as plug-in for acSendMail or acHTMLViewer components to replace some keywords in email/HTML templates (like %recipient_email% or %sender_name%) to their actual values.

How to use?

Drop component on your form and specify some keywords which you are using in your text templates to the **Name** properties of Params collection. Also specify the text strings, which should be replace that keywords, to the **Value** propertes of Params. You can modify the collection of Names/Values both at design and run-time.

When your program calls the **Convert** method, it will find all the keywords (specified in *Name* properties) within the text template and replace them to their actual values (specified in *Value* properties). For example, our Params property contains 2 following items:

Item1:
```
  Name  = %RECIPIENT_NAME%
  Value = Bill Gates
```
Item2:
```
  Name  = %RECIPIENT_EMAIL%
  Value = billgates@microsoft.com
```

Plus we have the text template, which contains %RECIPIENT_NAME% and %RECIPIENT_EMAIL% keywords instead of real name and email address. For example, the text like:

```
To: %RECIPIENT_NAME% <%RECIPIENT_EMAIL%>
Dear %RECIPIENT_NAME%,
etc, etc...
```

After calling the Convert method, all the keywords specified in Name property of each item will be replaced by the words specified in Value property. For example, after convertion, the text displayed above will looks like follows:

```
To: Bill Gates <billgates@microsoft.com>
Dear Bill Gates,
etc, etc...
```

The keywords can looks like **%KEYNAME%**, or **<KEYNAME>**, or just **KEYNAME**, whatever you like. In most cases you should specify the keywords which are used in your text templates just once, and you can do it at design-time and don't modify them at run-time at all.

However, your program should specify required *Values* every time before processing of the template (before the program calls Convert method to translate the keywords into the values). To specify the *Values* for each required keyword (*Names*) you can use ParamByName method, or just enumerate each item of Params collection. Here is some examples:

```
// how to add the the template parameters at run-time
var
  Param: TacTemplateParameter;
begin
  Param := acTextTemplateConverter1.Params.Add;
```

© 1998-2006, UtilMind Solutions®

```
  Param.Name := '%keyword%';
  Param.Value := 'Value';

// how to modify the template values at run-time
acTextTemplateConverter1.ParamByName('%keyword%').Value := 'value';
acTextTemplateConverter1.ParamByValue('value').Name :=
'%another_keyword%';

// also you can enumerate all parameters
var
  I: Integer;
begin
  I := acTextTemplateConverter1.Count;
  if I <> 0 then
   for I := 0 to I - 1 do
    with acTextTemplateConverter1 do
     Caption := Params[I].Name + ' = ' + Params[I].Value;
end;
```

If you are using this component as plug-in for acSendMail component, to replace the keywords in email templates, you don't need to call Convert method. All that you need is just point the acTextTemplateConverter to the TemplateConverter property of acSendMail component and all the text of outgoing email, the body of message and the message headers will be translated automatically.

Another component which automatically supports the TemplateConverter is acHTMLViewer. You can specify the HTML source to Template property and speicify parameters (to Params). The HTML document will be refreshed accordingly and specify the required text instead of keywords-parameters.

See also
acSendMail and acHTMLViewer components.

10.40.2 Properties

10.40.2.1IgnoreCaseOfParams

Applies to
acTextTemplateConverter component.

Declaration
```
property IgnoreCaseOfParams: Boolean; // True by default
```

Description
The IgnoreCaseOfParams property controls whether the component should ignore the case of character on convertion, when it checks the text for the keywords described in **Name** property of Params collection.

Set IgnoreCaseOfParams to True, and Convert method will replace the keywords with their values independently of case of characters, so the keyword like **%kEyWoRd%** is the same as **%Keyword%**. Otherwise, set it to False and the Convert method will find only keywords with the same charcase as specified in the Name properties of Params items.

See also
Params property;
Convert method.

© 1998-2006, UtilMind Solutions®

10.40.2.2Params

Applies to

acTextTemplateConverter component.

Declaration

```
type
  TacTemplateParameter = class(TCollectionItem)
  published
    property Name: String;
    property Value: String;
  end;

  TacTemplateParameters = class(TCollection)
  public
    function Add: TacTemplateParameter;
    function Convert(const Value: String): String;

    property Owner: TComponent read FOwner;
    property IgnoreCase: Boolean;
    property Items[Index: Integer]: TacTemplateParameter; default;
  end;

property Params: TacTextTemplateParameters; // True by default
```

Description

The Params is the collection of keywords (and their values) which should be converted in the template.

Each item of Params collection is the object which contains 2 properties: **Name** and **Value**. The Name property specifies the template keyword which can be used in the text templates, and Value property is the actual string which sould replace the template keyword.

For example, our Params property contains 2 following items:

Item1:
```
  Name  = %RECIPIENT_NAME%
  Value = Bill Gates
```
Item2:
```
  Name  = %RECIPIENT_EMAIL%
  Value = billgates@microsoft.com
```

Plus we have the text template, which contains %RECIPIENT_NAME% and %RECIPIENT_EMAIL% keywords instead of real name and email address. For example, the text like:

```
To: %RECIPIENT_NAME% <%RECIPIENT_EMAIL%>
Dear %RECIPIENT_NAME%,
etc, etc...
```

After calling the Convert method, all the keywords specified in Name property of each item will be replaced by the words specified in Value property. For example, after convertion, the text displayed above will looks like follows:

```
To: Bill Gates <billgates@microsoft.com>
Dear Bill Gates,
etc, etc...
```

Examples

```
// how to add the the template parameters at run-time
```

© 1998-2006, UtilMind Solutions®

```
var
  Param: TacTemplateParameter;
begin
  Param := acTextTemplateConverter1.Params.Add;
  Param.Name := '%keyword%';
  Param.Value := 'Value';

// how to modify the template values at run-time
acTextTemplateConverter1.ParamByName('%keyword%').Value := 'value';
acTextTemplateConverter1.ParamByValue('value').Name :=
'%another_keyword%';

// also you can enumerate all parameters
var
  I: Integer;
begin
  I := acTextTemplateConverter1.Count;
  if I <> 0 then
   for I := 0 to I - 1 do
    with acTextTemplateConverter1 do
     Caption := Params[I].Name + ' = ' + Params[I].Value;
end;
```

See also

IgnoreCaseOfParams property;
ParamByName, ParamByValue and Convert methods.

10.40.2.3Template

Applies to

acTextTemplateConverter component.

Declaration

```
property Template: TStringList;
```

Description

The Template property can be used in 2 cases:

1. To convert some text without using the Convert method (you can read the converted template, with required text instead of keywords-parameters, from TemplateOutput property).
2. If you're using the acTextTemplateConverter as template for acHTMLViewer component. In this case you need to specify the source of HTML document to Template property and the keywords-parameters to Params property. The content of acHTMLViewer will be refreshed accordingly to all changes.

See also

Params property;
OnTemplateChanged event;
acHTMLViewer component.

10.40.3 Methods

10.40.3.1Convert

Applies to

acTextTemplateConverter component.

Declaration

© 1998-2006, UtilMind Solutions®

```
function Convert(const Value: String): String;
```

Description

The Convert method checks the template string specified in Value parameter, found in that string all the **%keywords%** specified in **Name** property of Params collection, replaces the **%keywords%** with the text specified in **Value** property of Params.

Function returns the translated string, where all template keywords are replaced with their values.

See also

Params property;
ConvertStrings and ConvertStream methods.

10.40.3.2ConvertStream

Applies to

acTextTemplateConverter component.

Declaration

```
procedure ConvertStream(Stream: TStream);
```

Description

The ConvertStream method is the same as Convert, but translates the text in the streams or their descendants (MemoryStream's, FileStream's etc).

It checks the template string specified in Value parameter, found in that string all the **%keywords%** specified in **Name** property of Params collection, replaces the **%keywords%** with the text specified in **Value** property of Params.

After calling the ConvertStream method, the Stream will contain translated text, where all keywords replaced with their actual values.

Example

```
procedure TForm1.Button1Click(Sender: TObject);
var
  fs: TFileStream;
begin
  fs := TFileStream.Create('c:\file.txt', fmOpenReadWrite);
  try
    acTextTemplateConverter1.ConvertStream(fs);
  finally
    fs.Free;
  end;
end;
```

See also

Params property;
Convert and ConvertStrings methods.

10.40.3.3ConvertStrings

Applies to

acTextTemplateConverter component.

Declaration

```
procedure ConvertStrings(Strings: TStrings);
```

© 1998-2006, UtilMind Solutions®

Description
The ConvertStrings method is the same as Convert, but translates the string lists.

It checks the template string specified in Value parameter, found in that string all the **%keywords%** specified in **Name** property of Params collection, replaces the **%keywords%** with the text specified in **Value** property of Params.

After calling the ConvertStrings method, the Strings parameter will contain translated text, where all keywords replaced with their actual values.

Example
```
ConvertStrings(Memo1.Lines);
```

See also
Params property;
Convert and ConvertStream methods.

10.40.3.4ConvertToFile

Applies to
acTextTemplateConverter component.

Declaration
```
function ConvertToFile(const Template, FileName: String): Boolean;
```

Description
The ConvertToFile converts the text *Template* (translates all keywords specified in Params property) and saves the converted template to the file specified *FileName* parameter.

Example
```
procedure TForm1.Button1Click(Sender: TObject);
var
  I: Integer;
begin
  if acStringListContainer1.Count <> 0 then
   for I := 0 to acStringListContainer1.Count - 1 do
    begin
      acTemplateConverter1.ParamByName('%product_name%').Value :=
acStringListContainer1.Values[I];
     acTemplateConverter1.ParamByName('%product_id%').Value :=
acStringListContainer1.Names[I];
      acTemplateConverter1.ConvertToFile(TemplateMemo.Lines.Text,
                                         'output\' +
acStringListContainer1.Names[I] + '.html');
    end;
end;
```

See also
Params property;
Convert, ConvertStrings and ConvertStream methods.

10.40.3.5ParamByName

Applies to
acTextTemplateConverter component.

Declaration

© 1998-2006, UtilMind Solutions®

```
type
  TacTemplateParameter = class(TCollectionItem)
  published
    property Name: String;
    property Value: String;
  end;

function ParamByName(const ParamName: String): TacTemplateParameter;
```

Description

The ParamByName method returns the pointer to TacTemplateParameter object where the *Name* property is equal to the *ParamName* parameter.

You can use this method to quickly find the keyword and modify its *Value*.

Function returns NIL if the specified *ParamName* was not found in Params collection.

Example

```
acTemplateConverter1.ParamByName('%keyword%').Value := 'New Value for
Keyword';
```

See also

Params property;
ParamByValue and Convert methods.

10.40.3.6ParamByValue

Applies to

acTextTemplateConverter component.

Declaration

```
type
  TacTemplateParameter = class(TCollectionItem)
  published
    property Name: String;
    property Value: String;
  end;

function ParamByValue(const ParamValue: String): TacTemplateParameter;
```

Description

The ParamByValue method returns the pointer to TacTemplateParameter object where the *Value* property is equal to the *ParamValue* parameter.

You can use this method to quickly find some value in the Params collection and modify its *Name*.

Function returns NIL if the specified *ParamValue* was not found in Params collection.

Example

```
acTemplateConverter1.ParamByValue('The Value').Name :=
'%another_keyword%';
```

See also

Params property;
ParamByName and Convert methods.

© 1998-2006, UtilMind Solutions®

10.40.4 Events

10.40.4.1OnParamChanged

Applies to
acTextTemplateConverter component.

Declaration
```
type
  TacTemplateParameterChangedEvent = procedure(Sender: TObject; Param:
TacTemplateParameter) of object;

property OnParamChanged: TacTemplateParameterChangedEvent;
```

Description
The OnParamChanged event occurs when some item listed in Params property changes.

See also
Params property;
OnTemplateChanged event.

10.40.4.2OnTemplateChanged

Applies to
acTextTemplateConverter component.

Declaration
```
property OnTemplateChanged: TNotifyEvent;
```

Description
The OnTemplateChanged event occurs when the content of Template changes.

See also
Template property;
OnParamChanged event.

10.41 acTrackBar

10.41.1 TacTrackBar

Overview

How to use?

10.42 acTreeComboBox

10.42.1 TacTreeComboBox

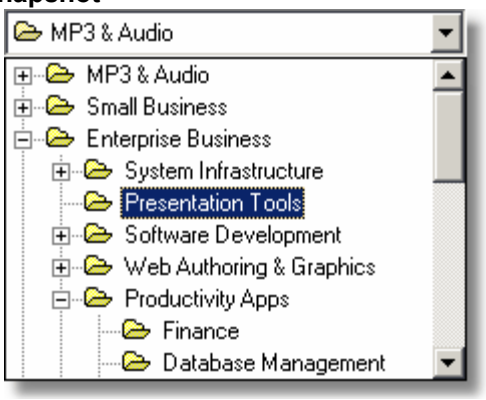
Overview
The TreeComboBox is the combo-box with the TreeView embedded to its drop-down window. So, instead of searching and selecting the necessary list item from huge list of drop-down window, user will be able to see an elegant tree-like structure, select the tree nodes, expand and collapse them, export and import the nodes to another TreeView controls and so on.

Like in standard TreeView, the TreeComboBox can display images for each tree node, show image

© 1998-2006, UtilMind Solutions®

in editor, accept the selection either on one or double click, contains the label attached to control, customizeable image for button + many other neat features, result of combination of ComboBox and TreeView.

Snapshot



How to use?

Drop acTreeComboBox control onto your form and specify the tree nodes to Items property + if the list items should contain an icons — specify the image list to Images property. In general, the TreeComboBox are ready for work… But let's change some its behaviours… For example we want the tree nodes to be selected by double click intead of one, and don't like that selection hovers the nodes under mouse. Then just set AcceptOnDblClick property to True (the nodes will be selected on double click), and set TreeOptions.MouseTrack property to False (mouse will not track the selection in drop-down view).

To change the selected node at run- or design-time — modify ItemIndex property (this is the absolute index of selected node).

To change the width and height of the drop-down window — set DropDownCount and DropDownWidth properties. To let user to enter or modify the text using keyboard — set ShowEditor property to True.

The EmptyItemText and EmptyItemImageIndex properties can be used to specify the text and image which should be displayed in the box when no tree node currently selected.

To take some specific actions when the drop-down window appears or disappears — write OnDropDown, OnAfterDropDown and OnCloseUp event handlers.

Also the tree nodes of TreeComboBox can be exported or imported from another containers of TTreeNodes object, like standard TreeView control. Check out ExportToTreeNodes and ImportFromTreeNodes methods for more details.

See also

acEdit and acTreeView components.

10.42.2 Properties

10.42.2.1AcceptOnDblClick

Applies to

acTreeComboBox component.

Declaration

© 1998-2006, UtilMind Solutions®

```
property AcceptOnDblClick: Boolean;
```

Description

The AcceptOnDblClick property controls wether the tree node can be selected by double click or by single click.

Set AcceptOnDblClick to True, if you think that user should double click the node to select it, or leave it False if you would like to make it selectable by single mouse click.

See also

TreeOptions property.

10.42.2.2AttachedLabel

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and acTreeComboBox components.

Unit

acAttachedLabel

Declaration

```
type
  TacAttachedLabelPosition = (lpAbove, lpBelow, lpLeft, lpRight);
  TacAttachedLabelSpacingKind = (skNearPoint, skFarPoint);
  TacAttachedLabel = class
    property Caption: TCaption;
    property Color: TColor default clBtnFace;
    property Cursor: TCursor default crDefault;
    property Enabled: Boolean;
    property FocusOnClick: Boolean default True;
    property Font: TFont;
    property Hint: String;
    property ParentColor: Boolean default True;
    property ParentFont: Boolean default True;
    property ParentShowHint: Boolean default True;
    property PopupMenu: TPopupMenu;
    property Position: TacAttachedLabelPosition default lpLeft;
    property ShowAccelChar: Boolean default True;
    property ShowHint: Boolean default False;
    property Spacing: Integer default 4;
    property SpacingKind: TacAttachedLabelSpacingKind default
skNearPoint;
    property Transparent: Boolean default False;
    property Visible: Boolean;
    property WordWrap: Boolean default False;
  end;
```

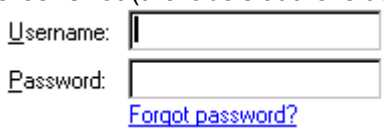
Description

The AttachedLabel structure lets you to operate with the label attached to the some side of control (at the left or right side, above or below). You can specify the behaviour of label, its position near the control, and accelerator key for the control (character in caption, after ampersand (&)).

The label always associated with the control. When user press the accelerator key (Alt + Character, specified after ampersand (&)), the input focus will be imediately switched to the labeled control. The input focus also will be moved to the control if user clicks the label with mouse.

© 1998-2006, UtilMind Solutions®

Screenshot (the labels at the left side is attached to the edit control):



See also

OnLabelClick and OnLabelDblClick events.

10.42.2.2.1 Caption

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property Caption: TCaption;
```

Description

The Caption property specified a text string which appers on label attached to control (attached label).

To underline a character in a Caption that labels a component, include an ampersand (&) before the character. This type of character is called an accelerator character. The user can then select the component by pressing Alt while typing the underlined character. To display an ampersand character in the caption, use two ampersands (&&).

The label always associated with the control. When user press the accelerator key (Alt + Character, specified after ampersand (&)), the input focus will be imediately switched to the labeled control.

Screenshot (the labels at the left side is attached to the edit control):

See also

ShowAccelChar, Color, Font, Transparent, Visible and WordWrap properties.

10.42.2.2.2 Color

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property Color: TColor;
```

Description

The Color property specifies the background color for label attached to control. Use Color to read or change the background color of the control.

If a control's ParentColor property is true, then changing the Color property of the control's parent automatically changes the Color property of the control. When the value of the Color property is

© 1998-2006, UtilMind Solutions®

changed, the control's ParentColor property is automatically set to False.

If you don't want to show background — set Transparent property to True.

See also
ParentColor and Transparent properties.

10.42.2.2.3 Cursor

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property Cursor: TCursor;
```

Description
The Cursor property specifies the image used to represent the mouse pointer when it passes into the region covered by this label attached to control.

Change the value of Cursor to provide feedback to the user when the mouse pointer enters the control. The value of Cursor is the index of the cursor in the list of cursors maintained by the global variable, Screen. In addition to the built-in cursors provided by TScreen, applications can add custom cursors to the list.

See also
Additional Cursors

10.42.2.2.4 Enabled

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property Enabled: Boolean;
```

Description
The Enabled property controls whether the control responds to mouse, keyboard, and timer events.

Use Enabled to change the availability of the control to the user. To disable a control, set Enabled to false. Disabled controls appear dimmed. If Enabled is false, the control ignores mouse, keyboard, and timer events.

To re-enable a control, set Enabled to true. The control is no longer dimmed, and the user can use the control.

See also
Caption and Color properties.

10.42.2.2.5 FocusOnClick

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

© 1998-2006, UtilMind Solutions®

Declaration

```
property FocusOnClick: Boolean; // True by default
```

Description

The FocusOnClick property controls whether the control, associated with the attached label, should accept focus when user clicks the label with mouse.

Set FocusOnClick to True if you want to switch focus to the control when user clicks the label, or set it to False otherwise.

Note

Additionally you can use OnLabelClick and OnLabelDblClick events of the control.

10.42.2.2.6 Font

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property Font: TFont;
```

Description

The Font property controls the attributes of text written on the attached label of control.

To change to a new font, specify a new TFont object. To modify a font, change the value of the Charset, Color, Height, Name, Pitch, Size, or Style of the TFont object.

See also

ParentFont property.

10.42.2.2.7 Hint

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property Hint: String;
```

Description

The Hint property contains the text string that can appear when the user moves the mouse over the label attached control.

If you are using the FormHelp component to show the context-sensitive help taken from secondary part of hint AND the Hint for attached label is not specified, the FormHelp will use the hint of main control.

See also

ShowHint and ParentShowHint properties;
acFormHelp and acFormHints components.

10.42.2.2.8 ParentColor

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox

© 1998-2006, UtilMind Solutions®

and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property ParentColor: Boolean;
```

Description

The ParentColor determines where an attached label looks for its color information.

To have a control use the same color as its parent control, set ParentColor to true If ParentColor is false the control uses its own Color property.

Set ParentColor to true for all controls in order to ensure that all the controls on a form have a uniform appearance. For example, if ParentColor is true for all controls in a form, changing the background color of the form to gray causes all the controls on the form to also have a gray background.

When the value of a control's Color property changes, ParentColor becomes false automatically.

See also

Color and Transparent properties.

10.42.2.2.9 ParentFont

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property ParentFont: Boolean;
```

Description

The ParentFont property determines where a control looks for its font information.

To have a control use the same font as its parent control, set ParentFont to true If ParentFont is false the control uses its own Font property.

Set ParentFont to true for all controls in order to ensure that all the controls on a form have a uniform appearance. For example, if ParentFont is true for all controls in a form, changing the form's Font property to 12-point Courier causes all controls on the form to use that font.

When the value of a control's Font property changes, ParentFont becomes false automatically.

When ParentFont is true for a form, the form uses the default font.

See also

Font property.

10.42.2.2.10 ParentShowHint

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property ParentShowHint: Boolean;
```

Description

The ParentShowHint property determines where an attached label looks to find out if its Help Hint

© 1998-2006, UtilMind Solutions®

should be shown.

Use ParentShowHint to ensure that all the controls on a form either uniformly show their Help Hints or uniformly do not show them.

If ParentShowHint is true, the control uses the ShowHint property value of its parent. If ParentShowHint is false, the control uses the value of its own ShowHint property.

To provide Help Hints for only selected controls on a form, set the ShowHint property for those controls that should have Help Hints to true, and ParentShowHint becomes false automatically.

Note

Enable or disable all Help Hints for the entire application using the ShowHint property of the application object.

See also

ShowHint and Hint properties;
acFormHelp and acFormHints components.

10.42.2.2.11 PopupMenu

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property PopupMenu: TPopupMenu;
```

Description

The PopupMenu property Identifies the pop-up menu associated with the label attached to control.

Assign a value to PopupMenu to make a pop-up menu appear when the user selects the label and clicks the right mouse button. If the TPopupMenu's AutoPopup property is True, the pop-up menu appears automatically. If the menu's AutoPopup property is False, display the menu with a call to its Popup method from the control's OnContextPopup event handler.

10.42.2.2.12 Position

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
type
  TacAttachedLabelPosition = (lpAbove, lpBelow, lpLeft, lpRight);

property Position: TacAttachedLabelPosition;
```

Description

The Position property of the attached label specifies the layout of the label near the control.

You can make the label visible above, below, at the left or the right side of control.

See also

Spacing and SpacingKind properties.

© 1998-2006, UtilMind Solutions®

10.42.2.2.13 ShowAccelChar

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property ShowAccelChar: Boolean;
```

Description

The ShowAccelChar property determines how an ampersand in the label text is displayed.

Set ShowAccelChar to True to allow the label to display an underlined accelerator key value. When ShowAccelChar is True, any character preceded by an ampersand (&) appears underlined. If the FocusControl property is set, the windowed control specified by the FocusControl property receives input focus when the user types that underlined character. To display an ampersand when ShowAccelChar is True, use two ampersands (&&) to stand for the single ampersand that is displayed.

Set ShowAccelChar to False to display the label text with all ampersands appearing as ampersands.

Example

This example uses two labels on a form. The first label has a caption with an accelerator character in it. The second label also includes an ampersand, but it does not appear as an accelerator character.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Label1.ShowAccelChar := True;
  Label1.Caption := 'An &Underlined character appears here';
  Label2.ShowAccelChar := False;
  Label2.Caption := 'An ampersand (&) appears here';
end;
```

See also

Caption property.

10.42.2.2.14 ShowHint

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property ShowHint: Boolean;
```

Description

The ShowHint property determines whether the label attached to control displays a Help Hint when the mouse pointer rests momentarily on the label.

The Help Hint is the value of the Hint property, which is displayed in a box just beneath the control. Use ShowHint to determine whether a Help Hint appears for the control.

To enable Help Hint for a particular control, the application ShowHint property must be true and either the control's own ShowHint property must be true, or the control's ParentShowHint property

© 1998-2006, UtilMind Solutions®

must be true and its parent's ShowHint property must be true.

For example, imagine a check box within a group box. If the ShowHint property of the group box is true and the ParentShowHint property of the check box is true, but the ShowHint property of the check box is false, the check box still displays its Help Hint.

Changing the ShowHint value automatically sets the ParentShowHint property to false.

See also
Hint property.

10.42.2.2.15 Spacing

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property Spacing: Integer;
```

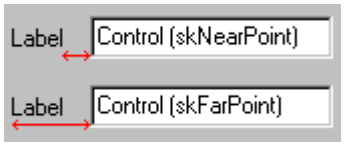
Description
The Spacing property determines the space (in pixels) between the control and attached label.

There is two spacing kinds which have an influence on distance between the control and label (see SpacingKind property).

When SpacingKind is `skNearPoint`, the Spacing property signifies the distance between control and nearest point of attached label.

When SpacingKind is `skFarPoint`, the Spacing property signigies the distance between control and far point of label.

Screenshot



See also
SpacingKind and Position properties.

10.42.2.2.16 SpacingKind

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
type
  TacAttachedLabelSpacingKind = (skNearPoint, skFarPoint);

property SpacingKind: TacAttachedLabelSpacingKind;
```

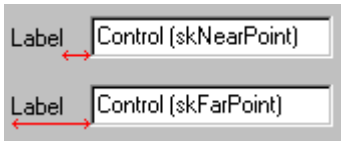
Description
The SpacingKind property specifies how to measure the spacing between the control and attached label.

© 1998-2006, UtilMind Solutions®

There is two possible values for SpacingKind property:

Value	Meaning
skNearPoint	the Spacing property signifies the distance between control and nearest point of attached label;
skFarPoint	the Spacing property signigies the distance between control and far point of label.

Screenshot



See also

Spacing and Position properties.

10.42.2.2.17 Transparent

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property Transparent: Boolean;
```

Description

The Transparent property specifies whether controls that sit below the label on a form can be seen through the label.

Set Transparent to True to prevent the label from obscuring other controls on the form. For example, if the label is used to add text to a graphic, set Transparent to True so that the label does not stand out as a separate object.

Writing text so that the background is transparent is slower than writing text when Transparent is False. If the label is not obscuring a complicated image, performance can be improved by setting the background color of the label to match the object beneath it and setting Transparent to False.

See also

Color property.

10.42.2.2.18 Visible

Applies to

acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration

```
property Visible: Boolean;
```

Description

The Visible property determines whether the label attached to the control appears on screen.

Use the Visible property to control the visibility of the control at runtime. If Visible is true, the control appears. If Visible is false, the control is not visible.

© 1998-2006, UtilMind Solutions®

10.42.2.2.19 WordWrap

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and TacTreeComboBox components as subproperty of **AttachedLabel** structure.

Declaration
```
property WordWrap: Boolean;
```

Description
The WordWrap property specifies whether the label text wraps when it is too long for the width of the label.

Set WordWrap to True to allow the label to display multiple line of text. When WordWrap is True, text that is too wide for the label control wraps at the right margin and continues in additional lines.

Set WordWrap to False to limit the label to a single line. When WordWrap is False, text that is too wide for the label appears truncated.

See also
Caption property.

10.42.2.3AutoComplete

Applies to
acTreeComboBox component.

Declaration
```
property AutoComplete: Boolean;
```

Description
The AutoComplete property specifies whether the combo box automatically completes words that user typed by selecting the first item that begins with the currently typed string.

When AutoComplete is True, the combo box responds to user keystrokes by searching the Items property array for the first item that matches the string entered so far. If it finds an item that begins with the prefix typed so far, the combo box selects that item, "completing" the user's typing. If the user continues to type, selection may move to another, later, item, as the currently typed prefix no longer matches the initial AutoComplete selection. If the user types a string that is not the prefix of a string in the combo box, the string is taken as a unique entry and none of the items in the list are selected.

When AutoComplete is False, this feature is disabled. Strings that the user types are taken literally, with no attempt to match them to a string in the combo box. Setting AutoComplete to False is necessary if you want to allow the user to enter a value that is not in the Items list but that is a prefix on one of the items.

See also
AutoCompleteDelay, AutoCompleteIgnoreCase, Items, ShowEditor and SelectedNode properties.

10.42.2.4AutoCompleteDelay

Applies to
acTreeComboBox component.

Declaration

© 1998-2006, UtilMind Solutions®

```
property AutoCompleteDelay: Integer; // 100 milliseconds by default
```

Description

The AutoCompleteDelay property speicifes the delay, in milliseconds, which should pass between keypress and attempt to auto-complete the input.

By default AutoCompleteDelay = 100 milliseconds, that means that when user typed the character and delayed 1/10 of second before typing the next character, the combo box will try to auto-complete the input searching the item with the text which begins with the currently typed string.

See also

AutoComplete, AutoCompleteIgnoreCase, Items, ShowEditor and SelectedNode properties.

10.42.2.5AutoCompleteIgnoreCase

Applies to

acTreeComboBox component.

Declaration

```
property AutoCompleteIgnoreCase: Boolean; // True by default
```

Description

The AutoCompleteIgnoreCase property specifies whether the case of typed characters does not matter, and whether you would like to auto-complete the input disregarding the case of characters.

For example, the drop-down list contains the line "Apple". If the AutoCompleteIgnoreCase is True and user type the "a", the compobox will try to put missing part of line "pple", disregarding of the charcase of the line in the history list. In case if the AutoComplet IgnoreCase is False, the combo box will try to find the line with exactly same case of characters.

See also

AutoComplete, AutoCompleteDelay, Items, ShowEditor and SelectedNode properties.

10.42.2.6Button

Applies to

acEdit, acNumberEdit and acTreeComboBox components.

Declaration

```
type
  TacEditButton = class(TPersistent)
  published
    property Cursor: TCursor;
    property Flat: Boolean; // in Delphi 3 and later !!!
    property Glyph: TBitmap;
    property NumGlyphs: TNumGlyphs; // 1..4
    property Hint: String;
    property Kind: TacEditButtonKind;
    property Width: Word;
    property Visible: Boolean; // False by default
  end;
```

Description

The acEdit component (and all its successors, like acNumberEdit or acTreeComboBox) is able to have a custom button at the right edge of edit control. The Button structure intended for specify settings for this button. To make button visible on edit control — set Button.Visible to True.

© 1998-2006, UtilMind Solutions®

You can specify the glyph image and number of images, hint, cursor and width for button.

The button which at the right of edit control is just usual TSpeedButton component. You can even gain direct access to all button's properties using hidden **EditButton** property.
For example, to specify caption for this button you may use following code:

```
acTreeComboBox1.EditButton.Caption := 'Test';
```

See also
OnButtonClick event.

10.42.2.6.1 Cursor

Applies to
acEdit, acNumberEdit and acTreeComboBox components as subproperty of Button structure.

Declaration

```
property Cursor: TCursor;
```

Description
The Cursor property specifies the image for mouse pointer when it passes over the custom button at the right side of edit control.

See also
CursorBorder property.

10.42.2.6.2 Flat

Applies to
acEdit, acNumberEdit and acTreeComboBox components as subproperty of Button structure.

Declaration

```
property Flat: Boolean;
```

Description
The Flat property determines whether the button has a 3D border that provides a raised or lowered look.

Set Flat to True to remove the raised border when the button is unselected and the lowered border when the button is clicked or selected. When Flat is True, use separate bitmaps for the different button states to provide visual feedback to the user about the button state.

Remarks
This property available in Delphi 3 and later.

10.42.2.6.3 Glyph

Applies to
acEdit, acNumberEdit and acTreeComboBox components as subproperty of Button structure.

Declaration

```
property Glyph: TBitmap;
```

Description
The Glyph property specifies the bitmap that appears on the custom button at the right side of edit control.

Set Glyph to a bitmap object that contains the image that should appear on the face of the button.

© 1998-2006, UtilMind Solutions®

Bring up the Open dialog box from the Object Inspector to choose a bitmap file (with a .BMP extension), or specify a bitmap file at runtime.

Glyph can provide up to four images within a single bitmap. All images must be the same size and next to each other in a horizontal row. Button displays one of these images depending on the state of the button.

Image position	Buttons state	Description
First	Up	This image appears when the button is unselected. If no other images exist in the bitmap, this image is used for all states.
Second	Disabled	This image usually appears dimmed to indicate that the button can't be selected.
Third	Clicked	This image appears when the button is clicked. The Up image reappears when the user releases the mouse button.
Fourth	Down	This image appears when the button stays down indicating that it remains selected.

If only one image is present, Button attempts to represent the other states by altering the image slightly for each state, although the Down state is always the same as the Up state.

If the bitmap contains multiple images, specify the number of images in the bitmap with the NumGlyphs property.

Note
The lower left hand pixel of the bitmap is reserved for the "transparent" color. Any pixel in the bitmap which matches that lower left hand pixel will be transparent.

See also
NumGlyphs property.

10.42.2.6.4 NumGlyphs

Applies to
acEdit, acNumberEdit and acTreeComboBox components as subproperty of Button structure.

Declaration
```
property NumGlyphs: TNumGlyphs; // 1..4
```

Description
The NumGlyphs is the number of images specified in the Glyph property.

Set NumGlyphs to the number of images provided by the bitmap assigned to the Glyph property. All images must be the same size and next to each other in a row. The Glyph property can provide up to four images.

See also
Glyph property.

10.42.2.6.5 Hint

Applies to
acEdit, acNumberEdit and acTreeComboBox components as subproperty of Button structure.

Declaration
```
property Hint: String;
```

© 1998-2006, UtilMind Solutions®

Description
The Hint property contains the text string that can appear when the user moves the mouse pointer over the button at right side of edit control.

10.42.2.6.6 Kind

Applies to
acEdit, acNumberEdit and acTreeComboBox components as subproperty of Button structure.

Declaration
```
type
  TacEditButtonKind = (bkCustom, bkEllipsis, bkDropDown, bkFileOpen,
bkFileSave, bkBrowse, bkCalc);

property Kind: TacEditButtonKind;
```

Description
The Kind property determines the kind of bitmap button.

You can use custom image (Kind = bkCustom) and specify your sign to Glyph property, or use one of following images ( ), just switching the Kind property.

See also
Glyph and NumGlyphs properties.

10.42.2.6.7 Width

Applies to
acEdit, acNumberEdit and acTreeComboBox components as subproperty of Button structure.

Declaration
```
property Width: Integer;
```

Description
The Width property controls the horizontal size of the button in pixels.

The button's height controlled by height of acTreeComboBox control.

10.42.2.6.8 Visible

Applies to
acEdit, acNumberEdit and acTreeComboBox components as subproperty of Button structure.

Declaration
```
property Visible: Boolean;
```

Description
The Visible property determines whether the button appears at the right side of acTreeComboBox control.

Use the Visible property to control the visibility of the button at runtime. If Visible is True, the button appears. If Visible is False, the button is not visible.

10.42.2.7ColorDisabled

Applies to
acEdit, acNumberEdit, acIPEdit and acTreeComboBox components.

© 1998-2006, UtilMind Solutions®

Declaration

```
property ColorDisabled: TColor; // clBtnFace by default
```

Description

The ColorDisabled property for acTreeComboBox component is the background color of the edit control at disabled state (when Enabled is False). Use the ColorDisabled property to specify color for disabled edit control.

See also

Color property.

10.42.2.8 CursorBorder

Applies to

acEdit, acNumberEdit and acTreeComboBox components.

Declaration

```
property CursorBorder: TCursor;
```

Description

The CursorBorder property specifies the image used to represent the mouse pointer when it passes over the non-client area of edit control.

See also

acAppCursors component and Additional cursors in the IDE.

10.42.2.9 DropDownCount

Applies to

acTreeComboBox component.

Declaration

```
property DropDownCount: Integer;
```

Description

The DropDownCount property specifies the number of items (tree nodes) which can be visible in the drop down window simultaneously. You can increase this value if you wish to let user to see more tree nodes when the list is dropped down.

See also

DropDownWidth property.

10.42.2.10 DropDownWidth

Applies to

acTreeComboBox component.

Declaration

```
property DropDownWidth: Integer;
```

Description

The DropDownWidth controls the width of the drop down window, when can be visible when user clicks on the TreeComboBox control or its button at the right side.

By default the width of drop down window is the same as the width of control (when the DropDownWidth equal to Width and you resize the control at design time, you will see that

© 1998-2006, UtilMind Solutions®

DropDownWidth will be changed accordingly to Width of control). But of course, you can modify this value, if you wish to make the width of popup window different than width of contol.

See also
DropDownCount property.

EmptyItemImageIndex
10.42.2.11

Applies to
acTreeComboBox component.

Declaration
```
property EmptyItemImageIndex: Integer;
```

Description
The EmptyItemImageIndex property used to specify the index of image from image list. The image which you selected will appear in the combo box if no tree node is selected.

See also
EmptyItemText and Images properties.

EmptyItemText
10.42.2.12

Applies to
acTreeComboBox component.

Declaration
```
property EmptyItemText: String;
```

Description
The EmptyItemText property used to specify the text which appears in the combo box if no tree node is selected (for example, you can specify "Please select something" text to this property if you don't want to show the blank box).

See also
EmptyItemImageIndex property.

Images
10.42.2.13

Applies to
acTreeComboBox component.

Declaration
```
property Images: TImageList;
```

Description
The Images property determines which image list is associated with the TreeView embedded to the drop-down window of acTreeComboBox.

Use Images to provide a customized list of bitmaps that can be displayed to the left of a node's label. Individual nodes specify the image from this list that should appear by setting their ImageIndex property (use Items property to specify the ImageIndex'es of the tree nodes at design-time).

© 1998-2006, UtilMind Solutions®

See also
StateImages properties.

ItemIndex
10.42.2.14

Applies to
acTreeComboBox component.

Declaration
```
property ItemIndex: Integer;
```

Description
The ItemIndex property determines the absolute index of selected node. You can read and specify this value both at run- and design-time to select the node.

See also
SelectedNode property;
OnCanSelectNode event.

Items
10.42.2.15

Applies to
acTreeComboBox component.

Declaration
```
property Items: TTreeNodes;
```

Description
The Items property lists the individual nodes that appear in the TreeView control, embedded to the drop-down window of acTreeComboBox.

Individual nodes in a tree view are TTreeNode objects. These individual nodes can be accessed by using the Items property along with the item's index into the tree view. For example, to access the second item in the tree view, you could use the following code.

MyTreeNode := acTreeComboBox1.Items[1];

When setting this property at design-time in the Object Inspector the TreeView Items Editor appears. Use the New Item and New SubItem buttons to add items to the tree view. Use the Text property to modify what text is displayed in the label of the item.

At run-time nodes can be added and inserted by using the TTreeNodes methods AddChildFirst, AddChild, AddChildObjectFirst, AddChildObject, AddFirst, Add, AddObjectFirst, AddObject and Insert.

Notes
Accessing tree view items by index can be time-intensive, particularly when the tree view contains many items. For optimal performance, try to design your application so that it has as few dependencies on the tree view's item index as possible;

Alternatively you can access the tree nodes at run-time through TreeView property.

See also
TreeView, TreeOptions and SelectedNode properties.

© 1998-2006, UtilMind Solutions®

SelectedNode
10.42.2.16

Applies to
acTreeComboBox component.

Declaration
```
property SelectedNode: TTreeNode; // read-only!!
```

Description
The SelectedNode is the read-only property which returns the pointer to TTreeNode object, which currently selected in the TreeView control embedded to drop-down window.

Alternatively you can determinate and specify the selected tree node by its absolute index using ItemIndex property.

The SelectedNode can be NIL if none selected in the drop-down list, or user specified custom text which does not match any node in the list (if ShowEditor is True).

See also
ItemIndex, ShowEditor properties;
OnCanSelectNode event.

ShowEditor
10.42.2.17

Applies to
acTreeComboBox component.

Declaration
```
property ShowEditor: Boolean;
```

Description
The ShowEditor property controls whether user should be able to enter of modify the text in the combo-box using keyboard.

Set ShowEditor to True if you would like to let user to modify the text with keyboard, otherwise, if user must select the tree node from drop-down list, set ShowEditor to False.

See also
AutoComplete, SelectedNode and TreeOptions properties.

ShowTreePathInEdit
10.42.2.18

Applies to
acTreeComboBox component.

Declaration
```
property ShowTreePathInEdit: Boolean;
```

Description
The ShowTreePathInEdit property determines whether the "edit box" should display the full path of the node.

For example, we have the following structure:

© 1998-2006, UtilMind Solutions®

```
Test
    Test 1
    Test 2
        Test 1 (selected)
        Test 2
    Test 3
```

Then, in case if ShowTreePathInEdit = True, we will have "Test\Test 2\Test 1" shown in the edit box of the combobox.

Note

This property will take effect only in case if ShowEditor property is False. You can't display the path when the combobox is editable.

See also

TreePathSeparator and ShowEditor properties.

StateImages

10.42.2.19

Applies to

acTreeComboBox component.

Declaration

```
property StateImages: TImageList;
```

Description

The StateImages property determines which image list to use for state images.

Use StateImages to provide a set of bitmaps that reflect the state of tree view nodes. The state image appears as an additional image to the left of the item's icon.

See also

Images, TreeOptions and TreeView properties.

TreeOptions

10.42.2.20

Applies to

acTreeComboBox component.

Declaration

```
type
  TacTreeViewOptions = class
  published
    property AutoExpand: Boolean default False;
    property ExpandTopLevelNodes: Boolean default False;
    property FullExpand: Boolean default False;
    property MouseTrack: Boolean default True;
    property RowSelect: Boolean default False;
    property ShowButtons: Boolean default True;
    property ShowLines: Boolean default True;
    property ShowRoot: Boolean default True;
    property Tooltips: Boolean default True;
  end;

property TreeOptions: TacTreeViewOptions;
```

© 1998-2006, UtilMind Solutions®

Description

The TreeOptions structure used to specify some behaviours of the TreeView embedded to drop-down window of acTreeComboBox control.

There are possible properties of embedded TreeView control:

Property	Purpose
AutoExpand	Specifies whether the nodes in the tree view automatically expand and collapse depending on the selection;
ExpandTopLevelNodes	Specifies whether the TreeComboBox should automatically expand all the top-level nodes (nodes without parent), upon DropDown (alternativaly you can just expand the nodes in OnDropDown event handler).
FullExpand	Specifies whether the component should automatically expand all the nodes and their sub-nodes upon DropDown (alternativaly you can just expand nodes in OnDropDown event handler);
MouseTrack	Specifies whether the selection should always hovers the tree node under mouse pointer, even when user move mouse over tree view without clicking;
RowSelect	Specifies whether the entire row of the selected item is highlighted;
ShowButtons	Specifies whether to display plus (+) and minus (-) buttons to the left side of each parent item;
ShowLines	Specifies whether to display the lines that link child nodes to their corresponding parent nodes;
ShowRoot	Specifies whether lines connecting top-level nodes are displayed;
Tooltips	Specifies whether the items in the tree view have tooltips.

See also

AcceptOnDblClick, ShowEditor, SelectedNode and TreeView properties;
ExpandTopLevelNodes method of acTreeView component.

TreePathSeparator
10.42.2.21

Applies to

acTreeComboBox component.

Declaration

```
property TreePathSeparator: Char; // '\' by default
```

Description

The TreePathSeparator specifies the separator character for the node path, in case if ShowTreePathInEdit = True.

See also

ShowTreePathInEdit and ShowEditor properties.

© 1998-2006, UtilMind Solutions®

TreeView
10.42.2.22

Applies to
acTreeComboBox component.

Declaration
```
property TreeView: TacPopupTreeView; // run-time and read-only!
```

Description
The TreeView is the *read-only* property which can be used *at run-time* to access the properties and methods of the TreeView control embedded to the drop-down window of acTreeComboBox.

Alternatively you can just specify some behaviours of TreeView using TreeOptions property.

Example
```
procedure TForm1.acTreeComboBox1DropDown(Sender: TObject);
begin
  acTreeComboBox1.TreeView.FullExpand;
end;
```

See also
TreeOptions and SelectedNode properties;
ExportToTreeNodes, ImportFromTreeNodes and ImportFromTreeView methods;
OnDropDown, OnAfterDropDown and OnCloseUp events.

10.42.3 Methods

10.42.3.1AddPath

Applies to
acTreeComboBox and acTreeView components.

Declaration
```
function AddPath(Node: TTreeNode; const Path: String; const
PathSeparator: String = '\'): TTreeNode;
```

Description
The AddPath method adds or inserts to the TreeView the *string* or nested *group of strings* (pathes) specified by *Path* parameter and separated by string specifed in *PathSeparator* parameter.

The declaration can be
```
AddPath(nil, 'D:\Test\Test');
```

So, if you call method above, the component will add following structure to the root item of TreeComboBox:
```
D:
  Test
    Test
```

Return value is the last added child node.

See also
ShowTreePathInEdit, TreePathSeparator and TreeView properties.

© 1998-2006, UtilMind Solutions®

10.42.3.2ExportToTreeNodes

Applies to
acTreeComboBox and acTreeView components.

Declaration
```
procedure ExportToTreeNodes(TargetTreeNodes: TTreeNodes;
  SourceRootNode: TTreeNode = nil; TargetRootNode: TTreeNode = nil);
```

Description
The ExportToTreeNodes method used to copy the tree nodes from current TreeCombBox to another container TTreeNodes object (i.e: standard TreeView or another TreeComboBox).

Example *(demonstrates how to fill the TreeView with nodes from TreeComboBox)*
acTreeComboBox1.ExportToTreeNodes(acTreeView1.Items);

See also
ImportFromTreeNodes and ImportFromTreeView methods.

10.42.3.3FindAbsoluteIndexByNode

Applies to
acTreeComboBox and acTreeView components.

Declaration
```
function FindAbsoluteIndexByNode(Node: TTreeNode): Integer;
```

Description
The FindAbsoluteIndexByNode method retreives the absolute index from node specified in the *Node* parameter.

To get the node by abolute index you can use following code:

```
Node := acTreeComboBox1.Items[AbsoluteIndex];
```

See also
FindNode method.

10.42.3.4FindNode

Applies to
acTreeComboBox and acTreeView components.

Declaration
```
function FindNode(const Text: String; IgnoreCase: Boolean = False;
  WholeText: Boolean = True): TTreeNode;
```

Description
The FindNode method search the tree node with the text specified in *Text* parameter.

The *IgnoreCase* parameter specified whether the search should be case sensitive.

The *WholeText* parameter specified whether it should find the node where the length of caption is the same as length of Text parameter (if *WholeText* is False, the method will try to find the node which caption begins with the text specified in *Text* parameter).

© 1998-2006, UtilMind Solutions®

Function returns the pointer to tree node if it successfully found, or NIL (NULL) otherwise (if the node with specified Text not exists in the tree view).

See also
FindAbsoluteIndexByNode methods.

10.42.3.5ImportFromTreeNodes

Applies to
acTreeComboBox and acTreeView components.

Declaration
```
procedure ImportFromTreeNodes(SourceTreeNodes: TTreeNodes;
  SourceRootNode: TTreeNode = nil; TargetRootNode: TTreeNode = nil);
```

Description
The ImportFromTreeNodes method used to copy the tree nodes from some container TTreeNodes object (i.e: standard TreeView or another TreeComboBox) to *this* TreeComboBox control.

Example *(demonstrates how to fill the acTreeComboBox control with nodes from some TreeView)*
```
acTreeComboBox1.ImportFromTreeNodes(acTreeView1.Items);
```

See also
ExportToTreeNodes and ImportFromTreeView methods.

10.42.3.6ImportFromTreeView

Applies to
acTreeComboBox and acTreeView components.

Declaration
```
procedure ImportFromTreeView(SourceTreeView: TCustomTreeView;
  SourceRootNode: TTreeNode = nil; TargetRootNode: TTreeNode = nil);
```

Description
The ImportFromTreeView method used to copy the tree nodes from some TreeView control to *this* TreeComboBox control.

Example *(demonstrates how to fill the acTreeComboBox control with nodes from some TreeView)*
```
acTreeComboBox1.ImportFromTreeView(acTreeView1);
```

See also
ExportToTreeNodes and ImportFromTreeNodes methods.

10.42.3.7SelectNodeByText

Applies to
acTreeComboBox component.

Declaration
```
function SelectNodeByText(const Text: String; IgnoreCase: Boolean =
False): TTreeNode;
```

Description
The SelectNodeByText method looking for the node in drop-down list, with the same text as specified in Text property. If it successfully found the node with specified text, it selects it.

© 1998-2006, UtilMind Solutions®

The *IgnoreCase* parameter specified whether the search should be case sensitive.

Function returns the pointer to selected node if succeed, or NIL (NULL) otherwise.

See also
ShowEditor property.

10.42.3.8SetSelection

Applies to
acEdit, acNumberEdit, acMRUComboBox, acLabeledComboBox and acTreeComboBox components.

Declaration
```
procedure SetSelection(SelStart: Integer = 0; SelEnd: Integer = -1);
```

Description
The SetSelection method used to select a range of characters in an edit control.

SelStart parameter specifies the starting character position of the selection.

SelEnd parameter specifies the ending character position of the selection.

Remarks
If the *SelStart* parameter is 0 and the *SelEnd* parameter is -1, all the text in the edit control is selected. If *SelStart* is -1, any current selection is removed. The caret is placed at the end of the selection indicated by the greater of the two values *SelEnd* and *SelStart*.

10.42.4 Events

10.42.4.1OnAfterDropDown

Applies to
acTreeComboBox component.

Declaration
```
property OnAfterDropDown: TNotifyEvent;
```

Description
The OnAfterDropDown event occurs at once the drop-down window (wich displays the tree nodes) appears below the combo-box.

Write this event handler if you need to perform some specific actions at once after the drop-down window appears on screen. If you need to do something (for example, specify the width of drop-down window) *before* it appears on screen — write OnDropDown event handler.

See also
OnDropDown and OnCloseUp events;
DropDownWidth property.

10.42.4.2OnButtonClick

Applies to
acEdit, acNumberEdit and acTreeComboBox components.

Declaration
```
property OnButtonClick: TNotifyEvent;
```

© 1998-2006, UtilMind Solutions®

Description
The OnButtonClick event occurs when user clicks the custom button at the right side of edit control.

To make the button appear on edit control — set Button.Visible to True.

See also
Button structure.

10.42.4.3OnButtonLeftMouseDown

Applies to
acEdit, acNumberEdit and acTreeComboBox components.

Declaration
```
property OnButtonLeftMouseDown: TNotifyEvent;
```

Description
The OnButtonLeftMouseDown event occurs when user presses the left mouse button over the button at the right side of edit control.

To make the button appear on edit control — set Button.Visible to True.

See also
Button structure.

10.42.4.4OnCanSelectNode

Applies to
acTreeComboBox component.

Declaration
```
type
  TacTreeComboBoxCanSelectNodeEvent = procedure(Sender: TObject; Node:
TTreeNode; var AllowSelect: Boolean) of object;

property OnCanSelectNode: TacTreeComboBoxCanSelectNodeEvent;
```

Description
The OnCanSelectNode event

See also
SelectedNode and TreeView properties;
OnDropDown, OnAfterDropDown and OnCloseUp events.

10.42.4.5OnCloseUp

Applies to
acTreeComboBox component.

Declaration
```
type
  TacDropDownCloseUpEvent = procedure(Sender: TObject; Accept: Boolean)
of object;

property OnCloseUp: TacDropDownCloseUpEvent;
```

Description

© 1998-2006, UtilMind Solutions®

The OnCloseUp event occurs when the drop-down window is about to disappear from screen.

The *Accept* parameter indicates whether the tree node has been selected by user or the drop-down window has been closed without selection (i.e: user has clicked somewhere else and moved the input focus to another control). When *Accept* is True, user has selected the node, otherwise — just canceled the drop-down window.

See also
OnDropDown, OnAfterDropDown and OnCanSelectNode events.

10.42.4.6OnDropDown

Applies to
acTreeComboBox component.

Declaration
```
property OnDropDown: TNotifyEvent;
```

Description
The OnDropDown event occurs when the user clicks the combo box (or its button at the right side), and the drop-down window with tree nodes is about to appear on screen.

Write the OnDropDown event handler to perform some specific actions (for example, resize the drop-down window) *before* the tree nodes appears on screen.

If you need to take some actions when the drop-down window already on screen — write OnAfterDropDown event handler.

Example (demonstrates how to change the width of drop-down window in OnDropDown event handler)
```
procedure TForm1.acTreeComboBox1DropDown(Sender: TObject);
begin
  acTreeComboBox1.DropDownWidth := Random(200) + 20;
  // also you can expand some nodes of embedded TreeView
  acTreeComboBox1.TreeView.FullExpand;
end;
```

See also
OnAfterDropDown and OnCloseUp events;
DropDownWidth, TreeOptions and TreeView properties.

10.42.4.7OnLabelClick

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and acTreeComboBox components.

Declaration
```
property OnLabelClick: TNotifyEvent;
```

Description
The OnLabelClick event occurs when user clicks the label attached to the control with a mouse.

To make the label appear near the edit control — set AttachedLabel.Visible to True, or to False if you'd like to hide the attached label.

© 1998-2006, UtilMind Solutions®

See also
AttachedLabel structure.

10.42.4.8 OnLabelDblClick

Applies to
acEdit, acNumberEdit, acIPEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox and acTreeComboBox components.

Declaration
```
property OnLabelDblClick: TNotifyEvent;
```

Description
The OnLabelDblClick event occurs when user double clicks the label attached to the control with a mouse.

To make the label appear near the edit control — set AttachedLabel.Visible to True, or to False if you'd like to hide the attached label.

See also
AttachedLabel structure.

10.42.4.9 OnMouseEnter

Applies to
acLabel, acURLLabel, acRichLabel, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox, acTreeComboBox, acHeader, acHeaderPanel, acTreeView and acListView components.

Declaration
```
property OnMouseEnter: TNotifyEvent;
```

Description
The OnMouseEnter occurs when mouse pointer hover the edit control.

See also
OnMouseLeave event.

10.42.4.10 OnMouseLeave

Applies to
acLabel, acURLLabel, acRichLabel, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox, acTreeComboBox, acHeader, acHeaderPanel, acTreeView and acListView components.

Declaration
```
property OnMouseLeave: TNotifyEvent;
```

Description
The OnMouseLeave occurs when mouse pointer leave the edit control.

See also
OnMouseEnter event.

© 1998-2006, UtilMind Solutions®

10.43 acTreeView

10.43.1 TacTreeView

Overview

The acTreeView is the slightly improved standard TreeView control, which allows to mark out some certain node with bold text, color, background, frame around the node, and small picture at the right side (see screenshot below). Though, In general, this control intended for those who still develop the apps using Delphi 2/3 or BCB 1/3. The acTreeView compatible with any 32-bit Delphi/BCB versions, but supports ALL features of TTreeView for Delphi 5/6, plus the possibility to choose custom cursor images for non-client parts of TreeView control — border, horizontal and vertical scroll bars.

new! Added methods that able very qucikly find the node or its absolute index in the TreeView with big number of items, can save the nodes structure to the file or registry.

Additional / improved properties

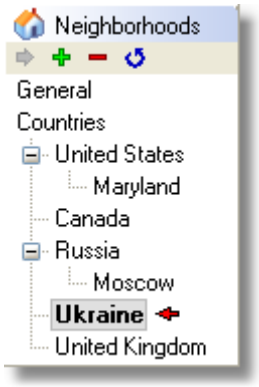
AlwaysExpandRoot	whether all root (top-level) nodes should be always expanded and never collapsed on mouse click;
AlwaysFullExpand	whether ALL nodes should be always expanded and never collapsed on mouse click;
AutoExpand	specifies whether the nodes in the tree view automatically expand and collapse depending on the selection;
BoldItem	set of properties used to mark out some certain node;
Cursors	specifies the cursor images for some non-client parts of TreeView control. You can choose custom cursors for border of TreeView and for horizontal / vertical scrollbars;
HotTrack	specifies whether list items are highlighted when the mouse passes over them;
RowSelect	specifies whether the entire row of the selected item is highlighted;
ToolTips	specifies whether the items in the tree view have tooltips.

Events

OnBoldItemBeforeSwitch	occurs before the marked out "bold" node index become changed;
OnBoldItemSwitched	notifies about successful switching of the "bold" mark from one node to another;
OnMouseEnter	occurs when mouse hovers the control;
OnMouseLeave	occurs when mouse leaves the control;

Screenshot

The acTreeView with selected BoldItem.



© 1998-2006, UtilMind Solutions®

See also
acListView component.

10.43.2 Properties

10.43.2.1AlwaysExpandRoot

Applies to
acTreeView component.

Declaration
```
property AlwaysExpandRoot: Boolean; // False by default
```

Description
The AlwaysExpandRoot property specifies whether the top-level nodes (root nodes) CAN or CAN'T be collapsed.

Set AlwaysExpandRoot to True, and root nodes will be always expanded to display their child nodes.

See also
AlwaysFullExpand and AutoExpand properties.

10.43.2.2AlwaysFullExpand

Applies to
acTreeView component.

Declaration
```
property AlwaysFullExpand: Boolean; // False by default
```

Description
The AlwaysFullExpand property specifies whether ALL nodes which have sub-nodes (child nodes) should be ALWAYS expanded, and whether they CAN or CAN'T be collapsed by mouse click.

Set AlwaysFullExpand to True, and all nodes will be always expanded to display their child nodes.

See also
AlwaysExpandRoot and AutoExpand properties.

10.43.2.3AutoExpand

Applies to
acTreeView component.

Declaration
```
property AutoExpand: Boolean;
```

Description
The AutoExpand property specifies whether the nodes in the tree view automatically expand and collapse depending on the selection.

Set AutoExpand to True to cause the selected item to expand and the unselected items to collapse.

See also
AlwaysExpandRoot and AlwaysFullExpand properties.

© 1998-2006, UtilMind Solutions®

10.43.2.4BoldItem

Applies to

acTreeView component.

Declaration

```
type
  { TacTreeViewBoldItem }
  TacItemIndexIs = (iiAbsoluteIndex, iiImageIndex);
  TacTreeViewBoldItem = class(TacPersistent)
  public
    procedure MakeBoldNodeSelected;

    property BoldNode: TTreeNode;
    property Text: String;
  published
    property ColorBack: TColor default $00E0E0E0;   // light-gray
    property ColorFrame: TColor default $00BBBBBB;  // gray
    property ColorText: TColor default clWindowText;
    property ImageIndex: Integer default -1;
    property ImageList: TImageList;
    property ImageSpacing: Byte default 4;
    property ItemIndex: Integer default -1;
    property ItemIndexIs: TacItemIndexIs default iiAbsoluteIndex;
    property ShowFrame: Boolean default True;
    property SwitchSelectionOnBold: Boolean default False;
    property SwitchToBoldOnDblClick: Boolean default False;
  end;

property BoldItem: TacTreeViewBoldItem;
```

Description

The BoldItem structure contains the set of properties required to mark out some certain node of TreeView with bold text, color, background, frame around the node, and small picture at the right side (see screenshot below).

The index of "bold" node are controlled by ItemIndex property, which can be either: absolute node index in the TreeView, OR the standard image index of node. To specify the way how the control should recognize the ItemIndex — set the value of ItemIndexIs property. It can be `iiAbsoluteIndex` (the ItemIndex is absolute index of node), or `iiImageIndex` (the ItemIndex is the image index, even if the image are not actually displayed (the source of images are not specified)).

Also you can specify the colors which additionally marks out the "bold" node — ColorText, ColorFrame and ColorBack[ground].

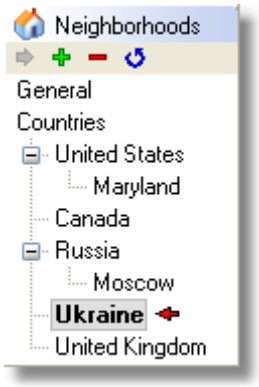
Note

This property set exists only in Delphi/BCB 5 and higher!

Screenshot

The item "Ukraine" is a "bold" node

© 1998-2006, UtilMind Solutions®



See also
OnBoldItemBeforeSwitch and OnBoldItemSwitched events.

10.43.2.4.1 BoldNode

Applies to
acTreeView component as subproperty of BoldItem structure.

Declaration
```
property BoldNode: TTreeNode; // read-only!!
```

Description
The BoldNode is read-only property which returns the pointer to the marked out "bold" node in the TreeView control.

The BoldNode can be NIL (NULL) if there is no "bold" node specified (ItemIndex is -1).

To retreive the text of the "bold" node you can use Text property.

See also
ItemIndex, ItemIndexIs and Text properties;
OnBoldItemBeforeSwitch and OnBoldItemSwitched events.

10.43.2.4.2 ColorBack

Applies to
acTreeView component as subproperty of BoldItem structure.

Declaration
```
property ColorBack: TColor; // $00E0E0E0 by default (light-gray)
```

Description
The ColorBack property specifies the background color under "bold" node.

If you don't want to mark out the background around the bold node — set ColorBack to clWindow (and set ShowFrame to False).

To specify the text color for bold node — use ColorText property.

See also
ColorFrame and ColorText properties.

© 1998-2006, UtilMind Solutions®

10.43.2.4.3 ColorFrame

Applies to
acTreeView component as subproperty of BoldItem structure.

Declaration
```
property ColorFrame: TColor; // $00BBBBBB by default (gray)
```

Description
The ColorFrame property specifies the frame around the background of "bold" node.

If you don't want to display the frame around the bold node — set ShowFrame property to False. If you don't want to mark out the background of bold node at all — set ColorBack to clWindow.

To specify the text color for bold node — use ColorText property.

See also
ColorBack, ColorText and ShowFrame properties.

10.43.2.4.4 ColorText

Applies to
acTreeView component as subproperty of BoldItem structure.

Declaration
```
property ColorText: TColor; // clWindowText by default
```

Description
The ColorText property specifies the text color for the "bold" node.

To specify the background under the bold node — use ColorBack property.

See also
ColorBack and ColorFrame properties.

10.43.2.4.5 ImageIndex

Applies to
acTreeView component as subproperty of BoldItem structure.

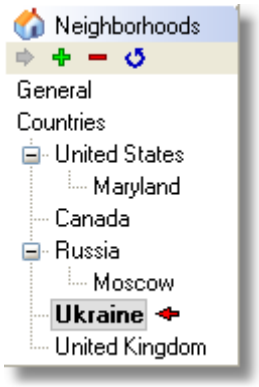
Declaration
```
property ImageIndex: Integer; // -1 by default
```

Description
The ImageIndex property specifies the image *index* in the list of images specified to ImageList property.

The image appears at the right from "bold" node. In case if you don't want to display the image — set ImageIndex to -1.

Screenshot
See the image from right of bold item (Ukraine)

© 1998-2006, UtilMind Solutions®



See also
ImageList and ImageSpacing properties.

10.43.2.4.6 ImageList

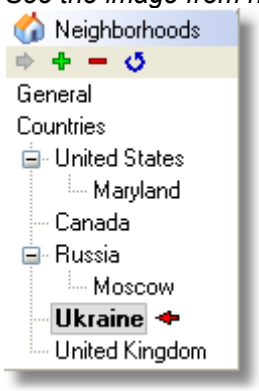
Applies to
acTreeView component as subproperty of BoldItem structure.

Declaration
```
property ImageList: TImageList;
```

Description
The ImageList property specifies the source of images for picture from right of "bold" node.

The index of image should be specified to ImageIndex property

Screenshot
See the image from right of bold item (Ukraine)



See also
ImageIndex and ImageSpacing properties.

10.43.2.4.7 ImageSpacing

Applies to
acTreeView component as subproperty of BoldItem structure.

Declaration
```
property ImageSpacing: Byte; // 4 by default
```

© 1998-2006, UtilMind Solutions®

Description
The ImageSpacing property specifies the space (in pixels) between the text and the image at the right from node.

If you want to display custom image near the bold node — set the source of images to ImageList property and specify the image index to ImageIndex property.

See also
ImageIndex and ImageList properties.

10.43.2.4.8 ItemIndex

Applies to
acTreeView component as subproperty of BoldItem structure.

Declaration
```
property ItemIndex: Integer; // -1 by default
```

Description
The ItemIndex specifies the index of node which should be marked out with bold text, colors, and image at the right side (if specified).

The "bold" node can be specified by either: *absolute index* of node within TreeView control, or by *image index* of the node (Node: even in this case, the standard image at the left from node may be not displayed, if ImageList for entire TreeView are not specified). To choose the way of specifying the "bold" node, by absolute index or image index — set it in the ItemIndexIs property.

The bold node can be switched programmatically by changing the ItemIndex value, or automatically on double click. If you want to allow switching it on double click — set SwitchToBoldOnDblClick to True.

If you don't want to mark out any node — set ItemIndex to -1 (clear the mark).

See also
ItemIndexIs, SwitchSelectionOnBold, SwitchToBoldOnDblClick and Text properties; OnBoldItemBeforeSwitch and OnBoldItemSwitched events.

10.43.2.4.9 ItemIndexIs

Applies to
acTreeView component as subproperty of BoldItem structure.

Declaration
```
type
  TacItemIndexIs = (iiAbsoluteIndex, iiImageIndex);

property ItemIndexIs: TTreeNode; // iiAbsoluteIndex by default
```

Description
The ItemIndexIs property specifies the meaning for the ItemIndex property value.

In case if you want to choose the "bold" node by its absolute index within the TreeView control — set ItemIndexIs to `iiAbsoluteIndex`.

In case if "bold" node should be selected by its standard ImageIndex — set ItemIndexIs to `iiImageIndex`.

© 1998-2006, UtilMind Solutions®

See also
ItemIndex property;
OnBoldItemBeforeSwitch and OnBoldItemSwitched events.

10.43.2.4.10 ShowFrame

Applies to
acTreeView component as subproperty of BoldItem structure.

Declaration
```
property ShowFrame: Boolean; // True by default
```

Description
The ShowFrame property specifies whether the "bold" node should display the frame around it (with color specified by ColorFrame property).

Set ShowFrame to False if you don't want to display the frame, or even set ColorBack to clWindow if you don't want to mark out the background under the bold node at all.

See also
ColorFrame and ColorBack properties.

10.43.2.4.11 SwitchSelectionOnBold

Applies to
acTreeView component as subproperty of BoldItem structure.

Declaration
```
property SwitchSelectionOnBold: Boolean; // False by default
```

Description
The SwitchSelectionOnBold property determines whether the selection and focus should be switched to the "bold" node, when the ItemIndex changes.

Set SwitchSelectionOnBold to True to automatically make the "bold" node selected when the "bold" mark switches from one node to other.

See also
ItemIndex and ItemIndexIs properties;
OnBoldItemBeforeSwitch and OnBoldItemSwitched events.

10.43.2.4.12 SwitchToBoldOnDblClick

Applies to
acTreeView component as subproperty of BoldItem structure.

Declaration
```
property SwitchToBoldOnDblClick: Boolean; // False by default
```

Description
The SwitchToBoldOnDblClick property determines whether "bold" node index should be automatically switched to node that was double clicked.

Set SwitchToBoldOnDblClick to True if you wish to make the node marked out as "bold" when user double clicks it with mouse.

If you need to be notified when the "bold" node index are switched — use

© 1998-2006, UtilMind Solutions®

OnBoldItemBeforeSwitch and OnBoldItemSwitched events.

See also
ItemIndex and ItemIndexIs properties;
OnBoldItemBeforeSwitch and OnBoldItemSwitched events.

10.43.2.4.13 Text

Applies to
acTreeView component as subproperty of BoldItem structure.

Declaration
```
property Text: String; // read-only!!
```

Description
The Text is the read-only property of BoldItem structure, used to determinate the text of marked out "bold" node.

To get the pointer to the "bold" node itself and get access to all its properties — use BoldNode property.

To get the absolute index or image index of the "bold" node — use ItemIndex property.

The Text property can return NIL (NULL) in case if there is no marked "bold" node (ItemIndex is -1).

See also
BoldNode and ItemIndex properties;
OnBoldItemBeforeSwitch and OnBoldItemSwitched events.

10.43.2.5Cursors

Applies to
acTreeView and acListView components.

Declaration
```
type
  TacTreeViewCursors = class
  published
    property Border: TCursor;
    property ScrollHorz: TCursor;
    property ScrollVert: TCursor;
  end;

property Cursors: TacTreeViewCursors;
```

Description
Use Cursors structure to specify custom cursor images for some non-client parts of acTreeView control. You can specify cursor for border of control, horizontal scrollbar and vertical scrollbar.

See also
acAppCursors component and Additional cursors in the IDE.

10.43.2.5.1 Border

Applies to
acTreeView and acListView components as subproperty of Cursors structure.

© 1998-2006, UtilMind Solutions®

Declaration
```
property Border: TCursor;
```

Description
The Border property specifies the image for mouse pointer, when it passes over border of acTreeView control.

See also
ScrollHorz and ScrollVert properties.

10.43.2.5.2 ScrollHorz

Applies to
acTreeView and acListView components as subproperty of Cursors structure.

Declaration
```
property ScrollHorz: TCursor;
```

Description
The ScrollHorz property specifies the image for mouse pointer, when it passes over horizontal scrollbar of acTreeView control.

See also
ScrollVert and Border properties.

10.43.2.5.3 ScrollVert

Applies to
acTreeView and acListView components as subproperty of Cursors structure.

Declaration
```
property ScrollVert: TCursor;
```

Description
The ScrollVert property specifies the image for mouse pointer, when it passes over vertical scrollbar of acTreeView control.

See also
ScrollHorz and Border properties.

10.43.2.6HotTrack

Applies to
acTreeView and acListView components.

Declaration
```
property HotTrack: Boolean;
```

Description
Specifies whether list items are highlighted when the mouse passes over them.

Set HotTrack to True to provide visual feedback about which item is under the mouse. Set HotTrack to False to suppress the visual feedback about which item is under the mouse.

10.43.2.7ItemPopupMenu

Applies to
acTreeView and acListView components.

© 1998-2006, UtilMind Solutions®

Declaration
```
property ItemPopupMenu: TPopupMenu;
```

Description
The ItemPopupMenu allows to specify the popup menu which could be pop up when user right-clicks the tree nodes, so the popup menu for items could be different of that standard menu for entire TreeView control.

10.43.2.8 RowSelect

Applies to
acTreeView and acListView components.

Declaration
```
property RowSelect: Boolean;
```

Description
Specifies whether the entire row of the selected item is highlighted. Set RowSelect to True to cause the entire row of the selected item to be highlighted.

RowSelect is ignored if ShowLines is True.

10.43.2.9 ToolTips

Applies to
acTreeView component.

Declaration
```
property ToolTips: Boolean;
```

Description
Specifies whether the items in the tree view have tooltips. Set ToolTips to True to specify that items in the tree view control have tooltips (Help Hints).

Specify the ToolTip text in an OnHint event handler using the Hint property.

10.43.3 Methods

10.43.3.1 AddPath

Applies to
acTreeView and acTreeComboBox components.

Declaration
```
function AddPath(Node: TTreeNode; const Path: String; const
PathSeparator: String = '\'): TTreeNode;
```

Description
The AddPath method adds or inserts to the TreeView the *string* or nested *group of strings* (pathes) specified by *Path* parameter and separated by string specifed in *PathSeparator* parameter.

The declaration can be
```
AddPath(nil, 'D:\Test\Test');
```

So, if you call method above, the component will add following structure to the root item of TreeComboBox:
```
D:
  Test
```

© 1998-2006, UtilMind Solutions®

```
Test
```

Return value is the last added child node.

10.43.3.2FindNode

Applies to
acTreeView and acTreeComboBox components.

Declaration
```
function FindNode(const Text: String; IgnoreCase: Boolean = False;
  WholeText: Boolean = True): TTreeNode;
```

Description
The FindNode method search the tree node with the text specified in *Text* parameter.

The *IgnoreCase* parameter specified whether the search should be case sensitive.

The *WholeText* parameter specified whether it should find the node where the length of caption is the same as length of Text parameter (if *WholeText* is False, the method will try to find the node which caption begins with the text specified in *Text* parameter).

Function returns the pointer to tree node if it successfully found, or NIL (NULL) otherwise (if the node with specified Text not exists in the tree view).

See also
FindAbsoluteIndexByNode methods.

10.43.3.3FindAbsoluteIndexByNode

Applies to
acTreeView and acTreeComboBox components.

Declaration
```
function FindAbsoluteIndexByNode(Node: TTreeNode): Integer;
```

Description
The FindAbsoluteIndexByNode method retreives the absolute index from node specified in the *Node* parameter.

To get the node by abolute index you can use following code:

```
Node := acTreeView1.Items[AbsoluteIndex];
```

See also
FindNode methods.

10.43.3.4ExpandTopLevelNodes

Applies to
acTreeView component (and acTreeComboBox as subproperty of TreeOptions structure).

Declaration
```
procedure ExpandTopLevelNodes;
```

Description
The ExpandTopLevelNodes method expands all the top-level nodes in the tree-view. The "top-level"

© 1998-2006, UtilMind Solutions®

nodes is the nodes where the Parent property is nil (no upper node).

Original code

```
procedure TacTreeView.ExpandTopLevelNodes;
var
  I: Integer;
begin
  I := Items.Count;
  if I <> 0 then
   begin
    Items.BeginUpdate;
    try
      for I := 0 to I - 1 do
       if Items[I].Parent = nil then
        Items[I].Expand(False);
    finally
      Items.EndUpdate;
    end;
   end;
end;
```

See also

TreeOptions property of acTreeComboBox component.

10.43.4 Events

10.43.4.1 OnBoldItemBeforeSwitch

Applies to

acTreeView component.

Declaration

```
type
  TacTreeViewBoldItemBeforeSwitchEvent = procedure(Sender: TObject;
    PrevItemIndex, NewItemIndex: Integer;
    PrevNode, NewNode: TTreeNode; var Continue: Boolean) of object;

property OnBoldItemBeforeSwitch: TacTreeViewBoldItemBeforeSwitchEvent;
```

Description

The OnBoldItemBeforeSwitch event occurs before the "bold" mark switches from one node to another.

The *PrevItemIndex* parameter determines the index of previous "bold" node, *NewItemIndex* determines the another node, which is about to become "bold".

In case if you don't want to allow the switching the mark to *NewNode* — set *Continue* parameter to False inside the event handler.

Note

This event available only in Delphi 5/BCB 5 or higher!

See also

BoldNode, ItemIndex and ItemIndexIs properties of BoldItem structure;
OnBoldItemSwitched event.

© 1998-2006, UtilMind Solutions®

10.43.4.2OnBoldItemSwitched

Applies to
acTreeView component.

Declaration
```
property OnBoldItemSwitched: TNotifyEvent;
```

Description
The OnBoldItemSwitched event notifies about successful switching of the "bold" mark from one node to another.

Note
This event available only in Delphi 5/BCB 5 or higher!

See also
BoldNode, ItemIndex and ItemIndexIs properties of BoldItem structure;
OnBoldItemSwitched event.

10.43.4.3OnMouseEnter

Applies to
acLabel, acURLLabel, acRichLabel, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox, acTreeComboBox, acHeader, acHeaderPanel, acTreeView and acListView components.

Declaration
```
property OnMouseEnter: TNotifyEvent;
```

Description
The OnMouseEnter occurs when mouse pointer hovers the control.

See also
OnMouseLeave event.

10.43.4.4OnMouseLeave

Applies to
acLabel, acURLLabel, acRichLabel, acCheckBox, acRadioButton, acButton, acBitBtn, acEdit, acNumberEdit, acLabeledComboBox, acMRUComboBox, acImagesComboBox, acTreeComboBox, acHeader, acHeaderPanel, acTreeView and acListView components.

Declaration
```
property OnMouseLeave: TNotifyEvent;
```

Description
The OnMouseLeave occurs when mouse pointer leaves the control.

See also
OnMouseEnter event.

© 1998-2006, UtilMind Solutions®

10.44 acUnixCrypt

10.44.1 TacUnixCrypt

Overview

The acUnixCrypt component is the ultra fast crypt implementation for Delphi/C++ Builder. It works exactly as **crypt(key, salt)** Unix comand (one-way encryption algorithm), which used to encrypt passwords in .htpasswd files in password protected Web directories.

Description

acUnixCrypt is the password encryption component. It is based on the Data Encryption Standard algorythm with variations intended (among other things) to discourage use of hardware implementations of a key search.

Key property is a user's typed password. Salt is a two characters string chosen from the set [a-zA-Z0-9./]. This string is used to perturb the algo-encrypt repeatedly a constant string (usually a string consisting of all zeros).

The returned value (Result property) is the encrypted password, a series of 13 printable ASCII characters (the first two characters represent the salt itsef).

How to use?

Drop component on your form, specify password in Key property, put any two cahracters to Salt property and get encrypted password from Result. You can play with it even at design-time.

Since the acUnixCrypt component uses one-way encryption algorithm, there is no way to *decrypt* the keys. For authentication you can only compare two encrypted passwords.

Example

```
procedure TForm1.AuthenticationBtnClick(Sender: TObject);
begin
  // we'd like to take salt from two first characters of username
  acUnixCrypt1.Salt := Copy(RealUsername, 1, 2);
  // asking for password
  acUnixCrypt1.Key := InputBox('Authentication',
                              'Enter password:', '');
  // comparing two encrypted passwords
  if acUnixCrypt.Result <> RealCryptedPassword then
   begin
    ShowMessage('Authentication Failed!');
    Application.Terminate;
   end;
end;
```

Warning

The key space consists of 2**56 equal 7.2e16 possible values. Exhaustive searches of this key space are possible using massively parallel computers. Software (cracks), is available which will search for portions of this key space that is generally used by humans for password. Hence, password selection should, at minimum, avoid common words and names.

The DES algorithm itself has a few quirks which make the use of the *crypt* interface a very poor choice for anything other than password authentication. If you are planning on using *crypt* interface for a cryptography project, don't do it: get a good book on encryption.

© 1998-2006, UtilMind Solutions®

10.44.2 Properties

10.44.2.1Key

Applies to
acUnixCrypt component.

Declaration
```
property Key: String;
```

Description
The Key property is a user's typed password. The password can be any length, but if used password with length over 8 characters, only 8 first characters will be used.

See also
Salt and Result properties.

10.44.2.2Salt

Applies to
acUnixCrypt component.

Declaration
```
property Salt: String;
```

Description
Salt is a two characters string chosen from the set [a-zA-Z0-9./]. This string is used to perturb the algo-encrypt repeatedly a constant string (usually a string consisting of all zeros).

See also
Key and Result properties.

10.44.2.3Result

Applies to
acUnixCrypt component.

Declaration
```
property Result: String; // Read-only!
```

Description
The Ruturn property is the read-only property, which contains the result of encryption. The returned value is the encrypted password, a series of 13 printable ASCII characters (the first two characters represent the Salt itsef).

See also
Key and Salt properties.

10.45 acURLLabel

10.45.1 TacURLLabel

Overview
The acURLLabel component is the text label used to redirect user to the Web page, or to compose email message to specified recipient, when user clicks this label.

The acURLLabel is successor of the acLabel component and inherits ALL its features. You can

© 1998-2006, UtilMind Solutions®

specify the highlight and shadow effects, or dynamic styles which changes label appearance when mouse hovers or clicks the label.

Snapshot

properties

Kind	controls the type and purpose of the acURLLabe compnent. The Kind determines whether the URLLabel used as link to the Web page (ukWWW) or to execute default email program (ukEmail).;
URL	specifies the URL location of the Web-resource to redirect user (if Kind = ukWWW);
Email	specifies the parameters which should be transmitted to the email program, opened when user clicks the URLLabel (if Kind is ukEmail)..

See also

acLabel component;
OpenURL and ComposeEmail procedures.

10.45.2 Properties

10.45.2.1Email

Applies to

acURLLabel component.

Declaration

```
type
  TacEmailParams = class
  published
    property Address: String;
    property Name: String;
    property CarbonCopy: String;
    property BlindCarbonCopy: String;
    property Subject: String;
    property Message: String;
  end;

property Email: TacEmailParams;
```

Description

The Email structure specifies the parameters which should be transmitted to the email program, opened when user clicks the URLLabel (if Kind is ukEmail).

To specify the email address and recipient name — use Address and Name properties. To pre-define the subject and message signature — use Subject and Message properties.

If the Address is not specified (leaved empty), the component will use the label text as the E-mail address.

See also

Kind and URL properties.

© 1998-2006, UtilMind Solutions®

10.45.2.1.1 Address

Applies to
acURLLabel component as subproperty of Email structure.

Declaration
```
property Address: String;
```

Description
The Address property specifies the E-mail address of the recipient.

When the user clicks the URLLabel, it will opens the default mailer program and automatically fill the address line with string specified in this property.

If the Address is not specified (leaved empty), the component will use the label text as the E-mail address.

See also
Name, CarbonCopy and BlindCarbonCopy properties.

10.45.2.1.2 BlindCarbonCopy

Applies to
acURLLabel component as subproperty of Email structure.

Declaration
```
property BlindCarbonCopy: String;
```

Description
The BlindCarbonCopy property specifies the email address(es) of the recipient.

When the user clicks the URLLabel, it will opens the default mailer program and automatically fills the "Bcc:" line with string specified in this property.

To specify multiple email addresses, separate them with semicolon (;).

Example
```
BlindCarbonCopy := 'Aleksey Kuznetsov <aleksey@utilmind.com>';
```
or
```
BlindCarbonCopy := 'info@appcontrols.com;aleksey@utilmind.com';
```

See also
Address and CarbonCopy properties.

10.45.2.1.3 CarbonCopy

Applies to
acURLLabel component as subproperty of Email structure.

Declaration
```
property CarbonCopy: String;
```

Description
The CarbonCopy property specifies the email address(es) of the recipient.

When the user clicks the URLLabel, it will opens the default mailer program and automatically fills

© 1998-2006, UtilMind Solutions®

the "Cc:" line with string specified in this property.

To specify multiple email addresses, separate them with semicolon (;).

Example

```
CarbonCopy := 'Aleksey Kuznetsov <aleksey@utilmind.com>';
```

or

```
CarbonCopy := 'info@appcontrols.com;aleksey@utilmind.com';
```

See also

Address and BlindCarbonCopy properties.

10.45.2.1.4 Message

Applies to

acURLLabel component as subproperty of Email structure.

Declaration

```
property Message: String;
```

Description

The Message property specifies the signature which should be pre-defined when user composes the message.

See also

Subject property.

10.45.2.1.5 Name

Applies to

acURLLabel component as subproperty of Email structure.

Declaration

```
property Name: String;
```

Description

The Name property specifies the name of the recipient.

When the user clicks the URLLabel, it will opens the default mailer program and automatically fill the address line with name specified in this property and address specified in the Address property.

See also

Address property.

10.45.2.1.6 Subject

Applies to

acURLLabel component as subproperty of Email structure.

Declaration

```
property Subject: String;
```

Description

The Subject property defines the subject line for message to compose.

See also

© 1998-2006, UtilMind Solutions®

Message property.

10.45.2.2Kind

Applies to
acURLLabel component.

Declaration
```
type
  TacURLLabelKind = (ukWWW, ukEmail);

property Kind: TacURLLabelKind;
```

Description
The Kind property controls the type and purpose of the acURLLabel compnent. The Kind determines whether the URLLabel used as link to the Web page (ukWWW) or to execute default email program (ukEmail).

Parameters	Meaning
ukWWW	starts the default Internet browser and opens the URL specified in the URL property. If the URL is not specified, the compenent will use the label text (Caption property) as URL;
ukEmail	starts the default E-mail program and compose the email message with parameters pre-configured in the Email property. If the Email.Address is not specified, the component will use the label text (Caption) as email address, for whom to compose message.

See also
URL and Email properties.

10.45.2.3URL

Applies to
acURLLabel component.

Declaration
```
property URL: String;
```

Description
The URL property specifies the URL location of the Web-resource to redirect user (if Kind = ukWWW).

When user clicks the URLLabel, the component will execute default browser and opens the Web-page specified in this property. You can specify the locations either with **http://**, **https://** or **ftp://** prefixes.

If URL is not specified (leaved empty), the component will use the label text as the URL and opens the *http://Caption* page.

See also
URLInNewWIndow, Kind and Email properties.

10.45.2.4URLInNewWindow

Applies to
acURLLabel component.

© 1998-2006, UtilMind Solutions®

Declaration

```
property URLInNewWindow: Boolean;
```

Description

The URLInNewWindow property controls how to open the URL: in new browser window (True) or in existing (False).

Set URLInNewWindow to True to open the new browser window before redirecting user to the URL. Or set it to False to open the URL in the existing, active browser window.

Remarks

This property works if MS Internet Explorer is default browser on user's PC. However, if default browser is Netscape Navigator, the URLLabel will *always* open the URL in new window, even if URLInNewWindow is False.

See also

URL property.

10.46 acWebImage

10.46.1 TacWebImage

Overview

The acWebImage is descendant of acProportionalImage component, which displays the picture located somewhere in the Web. All you need to do is just specify the URL of the picture and component will automatically download by HTTP protocol to display.

How to use?

See also

acProportionalImage and acHTTP components.

10.46.2 Properties

10.46.2.1 Alignment

Enter topic text here.

10.46.2.2 CacheOptions

Applies to

acWebImage, acHTTP and acAutoUpgrader components.

Declaration

```
type
  TacCacheOption    = (coAlwaysReload, coReloadIfNoExpireInformation,
                       coReloadUpdatedObjects, coPragmaNoCache,
                       coNoCacheWrite, coCreateTempFilesIfCantCache,
                       coUseCacheIfNetFail);
  TacCacheOptions = set of TacCacheOption;

property CacheOptions: TacCacheOptions;
```

Description

The CacheOptions property controls the cache options for the acWebImage component and determines how the component should use standard Internet Explorer's cache.

© 1998-2006, UtilMind Solutions®

The cache control has following options:

Value	Meaning
coAlwaysReload	Forces a download of the requested file, object, or directory listing from the origin server, not from the cache.;
coReloadIfNoExpireInformation	Forces a reload if there was no Expires time and no LastModified time returned from the server when determining whether to reload the item from the network.;
coReloadUpdatedObjects	Reloads HTTP resources if the resource has been modified since the last time it was downloaded;
coPragmaNoCache	Forces the request to be resolved by the origin server, even if a cached copy exists on the proxy;
coNoCacheWrite	Does not add the downloaded entity to the cache;
coCreateTempFilesIfCantCache	Causes a temporary file to be created if the file cannot be cached;
coUseCacheIfNetFail	Returns the resource from the cache if the network request for the resource fails due if connection with the server has been reset, or the attempt to connect to the server failed.

See also
InternetOptions property.

10.46.2.3ControlCanvas

Enter topic text here.

10.46.2.4DrawRect

Enter topic text here.

10.46.2.5FixedStretching

Enter topic text here.

10.46.2.6JPEGProperties

Enter topic text here.

10.46.2.7HTTPAgent

Applies to
acWebImage component.

Declaration
```
property HTTPAgent: String;
```

Description
The HTTPAgent property specifies the name of HTTP client which downloads the picture from the Web.

The "user agent" can be name of your program or program version. For example, user agent of MS Internet Explorer 5.01 which installed to Windows 98 is `Mozilla/4.0 (compatible; MSIE 5.01; Windows 98)`.

10.46.2.8HTTPPassword

Applies to
acWebImage and acAutoUpgrader components.

© 1998-2006, UtilMind Solutions®

Declaration

```
property HTTPPassword: String;
```

Description

The HTTPPassword property specifies the password to access the data in password protected Web directories. You don't need to specify the username/password if you reading non-protected data.

You can also specify login information dynamically, when it neccessary, in the OnPasswordRequest event handler.

See also

HTTPUsername property
OnPasswordRequest event.

10.46.2.9HTTPUsername

Applies to

acWebImage and acAutoUpgrader components.

Declaration

```
property HTTPUsername: String;
```

Description

The HTTPUsername property specifies the username to access the data in password protected Web directories. You don't need to specify the username/password if you reading non-protected data.

You can also specify login information dynamically, when it neccessary, in the OnPasswordRequest event handler.

See also

HTTPPassword property
OnPasswordRequest event.

HTTPProxy 10.46.2.10

Applies to

acWebImage, acHTTP and acAutoUpgrader components.

Declaration

```
type
  TacHTTPAccessType = (atPreconfig, atDirect, atUseProxy);
  TacHTTPProxy = class
  published
    property AccessType: TacHTTPAccessType; // atPreconfig by default
    property ProxyServer: String;
    property ProxyPort: Integer;    // 8080 by default
    property ProxyBypass: TStrings; // 127.0.0.1 by default
    property ProxyUsername: String;
    property ProxyPassword: String;
  end;

property Proxy: TacHTTPProxy;
```

Description

© 1998-2006, UtilMind Solutions®

The Proxy structure controls the connection type for the acHTTP component and settings for estabilishing connection via proxy. Connection type (AccessType) can be "pre-configured" (acHTTP will use settings from Control Panel), direct, or via specified proxy server.

If proxy requires authentication — write OnProxyAutenticationRequest event handler to prompt and specify user's login information.

See also
OnProxyAuthenticationRequest event.

10.46.2.10.1 AccessType

Applies to
acWebImage, acHTTP and acAutoUpgrader components as subproperty of HTTPProxy structure.

Declaration
```
type
  TacHTTPAccessType = (atPreconfig, atDirect, atProxy);

property AccessType: TacHTTPAccessType;
```

Description
The AccessType property controls how the acHTTP component should access the remote server to download data. The AccessType can be *direct* or via the *proxy* server. If you'd like to use the access type previously configured in the Control Panel, leave the AccessType = atPreconfig.

Values	Meaning
atPreconfig	retrieves the proxy or direct configuration from the registry (user can configure the access type in the Control Panel);
atDirect	uses the direct connection and resolves all host names locally;
atProxy	access the remote data via the proxy server. Passes all requests to the proxy, unless a proxy bypass list is not empty and the name to be resolved bypasses the proxy. To specify the proxy server and port — use ProxyServer and ProxyPort properties. To configure bypass list — use ProxyBypass property.

See also
ProxyServer, ProxyPort and ProxyBypass properties.

10.46.2.10.2 ProxyBypass

Applies to
acWebImage, acHTTP and acAutoUpgrader components as subproperty of HTTPProxy structure.

Declaration
```
property ProxyBypass: String;
```

Description
The ProxyBypass contains the list of host names or IP addresses, or both, that should not be routed through the proxy. The list can contain wildcards. If the ProxyBypass is empty, the acWebImage reads the bypass list from the registry.

You can use wild cards to match domain and host names or addresses — for example, www.*.com;128.*;240.*;*.mygroup.*;*x* and so on. Use semicolon (;) to separate entries.

See also

© 1998-2006, UtilMind Solutions®

ProxyServer and ProxyPort properties.

10.46.2.10.3 ProxyPassword

Applies to
acWebImage, acHTTP and acAutoUpgrader components as subproperty of HTTPProxy structure.

Declaration
```
property ProxyPassword: String;
```

Description
The ProxyPassword property specifies the password required for connection via proxy server, if it's require secure authentication.

Alternativaly you can write OnProxyAuthenticationRequest event handler and specify proxy username and password dynamically.

Note
If your proxy server uses some "challenge-response" authentication scheme — please make sure that `ioKeepConnection` option of the InternetOptions is set to True. Otherwise, connection with proxy could be dropped.

See also
ProxyUsername and InternetOptions properties;
OnProxyAuthenticationRequest event.

10.46.2.10.4 ProxyPort

Applies to
acWebImage, acHTTP and acAutoUpgrader components as subproperty of HTTPProxy structure.

Declaration
```
property ProxyPort: Integer;
```

Description
The ProxyPort property specifies the port number for the proxy server. The default value for proxy ports is **8080**.

See also
ProxyServer and ProxyBypass properties.

10.46.2.10.5 ProxyServer

Applies to
acWebImage, acHTTP and acAutoUpgrader components as subproperty of HTTPProxy structure.

Declaration
```
property ProxyServer: String;
```

Description
The ProxyServer property specifies the host name of the proxy server to use if the proxy access was specified in the AccessType property.

See also
ProxyPort and ProxyBypass properties.

© 1998-2006, UtilMind Solutions®

10.46.2.10.6 ProxyUsername

Applies to
acWebImage, acHTTP and acAutoUpgrader components as subproperty of HTTPProxy structure.

Declaration
```
property ProxyUsername: String;
```

Description
The ProxyUsername property specifies the username required for connection via proxy server, if it's require secure authentication.

Alternativaly you can write OnProxyAuthenticationRequest event handler and specify proxy username and password dynamically.

Note
If your proxy server uses some "challenge-response" authentication scheme — please make sure that `ioKeepConnection` option of the InternetOptions is set to True. Otherwise, connection with proxy could be dropped.

See also
ProxyPassword and InternetOptions properties;
OnProxyAuthenticationRequest event.

InternetOptions
10.46.2.11

Applies to
acWebImage, acHTTP and acAutoUpgrader components.

Declaration
```
type
  TacInternetOption = (ioIgnoreCertificateInvalid,
ioIgnoreCertificateDateInvalid,
                       ioIgnoreRedirectToHTTP, ioIgnoreRedirectToHTTPS,
                       ioKeepConnection, ioNoAuthentication,
                       ioNoAutoRedirect, ioNoCookies);
  TacInternetOptions := set of TacInternetOption;

property InternetOptions: TacInternetOptions;
```

Description
The InternetOptions property is the set of options used to specify some behaviors of acHTTP component.

The property is set of following options:

Value	Meaning
ioIgnoreCertificateInvalid	Disables checking of SSL/PCT-based certificates that are returned from the server against the host name given in the request. WinINet functions use a simple check against certificates by comparing for matching host names and simple wildcarding rules;
ioIgnoreCertificateDateInvalid	Disables checking of SSL/PCT-based certificates for proper validity dates;
ioIgnoreRedirectToHTTP	Disables detection of this special type of redirect. When this flag

© 1998-2006, UtilMind Solutions®

	is used, WinINet functions transparently allow redirects from HTTPS to HTTP URLs;
ioIgnoreRedirectToHTTPS	Disables detection of this special type of redirect. When this flag is used, WinINet functions transparently allow redirects from HTTP to HTTPS URLs;
ioKeepConnection	Uses keep-alive semantics, if available, for the connection. This flag is required for Microsoft Network (MSN), NT LAN Manager (NTLM), and other types of authentication;
ioNoAuthentication	Does not attempt authentication automatically;
ioNoAutoRedirect	Does not automatically handle redirection;
ioNoCookies	Does not automatically add cookie headers to requests, and does not automatically add returned cookies to the cookie database.

See also
CacheOptions property;
OnProxyAuthenticationRequest events.

ThreadPriority
10.46.2.12

Enter topic text here.

URL
10.46.2.13

Enter topic text here.

10.46.3 Methods

10.46.3.1Retry

Applies to
acWebImage component.

Declaration
```
procedure Retry;
```

Description
The Retry method retries to download the image from location specified in URL property.

You can use this method to re-download the picture in case if some error occured or downloading was terminated after calling Abort method.

Example
```
procedure TPreview.RetryBtnClick(Sender: TObject);
begin
  acWebImage1.Retry;
end;
```

See also
Abort method;
OnError and OnAborted events.

© 1998-2006, UtilMind Solutions®

10.46.3.2Abort

Applies to
acWebImage, acHTTP, acAutoUpgrader, acSendMail and acSQLQueryThread components.

Declaration
```
procedure Abort;
```

Description
The Abort method imediately terminates execution of thread which reads the data from the Web by HTTP(s) protocol. After calling the Abort method, the OnAborted event occurs.

See also
Retry method; OnAborted event.

10.46.4 Events

10.46.4.1OnAborted

Enter topic text here.

10.46.4.2OnDone

Enter topic text here.

10.46.4.3OnDownloadProgress

Enter topic text here.

10.46.4.4OnError

Enter topic text here.

10.46.4.5OnPasswordRequest

Applies to
acWebImage, acHTTP and acAutoUpgrader components.

Declaration
```
type
  TacHTTPPasswordRequestEvent = procedure(Sender: TObject;
    const Realm: String; var TryAgain: Boolean) of object;

property OnPasswordRequest: TacHTTPPasswordRequestEvent;
```

Description
The OnPasswordRequest event can be used to implement the dialog which asks user for his username and password to access the password protected Web area.

Set *TryAgain* parameter to True to retry the HTTP query with correct username and password. (Don't forget to specify correct login information in this event handler to Username and Password properties.)

Note
If you leave this event unhandled (set TryAgain parameter to False), the component will generate error code #401 (Access Denied) and OnError event occur.

Example

© 1998-2006, UtilMind Solutions®

Delphi:

```
procedure TForm1.WebImage1PasswordRequest(Sender: TObject;
  const Realm: String; var TryAgain: Boolean);
begin
  { UserPassForm is any form with two edit boxes used for entering the
login information (username and password) }
  UserPassForm.RealmLabel := Realm;
  if UserPassForm.ShowModal = ID_OK then
  begin
    acHTTP1.Username := UserPassForm.UsernameEdit.Text;
    acHTTP1.Password := UserPassForm.PasswordEdit.Text;
    TryAgain := True; // Retry HTTP query (download attempt)
  end;
end;
```

C++ Builder:

```
void __fastcall TForm1::acHTTP1PasswordRequest(TObject *Sender,
      AnsiString Realm, bool &TryAgain)
{
  UserPassForm->RealmLabel = Realm;
  if (UserPassForm->ShowModal() == ID_OK)
  {
    acHTTP1->Username = UserPassForm->UsernameEdit->Text;
    acHTTP1->Password = UserPassForm->PasswordEdit->Text;
    TryAgain = True; // Retry HTTP query (download attempt)
  };
}
```

See also

Username and Password properties; OnHTTPError event;
HTTP Status Codes.

10.46.4.6OnProxyAuthenticationRequest

Applies to

acWebImage, acHTTP and acAutoUpgrader components.

Declaration

```
type
  TacHTTPProxyAuthenticationRequestEvent = procedure(Sender: TObject;
    var ProxyUsername, ProxyPassword: String;
    var TryAgain: Boolean) of object;

property OnProxyAuthenticationRequest:
TacHTTPProxyAuthenticationRequestEvent;
```

Description

The OnProxyAuthenticationRequest event should be used to prompt users for their username/password to access the Web via secure proxy server which requires authentication.

Write this event to prompt and specify the *ProxyUsername* and *ProxyPassword* parameters, required for the proxy authentication, and set *TryAgain* parameter to True, to retry the HTTP query with provided login information.

Alternatively you can specify ProxyUsername and ProxyPassword properties in the Proxy

© 1998-2006, UtilMind Solutions®

structure, before the request. In case if specified username and password is okay, the OnProxyAuthenticationRequest event will not occur.

Notes

If your proxy server uses some "challenge-response" authentication scheme — please make sure that `ioKeepConnection` option of the InternetOptions is set to True. Otherwise, connection with proxy could be dropped.

If you leave this event unhandled (if you set TryAgain parameter to False), the component will generate error code #407 (Proxy Authentication Required).

See also

Proxy and InternetOptions properties;
HTTP Status Codes.

10.46.4.7OnUnsupportedFormat

Enter topic text here.

11 Additions to the Delphi/BCB IDE

11.1 Additional Colors

Unit

acGraphics

Description

The "AppControls pack" installs an additional colors to the Delphi / C++ Builder IDE, together with components. You can see these new colors in the Object Inspector, selecting any property with TColor type.

TColor is used to specify the color of an object. It is used by the Color property of many components and by a number of other properties that specify color values. The acGraphics unit contains definitions of useful additional constants for TColor.

If you specify TColor as a specific 4-byte hexadecimal number instead of using the constants defined in the acGraphics unit, the low three bytes represent RGB color intensities for blue, green, and red, respectively. The value $00FF0000 represents full-intensity, pure blue, $0000FF00 is pure green, and $000000FF is pure red. $00000000 is black and $00FFFFFF is white.

New system colors

Constant	Value	Sample
clHotLight	COLOR_HOTLIGHT or $80000000	
clGradientActiveCaption	COLOR_GRADIENTACTIVECAPTION or $80000000	
clGradientInactiveCaption	COLOR_GRADIENTINACTIVECAPTION or $80000000	

Color samples listed above may differ to real colors on your machine. Theay're always equal to system colors settings on user's PC.

New custom colors

Contstant	Value	Sample
clContextHelp	$EEFFFF	
clBrown	$003090	
clOliveGreen	$003030	

© 1998-2006, UtilMind Solutions®

clDarkGreen	$003000	
clDarkTeal	$603000	
clIndigo	$903030	
clNight	$303030	
clOrange	$0068FF	
clBlueGray	$906060	
clLightOrange	$0098FF	
clSeaGreen	$609830	
clLightBlue	$FF6830	
clMetal	$909090	
clGold	$00C8FF	
clSkyBlue	$FFC800	
clPlum	$603090	
clRose	$D098FF	
clTan	$A0C8FF	
clLightYellow	$90FFFF	
clLightGreen	$D0FFD0	
clLightTurquoise	$FFFFC0	
clPaleBlue	$FFC890	
clLavender	$FF98C0	
clCream	$A6CAF0	
clMoneyGreen	$C0DCC0	

See also
Additional Cursors

11.2 Additional Cursors

Unit
acGraphics

Description
The "AppControls pack" installs an additional cursors to the Delphi / C++ Builder IDE, together with components. You can see these new cursors in the Object Inspector, selecting any property with TCursor type.

TCursor is an index into the global list of available cursors maintained by the Screen variable. TCursor values can include indexes to custom cursors added by an application, in addition to the predefined constants defined in the controls unit. The following table lists the **additional cursor constants** and their position in the Cursors property array:

New Cursors

Constant	Value	Image
crLinkSelect	-151	
crHandwriting	-152	
cr3dAppStart	-153	
cr3dArrow	-154	
cr3dArrowNE	-155	

© 1998-2006, UtilMind Solutions®

cr3dArrowScrollHorz	-156
cr3dArrowScrollVert	-157
cr3dArrowSize	-158
cr3dArrowResize	-159
cr3dArrowWarning	-160
cr3dCross	-161
cr3dDragSelection	-162
cr3dDragSelectionPlus	-163
cr3dHandwriting	-164
cr3dHelp	-165
cr3dHourGlass	-166
cr3dIBeam	-167
cr3dNo	-168
cr3dSize	-169
cr3dSizeNESW	-170
cr3dSizeNS	-171
cr3dSizeNWSE	-172
cr3dSizeWE	-173
cr3dTool	-174
cr3dUpArrow	-175
crArrowNE	-176
crArrowNW	-177
crArrowScrollHorz	-178
crArrowScrollVert	-179

© 1998-2006, UtilMind Solutions®

crArrowSE	-180	
crArrowSize	-181	
crArrowResize	-182	
crArrowSW	-183	
crDragFile	-184	
crDragFiles	-185	
crDragSelection	-186	
crDragSelectionPlus	-187	
crHand	-188	
crItBeam	-189	
crPen	-190	
crTool	-191	
crZoomIn	-192	
crZoomOut	-193	

Note

If you would like to use these cursors in your application, make sure that acGraphics module listed in the units section of your program.

See also

Additional Colors

12 Bonus Routines

List of routines provided by **acGraphics** unit:

Advanced GDI routines

acMakeTranslucentWindow
acMakeOpaqueWindow
acAnimateWindow
acFlashWindow

AnimateWindow
FlashWindowEx
SetLayeredWindowAttributes

© 1998-2006, UtilMind Solutions®

List of routines provided by **acUtils** unit:

Numerical conversion
- DecToHex
- HexToDec
- DecToBin
- BinToDec
- DecToRoman
- Int2x32To64

Shell utilities
- RunControlPanel
- RunCPL
- ExploreFolder
- OpenURL
- OpenWithDlg
- RepaintScreen
- MinimizeAll
- RestoreAll
- GetExecutableByExtension
- GetSystemImageList
- ComposeEmail

String/filename routines
- SplitStr
- LeftPart
- RightPart
- LeftPartR
- RightPartR
- PosR
- SpitFileNameAndParams
- IsValidEmail
- IncludeTrailingBackslash
- ExcludeTrailingBackslash
- IncludeLeadingChar
- ExcludeLeadingChar
- StringReplace
- HasParamStr
- FullParamStr
- CaseAnsiCompareText
- URLEncode
- URLDecode
- StateNameToStateCode
- StateCodeToStateName
- StrToStrList
- FileReadString
- FileWriteString

MySQL String routines
- ESC
- SQLDateTimeToDateTime
- DateTimeToSQLDateTime
- DateTimeToSQLDate
- DateTimeToSQLTime

© 1998-2006, UtilMind Solutions®

Date/Time routines
- StrToDateTimeDef
- IsLeapYear
- DaysPerMonth
- GMTToLocalTime
- LocalTimeToGMT
- GMT
- InternetTimeToDateTime
- DateTimeToFileTime
- FileTimeToDateTime
- SetFileTimes
- SameDateTime

System Information
- GetOS
- IsNT
- IsWin2k
- IsXP
- Is256ColorTrayIconAllowed
- GetCPUSpeed
- IsScreenSaverRunning
- GetTempDir
- GetSystemDir
- GetWindowsDir
- GetEnvironmentString
- GetSystemIconIndex

PC Speaker and System Sound routines
- Sound
- NoSound
- Delay
- PlaySystemSound
- StopSystemSound

Easy Registry utilities
- IsRegValueExists
- RegReadStr
- RegWriteStr
- RegReadInt
- RegWriteInt

Miscellaneous
- IsDirectoryExists
- IsDirectoryEmpty
- ExtractFileSize
- FileSizeToStr
- GetShiftState
- GetKeyName
- GetWorkArea
- GetWindowTitle
- PercentsOf
- StartScreenSaver
- LoadResourceToStream
- ExtractResourceToFile
- StreamToString

© 1998-2006, UtilMind Solutions®

PutStringToStream
GetStringFromStream
MsgBox
InfoBox
ErrorBox
XORStr
ShutdownApplication
HideFromTaskList
IsMetricSystem
IsDebug
Wait
StopWait
LogToFile
ExecAndWait
ControlAtPoint

List of routines provided by **acInet** unit (Internet functions):
IsOnline
GetLocalHostName
GetLocalIP
IPToStr
StrToIP
ParseURL
URLToHostName
URLToFileName
SockConnect
SockDisconnect

List of routines provided by **acRusCodeTables** unit (text conversion for major Russian code tables):
DOSToKOI
DOSToWin
WinToDOS
WinToKOI
KOIToDOS
KOIToWin

List of routines provided by **acXPThemes** unit:
IsThemeActive
IsAppThemed
IsThemeEnabled
OpenThemeData
CloseThemeData
DrawThemeBackground
DrawThemeParentBackground
DrawThemedControl

12.1 Advanced GDI routines

Unit
acGraphics

© 1998-2006, UtilMind Solutions®

Declarations

```
procedure acMakeTranslucentWindow(Wnd: THandle; Color: TColor;
   TransparencyRate: Byte; UseColor: Boolean);
procedure acMakeOpaqueWindow(Wnd: THandle);

procedure acAnimateWindow(Form: TForm; Time: Word; AnimationType:
   TacWin2kAnimationFlags);
procedure acFlashWindow(Wnd: THandle; Count: Integer; FlashTaskIcon:
   Boolean);

{ exported from user32.dll }
function AnimateWindow(hWnd: HWND; dwTime: DWORD; dwFlags: DWORD): BOOL;
function FlashWindowEx(fwi: FLASHWINFO): BOOL;
function SetLayeredWindowAttributes(hWnd: hWnd; crKey: COLORREF; bAlpha:
   Byte; dwFlags: LongInt): LongInt;
```

Description

Some advanced GDI routines which works on Windows 98 and higher.

Procedures with "ac" prefix is enchancements of GDI routines exported from user32.dll. For example, acAnimateWindow does same as AnimateWindow, plus refreshes all controls on the form.

acMakeTranslucentWindow set up the WS_EX_LAYERED attribute to the form style before making the window translucent (using SetLayeredWindowAttributes API). acMakeOpaqueWindow clears the layered attributes from the form.

See also

Windows XP Themes API functions

12.1.1 acMakeTranslucentWindow

Unit

acGraphics

Declaration

```
procedure acMakeTranslucentWindow(Wnd: THandle; TransparencyRate: Byte);
  // TransparencyRate must be 0%..100%
```

Description

The **acMakeTranslucentWindow** makes the form translucent. The rate of transparency specified by *TransparencyRate* parameter. To return the form style to normal state — use acMakeOpaqueWindow procedure.

The acMakeTranslucentWindow makes the window layered, setting up the WS_EX_LAYERED attribute to the form style, then makes it translucent using the SetLayeredWindowAttributes function.

Using a layered window can significantly improve performance and visual effects for a window that has a complex shape, animates its shape, or wishes to use alpha blending effects. The system automatically composes and repaints layered windows and the windows of underlying applications. As a result, layered windows are rendered smoothly, without the flickering typical of complex window regions. In addition, layered windows can be partially translucent, that is, alpha-blended.

Requirements

Windows NT/2000: Requires Windows 2000 or later.
Windows 95/98: Unsupported.

© 1998-2006, UtilMind Solutions®

If you call this procedure on Win95, Win98 or WinNT, nothing will happends.

See also
acMakeOpaqueWindow procedure and acWin2kEffects component.

12.1.2 acMakeOpaqueWindow

Unit
acGraphics

Declaration
```
procedure acMakeOpaqueWindow(Wnd: THandle);
```

Description
The **acMakeOpaqueWindow** procedure returns the translucent form to its normal state clearing the layered attributes from the window style (clearing the WS_EX_LAYERED attribute).

To make the form translucent — use acMakeTranslucentWindow.

See also
acMakeTranslucentWindow procedure and acWin2kEffects component.

12.1.3 acAnimateWindow

Unit
acGraphics

Declaration
```
procedure acAnimateWindow(Form: TForm; Time: Word; AnimationType:
  TacWin2kAnimationFlags; Open: Boolean);
```

Description
The **acAnimateWindow** procedure produces special effects when showing or hiding windows. There are three types of animation: roll, slide and alpha-blended fade (blending works on Win2000 and higher).

Use the **acAnimateWindow** in *OnShow* and *OnClose* events of your form. Also you can use acWin2kEffects component instead of this procedure.

Parameters

Form	Pointer to the form to animate;
Time	Duration of animation;
AnimationType	Flags specifying the type of the animation;
Open	Which effect to draw. Set Open parameter to True to show the opening effect and False to show the hiding effect.

Requirements
Windows NT/2000: Requires Windows 2000 or later.
Windows 95/98: Requires Windows 98 or later.

If you call this function on Win95, or on Win98 with AW_BLEND parameter — nothing happends.

See also
acWin2kEffects component and AnimateWindow function.

© 1998-2006, UtilMind Solutions®

12.1.4 acFlashWindow

Unit
acGraphics

Declaration
```
procedure acFlashWindow(Wnd: THandle; Count: Integer; FlashTaskIcon:
  Boolean; StopOnActivate: Boolean [= False]);
```

Description
The **acFlashWindow** procedure flashes the caption of specified window and the application taskbar icon, if *FlashTaskIcon* parameter is True. Number of times to flash the window specifyed in the *Count* parameter. Set *StopOnActivate* parameter to True if you wish to flash the caption and task icon of current form, until the window comes to the foreground.

Typically, you flash a window to inform the user that the window requires attention but does not currently have the keyboard focus. When a window flashes, it appears to change from inactive to active status. An inactive caption bar changes to an active caption bar; an active caption bar changes to an inactive caption bar.

Examples
```
// this will flash the window 4 times
acFlashWindow(Handle, 4, True);
// flash continuously, until the window comes to foreground
acFlashWindow(Handle, MaxInt, True, True);
```

Requirements
Windows NT/2000: Requires Windows 2000 or later.
Windows 95/98: Requires Windows 98 or later.

If you call this function on Win95/NT4 — nothing happends.

See also
acWin2kEffects component and FlashWindowEx function.

12.1.5 AnimateWindow

Unit
acGraphics

Declaration
```
function AnimateWindow(
   hWnd: HWND;      // handle to window
   dwTime: DWORD;   // duration of animation
   dwFlags: DWORD   // animation type
): BOOL;
```

Description
The AnimateWindow function enables you to produce special effects when showing or hiding windows. There are three types of animation: roll, slide, and alpha-blended fade.

Parameters
hwnd
Handle to the window to animate. The calling thread must own this window.
dwTime

© 1998-2006, UtilMind Solutions®

Specifies how long it takes to play the animation, in milliseconds. Typically, an animation takes 200 milliseconds to play.

dwFlags

Specifies the type of animation. This parameter can be one or more of the following values.

Value	Description
AW_SLIDE	Uses slide animation. By default, roll animation is used. This flag is ignored when used with AW_CENTER.
AW_ACTIVATE	Activates the window. Do not use this value with AW_HIDE.
AW_BLEND	Uses a fade effect. This flag can be used only if *hwnd* is a top-level window. This effect works on **Win2000** or later.
AW_HIDE	Hides the window. By default, the window is shown.
AW_CENTER	Makes the window appear to collapse inward if AW_HIDE is used or expand outward if the AW_HIDE is not used.
AW_HOR_POSITIVE	Animates the window from left to right. This flag can be used with roll or slide animation. It is ignored when used with AW_CENTER or AW_BLEND.
AW_HOR_NEGATIVE	Animates the window from right to left. This flag can be used with roll or slide animation. It is ignored when used with AW_CENTER or AW_BLEND.
AW_VER_POSITIVE	Animates the window from top to bottom. This flag can be used with roll or slide animation. It is ignored when used with AW_CENTER or AW_BLEND.
AW_VER_NEGATIVE	Animates the window from bottom to top. This flag can be used with roll or slide animation. It is ignored when used with AW_CENTER or AW_BLEND.

Return Values

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. The function will fail in the following situations:

- The window uses the window region.
- The window is already visible and you are trying to show the window.
- The window is already hidden and you are trying to hide the window.

To get extended error information, call the **GetLastError** function.

Remarks

You can combine AW_HOR_POSITIVE or AW_HOR_NEGATIVE with AW_VER_POSITIVE or AW_VER_NEGATIVE to animate a window diagonally.

The window procedures for the window and its child windows may need to handle any **WM_PRINT** or **WM_PRINTCLIENT** messages. Dialog boxes, controls, and common controls already handle **WM_PRINTCLIENT**. The default window procedure already handles **WM_PRINT**.

Requirements

Windows NT/2000: Requires Windows 2000 or later.
Windows 95/98: Requires Windows 98 or later.

If you call this function on Win95, or on Win98 with AW_BLEND parameter — nothing happends.

Example

© 1998-2006, UtilMind Solutions®

```
procedure TForm1.FormShow(Sender: TObject);
var
  I: Integer;
begin
  AnimateWindow(Handle, 300, AW_HOR_POSITIVE + AW_VER_POSITIVE +
    AW_SLIDE + AW_ACTIVATE);
end;
```

See also
acAnimateWindow function and acWin2kEffects component.

12.1.6 FlashWindowEx

Unit
acGraphics

Declaration
```
function FlashWindowEx(
  fwi: FLASHWINFO // flash status information
): BOOL;
```

Description
The **FlashWindowEx** function flashes the specified window. It does not change the active state of the window.

Parameters
fwi The **FLASHWINFO** structure.

Return Values
The return value specifies the window's state before the call to the **FlashWindowEx** function. If the window caption was drawn as active before the call, the return value is nonzero. Otherwise, the return value is zero.

Remarks
Typically, you flash a window to inform the user that the window requires attention but does not currently have the keyboard focus. When a window flashes, it appears to change from inactive to active status. An inactive caption bar changes to an active caption bar; an active caption bar changes to an inactive caption bar.

Requirements
Windows NT/2000: Requires Windows 2000 or later.
Windows 95/98: Requires Windows 98 or later.

If you call this function on Win95/NT4 — nothing happends.

See also
FLASHWINFO structure and usage example.
acFlashWindow procedure.
acWin2kEffects component.

12.1.7 SetLayeredWindowAttributes

Unit
acGraphics

Declaration

© 1998-2006, UtilMind Solutions®

```
function SetLayeredWindowAttributes(
   hWnd: hWnd;        // handle to the layered window
   crKey: COLORREF;   // specifies the color key
   bAlpha: Byte;      // value for blend function
   dwFlags: LongInt   // action
): LongInt;
```

Description

The **SetLayeredWindowAttributes** function sets the opacity and transparency color key of a layered window.

Parameters

hwnd

[in] Handle to the layered window. A layered window is created by specifying WS_EX_LAYERED when creating the window with the **CreateWindowEx** function or by setting WS_EX_LAYERED via **SetWindowLong** after the window has been created.

crKey

[in] Pointer to a **COLORREF** value that specifies the transparency color key to be used when composing the layered window. All pixels painted by the window in this color will be transparent. To convert TColor value to the **COLORREF**, use **ColorToRGB** function.

bAlpha

[in] Alpha value used to describe the opacity of the layered window. Similar to the **SourceConstantAlpha** member of the **BLENDFUNCTION** structure. When *bAlpha* is 0, the window is completely transparent. When *bAlpha* is 255, the window is opaque.

dwFlags

[in] Specifies an action to take. This parameter can be one or more of the following values.

Value	Meaning
LWA_COLORKEY	Use *crKey* as the transparency color.
LWA_ALPHA	Use *bAlpha* to determine the opacity of the layered window.

Return Values

If the function succeeds, the return value is nonzero.
If the function fails, the return value is zero. To get extended error information, call **GetLastError**.

Requirements

Windows NT/2000: Requires Windows 2000 or later.
Windows 95/98: Unsupported.

If you call this function on Win95, or on Win98 with AW_BLEND parameter — nothing happends.

See also

acWin2kEffects component.
acMakeTranslucentWindow and acMakeOpaqueWindow procedures.

12.2 Numerical conversion routines

Unit

acUtils

Declarations

```
function DecToHex(Value: LongInt): String;
function HexToDec(Value: String): LongInt;
function DecToBin(Value: LongInt): String;
function BinToDec(Value: String): LongInt;
```

© 1998-2006, UtilMind Solutions®

```
function DecToRoman(Value: Integer): String;
function Int2x32To64(IntHi, IntLo: DWord): LongLong;
```

Description
Routines for numerical conversions for major scales of notation.

12.2.1 DecToHex

Unit
acUtils

Declaration
```
function DecToHex(Value: LongInt): String;
```

Description
The **DecToHex** function converts decimal integer value (aValue parameter) to hexadecimal string value (Result).

For example, result of `DecToHex(123456)` will be `'1E240'`.

See also
HexToDec, DecToBin and BinToDec functions.

12.2.2 HexToDec

Unit
acUtils

Declaration
```
function HexToDec(Value: String): LongInt;
```

Description
The **HexToDec** function converts hexadecimal string value (aValue parameter) to decimal integer value (Result).

For example, result of `HexToDec('1E240')` will be `123456`.

See also
DecToHex, DecToBin and BinToDec functions.

12.2.3 DecToBin

Unit
acUtils

Declaration
```
function DecToBin(Value: LongInt): String;
```

Description
The **DecToBin** function converts decimal integer value (aValue parameter) to binary string value (Result).

For example, result of `DecToBin(123456)` will be `11110001001000000`.

See also
BinToDec, DecToHex and HexToDec functions.

© 1998-2006, UtilMind Solutions®

12.2.4 BinToDec

Unit
acUtils

Declaration
```
function BinToDec(Value: String): LongInt;
```

Description
The **BinToDec** function converts binary string value (aValue parameter) to decimal integer value (Result).

For example, result of `BinToDec('11110001001000000')` will be `123456`.

See also
DecToBin, DecToHex and HexToDec functions.

12.2.5 DecToRoman

Unit
acUtils

Declaration
```
function DecToRoman(Value: Integer): String;
```

Description
The **DecToRoman** function converts the integer value (with Arabic digits) to the string with Roman digits.

12.2.6 Int2x32ToInt64

Unit
acUtils

Declaration
```
function Int2x32ToInt64(IntHi, IntLo: DWord): LongLong;
```

Description
The **Int2x32ToInt64** function function multiplies two signed 32-bit integers, returning a signed 64-bit integer result.

Original code
```
function Int2x32ToInt64(const IntHi, IntLo: DWord): LongLong;

{$IFNDEF D4}
  function UIntToInt(const Int: DWord): LongLong;
  type
    TDoubleWord = packed record
      case Integer of
        0: (LoWord, HiWord: Word);
        1: (DWord: DWord);
    end;
  var
    DW: TDoubleWord;
  begin
    Result := Int;
```

© 1998-2006, UtilMind Solutions®

```
    if Int < 0 then
     with DW do
      begin
        DWord := Int;
        Result := HiWord;
        Result := Result * $10000 + LoWord; // shl 16 + LoWord
      end;
  end;
{$ENDIF}

begin
{$IFDEF D4}
  Result := IntHi;
  Result := Result shl 32 + IntLo;
{$ELSE}
  // IntHi shl 32 + IntLo
  Result := UIntToInt(IntHi) * $10000 * $10000 + UIntToInt(IntLo);
{$ENDIF}
end;
```

12.3 Shell utilities routines

Unit
acUtils

Declarations

```
procedure ComposeEmail(Address, Name, Subject, Message, CC, BCC:
  String);
function  GetExecutableByExtension(Extension: String): String;
function  GetSystemImageList(Size: Integer): THandle;
procedure MinimizeAll;    // minimizes ALL windows
procedure ExploreFolder(URL: String);
procedure OpenURL(URL: String; InNewWindow: Boolean);
function  OpenWithDlg(FileName: String): Boolean;
procedure RepaintScreen; // repains whole screen
procedure RestoreAll;    // restore ALL windows
procedure RunControlPanel(CmdShow: Integer);
procedure RunCPL(FileName: String; CmdShow: Integer);
```

Descriptions
Various shell routines.

12.3.1 ComposeEmail

Unit
acUtils

Declaration

```
procedure ComposeEmail(Address, Name, Subject, Message, CC, BCC:
   String);
```

Description
The **ComposeEmail** procedure opens the default email program and automatically fills the form fields with specified parameters:

Parameter	Meaning

© 1998-2006, UtilMind Solutions®

Address	the email address of recipient;
Name	recipient name;
Subject	pre-defined subject line;
Message	pre-defined signature;
CC	carbon-copies (separate multiple email addresses with semicolon ";")
BC*C*	blind carbon-copy.

See also
OpenURL and IsOnline functions;
acURLLabel component.

12.3.2 GetExecutableByExtension

Unit
acUtils

Declaration
```
function GetExecutableByExtension(Extension: String): String;
```

Description
The **GetExecutableByExtension** function returns the path to *executable file* for file extension specified by *Extension* string parameter.

Examples
Result of *GetExecutableByExtension('html')* will be the path to default browser.
Result of *GetExecutableByExtension('wav')* will be the path to default sound player.
Result of *GetExecutableByExtension('bmp')* will be the path to MSPaint (or other bitmap editor).

See also
acExtAssociation component.

12.3.3 GetSystemImageList

Unit
acUtils

Declaration
```
function GetSystemImageList(Size: Integer): THandle;
   // Size can be only 16 (16x16) or 32 (32x32)
```

Description
The **GetSystemImageList** function retrieves handle of the system image list (which contains icons for every shell object, i.e: file types, exe-icons, My Computer/Recycled Bin icons etc).

In the Windows NT this function also adds required icon overlays (shortcut and shared folder icons) to the end of image list.

Remarks
The Windows shell operates only with 16x16 and 32x32 icons. This means that value of Size parameter can be only 16 or 32.

See also
acSystemImageList component.

© 1998-2006, UtilMind Solutions®

12.3.4 MinimizeAll

Unit
acUtils

Declaration
```
procedure MinimizeAll;
```

Description
Minimizes ALL windows to the task bar.

See also
RestoreAll procedure.

12.3.5 ExploreFolder

Unit
acUtils

Declaration
```
procedure ExploreFolder(const Folder: String);
```

Description
The **ExploreFolder** function opens the shell explorer window and displays the content of specified folder.

You can specify the file name instead of folder, but the function will open required folder anyway.

See also
OpenURL and ComposeEmail functions.

12.3.6 OpenURL

Unit
acUtils

Declaration
```
procedure OpenURL(URL: String; InNewWindow: Boolean = True);
```

Description
The **OpenURL** procedure opens the Internet location, specified by *URL* string parameter in default browser. If you would like to open URL in new browser window — set *InNewWindow* parameter to True (or False if you wish to show the Web page in already opened window).

Remarks
The *InNewWindow* parameter works if MS Internet Explorer is default browser on user's PC. However, if default browser is Netscape Navigator, the URLLabel will *always* open the URL in new window, even if URLInNewWindow is False.

See also
ComposeEmail, IsOnline and ExploreFolder functions;
acURLLabel component.

© 1998-2006, UtilMind Solutions®

12.3.7 OpenWithDlg

Unit
acUtils

Declaration
```
function OpenWithDlg(FileName: String): Boolean;//True if succeed
```

Description
The **OpenWithDlg** function executes "Open With" dialog for file specified by *FileName* parameter. The "Open With" dialog mostly used to execute file with unknown type and specify which program can execute that file. Function returns True if succeed or False otherwise.

Original code
```
function OpenWithDlg(FileName: String): Boolean;
begin
  try
    Result := WinExec(PChar('rundll32 shell32,OpenAs_RunDLL ' +
FileName), SW_SHOWNORMAL) > 31;
  except
  end;
end;
```

12.3.8 RepaintScreen

Unit
acUtils

Declaration
```
procedure RepaintScreen;
```

Description
The **RepaintScreen** procedure completely refreshes user's desktop. Call RepaintScreen to repaint all windows of running programs.

Original code
```
procedure RepaintScreen;
begin
  try
    WinExec('rundll32 user,repaintscreen', SW_SHOWNORMAL);
  except
  end;
end;
```

12.3.9 RestoreAll

Unit
acUtils

Declaration
```
procedure RestoreAll;
```

Description
Restores ALL minimized windows from the task bar.

See also

© 1998-2006, UtilMind Solutions®

MinimizeAll procedure.

12.3.10 RunControlPanel

Unit
acUtils

Declaration
```
procedure RunControlPanel(CmdShow: Integer = SW_SHOWNORMAL);
```

Description
The **RunControlPanel** procedure executes the Control Panel.

Parameters
CmdShow Specifies how the window is to be shown. The default value in this parameter is SW_SHOWNORMAL, but the possible values are following:

Value	Meaning
SW_HIDE	Hides the window and activates another window.
SW_MAXIMIZE	Maximizes the specified window.
SW_MINIMIZE	Minimizes the specified window and activates the next top-level window in the Z order.
SW_RESTORE	Activates and displays the window. If the window is minimized or maximized, Windows restores it to its original size and position. An application should specify this flag when restoring a minimized window.
SW_SHOW	Activates the window and displays it in its current size and position.
SW_SHOWDEFAULT	Sets the show state based on the SW_ flag specified in the STARTUPINFO structure passed to the CreateProcess function by the program that started the application.
SW_SHOWMAXIMIZED	Activates the window and displays it as a maximized window.
SW_SHOWMINIMIZED	Activates the window and displays it as a minimized window.
SW_SHOWMINNOACTIVE	Displays the window as a minimized window. The active window remains active.
SW_SHOWNA	Displays the window in its current state. The active window remains active.
SW_SHOWNOACTIVATE	Displays a window in its most recent size and position. The active window remains active.
SW_SHOWNORMAL	Activates and displays a window. If the window is minimized or maximized, Windows restores it to its original size and position. An application should specify this flag when displaying the window for the first time.

Original code
```
procedure RunControlPanel(CmdShow: Integer);
begin
  try
    WinExec('rundll32 shell32,Control_RunDLL', CmdShow);
  except
  end;
end;
```

© 1998-2006, UtilMind Solutions®

See also
RunCPL routine.

12.3.11 RunCPL

Unit
acUtils

Declaration
```
procedure RunCPL(FileName: String; CmdShow: Integer = SW_SHOWNORMAL);
```

Description
The **RunCPL** routine executes the CPL files which used for system settings in Control Panel. *FileName* parameter specifies the CPL file name ('something.cpl') and *CmdShow* parameter specifies how a CPL window is to be shown.

Parameters

FileName CPL file name. For example, 'desk.cpl' (Display properties) or 'intl.cpl' (Regional settings properties)

CmdShow Specifies how the window is to be shown. The default value in this parameter is SW_SHOWNORMAL, but the possible values are following:

Value	Meaning
SW_HIDE	Hides the window and activates another window.
SW_MAXIMIZE	Maximizes the specified window.
SW_MINIMIZE	Minimizes the specified window and activates the next top-level window in the Z order.
SW_RESTORE	Activates and displays the window. If the window is minimized or maximized, Windows restores it to its original size and position. An application should specify this flag when restoring a minimized window.
SW_SHOW	Activates the window and displays it in its current size and position.
SW_SHOWDEFAULT	Sets the show state based on the SW_ flag specified in the STARTUPINFO structure passed to the CreateProcess function by the program that started the application.
SW_SHOWMAXIMIZED	Activates the window and displays it as a maximized window.
SW_SHOWMINIMIZED	Activates the window and displays it as a minimized window.
SW_SHOWMINNOACTIVE	Displays the window as a minimized window. The active window remains active.
SW_SHOWNA	Displays the window in its current state. The active window remains active.
SW_SHOWNOACTIVATE	Displays a window in its most recent size and position. The active window remains active.
SW_SHOWNORMAL	Activates and displays a window. If the window is minimized or maximized, Windows restores it to its original size and position. An application should specify this flag when displaying the window for the first time.

Examples *(click the links to see demo)*
RunCPL('desk.cpl'); Display properties
RunCPL('main.cpl', sw_ShowNormal'); Mouse properties
RunCPL('mmsys.cpl', sw_ShowNormal); Multimedia properties

© 1998-2006, UtilMind Solutions®

RunCPL('modem.cpl', sw_ShowNormal); Modems properties

Original code

```
procedure RunCPL(FileName: String; CmdShow: Integer);
begin
  try
    WinExec(PChar('rundll32 shell32,Control_RunDLL ' + FileName),
CmdShow);
  except
  end;
end;
```

See also

RunControlPanel procedure.

12.4 String/Filename routines

Unit

acUtils

Declarations

```
procedure SplitStr(const SubStr, Str: String; var FirstPart, SecondPart:
  String;
  MainSide: TacSplitStrSide; LeaveSeparatorOn: TacSplitStrSides);
function LeftPart(const SubStr, Str: String): String;
function RightPart(const SubStr, Str: String): String;
function LeftPartR(const SubStr, Str: String): String;
function RightPartR(const SubStr, Str: String): String;
function PosR(const SubStr, Str: String): Integer;
procedure SplitFileNameAndParams(var FileName, Params: String);
function IsValidEmail(const Email: String): String;
function IncludeTrailingBackslash(const St: String): String;
function ExcludeTrailingBackslash(const St: String): String;
function IncludeLeadingChar(const St: String; Ch: Char): String;
function ExcludeLeadingChar(const St: String; Ch: Char): String;
function StringReplace(const S, OldPattern, NewPattern: String; Flags:
  TReplaceFlags): String;
function HasParamStr(const Param: String): Boolean;
function FullParamStr: String;
function CaseAnsiCompareText(const S1, S2: String; CaseSensitive:
  Boolean = True): Integer;
function URLEncode(const Str: String): String;
function URLDecode(const Str: String): String;
function StateNameToStateCode(const StateName: String): String;
function StateCodeToStateName(const StateCode: String): String;
procedure StrToStrList(const SourceStr: String; StrList: TStrings; const
  StrSeparator: String = #13#10);
function FileReadString(const Filename: String): String;
function FileWriteString(const Filename, St: String): Boolean;
```

12.4.1 SplitStr

Unit

acUtils

Declaration

```
type
```

© 1998-2006, UtilMind Solutions®

```
  TacSplitStrSide = (LEFT, RIGHT);
  TacSplitStrSides = set of TacSplitStrSide;

procedure SplitStr(const SubStr, Str: String;
            var LeftPart, RightPart: String;
            MainSide: TacSplitStrSide = LEFT;
            LeaveSeparatorOn: TacSplitStrSides = []);
```

Description

The **SplitStr** procedure separates the string specified in *Str* parameter on two parts (FirstPart and SecondPart parameters) . The separator character or string specified in *SubStr* parameter.

MainSide specifies the part which should contains an entire string (*Str*), if *SubStr* not found in *Str*. This value can be LEFT or RIGHT.

LeaveSeparatorOn specifies the separated parts where you would like to leave the separator string.

Example

```
SplitStr('@', 'email@address.com', FirstPart, SecondPart);
{ Results:
    FirstPart = email
    SecondPart = address.com
}

{ URL = https://secure.element5.com:443/register.html?productid=140005 }
SplitStr('://', URL, Protocol, HostName, RIGHT, []);
SplitStr('/', HostName, HostName, ObjectName, LEFT, [RIGHT]);
SplitStr(':', HostName, HostName, PortNumber, LEFT, []);
{ Results:
    Protocol = "https"
    HostName = "secure.element5.com"
    ObjectName = "/register.html?productid=140005"
    PortNumber = 443
}
```

Original code

```
procedure SplitStr(const SubStr, Str: String;
            var LeftPart, RightPart: String;
            MainSide: TacSplitStrSide {$IFDEF D4} = LEFT {$ENDIF};
            LeaveSeparatorOn: TacSplitStrSides {$IFDEF D4} = []
{$ENDIF});
var
  I: Integer;
begin
  I := Pos(SubStr, Str);
  if I <> 0 then
   begin
    LeftPart := Copy(Str, 1, I - 1);
    RightPart := Copy(Str, I + Length(SubStr), Length(Str));

    if LEFT in LeaveSeparatorOn then
      LeftPart := LeftPart + SubStr;

    if RIGHT in LeaveSeparatorOn then
      RightPart := SubStr + RightPart;
   end
  else // if SubStr not found
```

© 1998-2006, UtilMind Solutions®

```
    if MainSide = LEFT then
     begin
      LeftPart := Str;
      RightPart := '';
     end
    else
     begin
      LeftPart := '';
      RightPart := Str;
     end;
end;
```

See also
LeftPart, RightPart and SplitFileNameAndParams routines.

12.4.2 LeftPart

Unit
acUtils

Declaration
```
function LeftPart(const SubStr, Str: String): String;
```

Description
The **LeftPart** function finds the first occurence of any sub-string (*SubStr* parameter) in specified string (*Str* parameter) and returns the part of string (*SubStr*) BEFORE the separator (*Str*). If the *SubStr* can not be found in the *Str*, the function returns empty string.

If you would like to find LAST occurence of sub-string in specified string — use LeftPartR function.

Example
```
// Get first line from any text which contains line-breaks
FirstLine := LeftPart(#13#10, Str);
if FirstLine = '' then
  FirstLine := LeftPart(#10, Str);
```

Original code
```
function LeftPart(const SubStr, Str: String): String;
var
  I: Integer;
begin
  I := Pos(SubStr, Str);
  if I <> 0 then
    Result := Copy(Str, 1, I - 1)
  else
    Result := '';
end;
```

See also
LeftPart, RightPart, LeftPartR, RightPartR, PosR and SplitStr routines.

12.4.3 RightPart

Unit
acUtils

© 1998-2006, UtilMind Solutions®

Declaration

```
function RightPart(const SubStr, Str: String): String;
```

Description

The **LeftPart** function finds the first occurence of any sub-string (*SubStr* parameter) in specified string (*Str* parameter) and returns the part of string (*SubStr*) AFTER the separator (*Str*). If the *SubStr* can not be found in the *Str*, the function returns empty string.

If you would like to find LAST occurence of sub-string in specified string — use RightPartR function.

Example

```
// For example, we need to get the REALM NAME (between doublequotes)
RealmName := 'Basic realm="REALM NAME"';

// let's try...
RealmName := RightPart('="', RealmName); // result: REALM NAME"
// and get rid from next doublequote
RealmName := LeftPart('"', RealmName);   // result: REALM NAME
```

Original code

```
function RightPart(const SubStr, Str: String): String;
var
  I: Integer;
begin
  I := Pos(SubStr, Str);
  if I <> 0 then
    Result := Copy(Str, I + Length(SubStr), Length(Str))
  else
    Result := '';
end;
```

See also

LeftPart, RightPart, LeftPartR, RightPartR, PosR and SplitStr routines.

12.4.4 LeftPartR

Unit

acUtils

Declaration

```
function LeftPartR(const SubStr, Str: String): String;
```

Description

The **LeftPartR** function finds the LAST occurence of any sub-string (*SubStr* parameter) in specified string (*Str* parameter), and returns the part of string (*SubStr*) BEFORE the separator (*Str*). If the *SubStr* can not be found in the *Str*, the function returns empty string. Unlike LeftPart function, the LeftPartR starts searching the sub-string from the right side of string.

If you would like to find FIRST occurence of sub-string in specified string — use LeftPart function.

Example

```
ParentRegKey := RightPartR('\', 'Software\Microsoft\CurrentVersion');
// the result will be 'Software\Microsoft'
```

Original code

© 1998-2006, UtilMind Solutions®

```
function LeftPartR(const SubStr, Str: String): String;
var
  I: Integer;
begin
  I := PosR(SubStr, Str);
  if I <> 0 then
    Result := Copy(Str, 1, I - 1)
  else
    Result := '';
end;
```

See also
LeftPart, RightPart, LeftPartR, RightPartR, PosR and SplitStr routines.

12.4.5 RightPartR

Unit
acUtils

Declaration
```
function RightPartR(const SubStr, Str: String): String;
```

Description
The **RightPartR** function finds the LAST occurence of any sub-string (*SubStr* parameter) in specified string (*Str* parameter), and returns the part of string (*SubStr*) AFTER the separator (*Str*). If the *SubStr* can not be found in the *Str*, the function returns empty string. Unlike RightPart function, the RightPartR starts searching the sub-string from the right side of string.

If you would like to find FIRST occurence of sub-string in specified string — use RightPart function.

Example
```
SubKey := RightPartR('\', 'Software\Microsoft\CurrentVersion');
// the result will be 'CurrentVersion'
```

Original code
```
function RightPartR(const SubStr, Str: String): String;
var
  I: Integer;
begin
  I := PosR(SubStr, Str);
  if I <> 0 then
    Result := Copy(Str, I + Length(SubStr), Length(Str))
  else
    Result := '';
end;
```

See also
LeftPart, RightPart, LeftPartR, RightPartR, PosR and SplitStr routines.

12.4.6 PosR

Unit
acUtils

Declaration

© 1998-2006, UtilMind Solutions®

```
function PosR(const SubStr, Str: String): Integer;
```

Description

The PosR function searches for a substring in a string, from right side of the string, unlike standard Pos function.

PosR searches for the LAST occurence of *SubStr* within *Str* (beginning from right side of *Str*), and returns an integer value that is the index of the first character of *SubStr* within *Str*. If *SubStr* is not found, PosR returns zero.

Original code

```
function PosR(const SubStr, Str: String): Integer;
var
  I: Integer;
begin
  if (Length(SubStr) <> 0) and (Length(Str) >= Length(SubStr)) then
   for I := Length(Str) - Length(SubStr) downto 0 do
    if Copy(Str, I, Length(SubStr)) = SubStr then
     begin
      Result := I;
      Exit;
     end;
  Result := 0;
end;
```

See also

LeftPart, RightPart, LeftPartR, RightPartR and SplitStr routines.

12.4.7 SplitFileNameAndParams

Unit

acUtils

Declaration

```
procedure SplitFilenameAndParams(var FileName, Params: String);
```

Description

The **SplitFilenameAndParams** procedure divieds the separates the command file parameters from the file name.

Example

```
var
  Filename, Parameters: String;
begin
  Filename := 'C:\WINDOWS\Rundll32.exe
powrprof.dll,LoadCurrentPwrScheme';
  SplitFilenameAndParams(Filename, Parameters);
  { Results:
     Filename = "C:\WINDOWS\Rundll32.exe"
     Parameters = "powrprof.dll,LoadCurrentPwrScheme"
  }
```

See also

SplitStr procedure.

© 1998-2006, UtilMind Solutions®

12.4.8 IsValidEmail

Unit
acUtils

Declaration
```
function IsValidEmail(Email: String): Boolean;
```

Description
Ensures that email address (passed in Email parameter) is valid. Returns False if email is invalid (has incorrect form or contains invalid characters).

12.4.9 IncludeTrailingBackslash

Unit
acUtils

Declaration
```
function IncludeTrailingBackslash(St: String): String;
```

Description
The **IncludeTrailingBackslash** function puts a backslash '\' at the end of a string (i.e: file path) if it's not already there. For example result of `IncludeTrailingBackslash('c:\windows')` will be "c:\windows\". It's a typical function that gets rewritten by every programmer.

Note
This function used in the AppControls for Delphi 2/3/4 and BCB 1/3/4 only. In the Delphi 5 and later please use same routine from standard SysUtils unit.

Original code
```
function IncludeTrailingBackslash(const St: String): String;
begin
 if (St = '') or (St[Length(St)] <> '\') then
   Result := St + '\'
 else
   Result := St;
end;
```

12.4.10 ExcludeTrailingBackslash

Unit
acUtils

Declaration
```
function ExcludeTrailingBackslash(St: String): String;
```

Description
The **ExcludeTrailingBackslash** function removes a backslashes '\' from the end of string (i.e: file path). For example result of `ExcludeTrailingBackslash('c:\windows\')` will be "c:\windows".

Note
This function used in the AppControls for Delphi 2/3/4 and BCB 1/3/4 only. In the Delphi 5 and later please use same routine from standard SysUtils unit.

Original code

© 1998-2006, UtilMind Solutions®

```
function ExcludeTrailingBackslash(const St: String): String;
begin
  Result := St;
  while (Result <> '') and (Result[Length(Result)] = '\') do
   SetLength(Result, Length(Result) - 1);
end;
```

12.4.11 IncludeLeadingChar

Unit
acUtils

Declaration
```
function IncludeLeadingChar(St: String; Ch: Char): String;
```

Description
The **IncludeLeadingChar** function puts specified character at the beginning of a string if it's not already there. For example result of `IncludeTrailingChar('Software\Microsoft\CurrentVersion', '\')` will be "\Software\Microsoft\CurrentVersion". It's a typical function that gets rewritten by every programmer.

Original code
```
function IncludeLeadingChar(const St: String; Ch: Char): String;
begin
  if (St = '') or (St[1] <> Ch) then
    Result := Ch + St
  else
    Result := St;
end;
```

12.4.12 ExcludeLeadingChar

Unit
acUtils

Declaration
```
function ExcludeLeadingChar(St: String; Ch: Char): String;
```

Description
The **ExcludeLeadingChar** function removes specified character from the beginning of a string if it's exists there. For example result of `IncludeTrailingChar('\Software\Microsoft\CurrentVersion', '\')` will be "Software\Microsoft\CurrentVersion". It's a typical function that gets rewritten by every programmer.

Original code
```
function ExcludeLeadingChar(const St: String; Ch: Char): String;
begin
  Result := St;
  while (Result <> '') and (Result[1] = Ch) do
   Delete(Result, 1, 1);
end;
```

12.4.13 StringReplace

Unit
acUtils

© 1998-2006, UtilMind Solutions®

Declaration

```
type
  TReplaceFlags = set of (rfReplaceAll, rfIgnoreCase);

function StringReplace(const S, OldPattern, NewPattern: String; Flags:
  TReplaceFlags): String;
```

Description

The **StringReplace** function replaces occurrences of the substring specified by OldPattern with the substring specified by NewPattern parameter. StringReplace assumes that the source string, specified by S, may contain Multibyte characters.

If the Flags parameter does not include `rfReplaceAll`, StringReplace only replaces the first occurrence of OldPattern in S. Otherwise, all instances of OldPattern are replaced by NewPattern.

If the Flags parameter includes `rfIgnoreCase`, The comparison operation is case insensitive.

Note

This function used in the AppControls for Delphi 2/3 and BCB 1/3 only. In the Delphi 4 and later please use same routine from standard SysUtils unit.

12.4.14 HasParamStr

Unit

acUtils

Declaration

```
function HasParamStr(const Param: String): Boolean;
```

Description

The HasParamStr function used to check whether the application was executed with some command-line argument (*Param* parameter). It returns True if the parameter found within command line arguments, or False otherwise.

Example

```
// code from Loaded method of acTrayIcon component.
if HasParamStr('/minimized') then
  StartMinimized := True;
```

Original Code

```
function HasParamStr(const Param: String): Boolean;
var
  I: Integer;
begin
  if ParamCount <> 0 then
   for I := 1 to ParamCount do
    if AnsiLowerCase(ParamStr(I)) = AnsiLowerCase(Param) then
     begin
      Result := True;
      Exit;
     end;
  Result := False;
end;
```

© 1998-2006, UtilMind Solutions®

12.4.15 FullParamStr

Unit
acUtils

Declaration
```
function FullParamStr: String;
```

Description
The FullParamStr function retreives all parameters as one string.

For example, if your project was executed as
```
  c:\Documents And Settings\UserX\My Documents\My Project\Project1.exe
c:\Documents And Settings\UserX\My Documents\My Pictures\Picture.jpg
```

the function returns
```
  c:\Documents And Settings\UserX\My Documents\My Pictures\Picture.jpg
```

12.4.16 CaseAnsiCompareText

Unit
acUtils

Declaration
```
function CaseAnsiCompareText(const S1, S2: String; CaseSensitive:
Boolean = True): Integer;
```

Description
The CaseAnsiCompareText compares *S1* to *S2*, with case sensitivity if *CaseSensitive* parameter is True. The compare operation is controlled by the current Windows locale. CaseAnsiCompareText returns a value less than 0 if S1 < S2, a value greater than 0 if S1 > S2, and returns 0 if S1 = S2.

Original Code
```
function CaseAnsiCompareText(const S1, S2: String; CaseSensitive:
Boolean {$IFDEF D4} = True {$ENDIF}): Integer;
begin
  if CaseSensitive then
    Result := AnsiCompareStr(S1, S2)
  else
    Result := AnsiCompareText(S1, S2);
end;
```

12.4.17 URLEncode

Unit
acUtils

Declaration
```
function URLEncode(const Str: String): String;
```

Description
The URLEncode function turns all unsafe characters like '*', '#', '%', '<', '>', '+', ' ' into their escape codes. So if the *Str* passed as parameter was "Hello World", the output string will be "Hello%20World".

See also

© 1998-2006, UtilMind Solutions®

URLDecode function.

12.4.18 URLDecode

Unit
acUtils

Declaration
```
function URLDecode(const Str: String): String;
```

Description
The URLEncode function turns all escape codes like like %20 or %3C into normal characters. So if the *Str* passed as parameter was `"Hello%20World"`, the output string will be `"Hello World"`.

See also
URLEncode function.

12.4.19 StateNameToStateCode

Unit
acUtils

Declaration
```
function StateNameToStateCode(const StateName: String): String;
```

Description
Function returns the the 2-letter state code by state name (for USA states only).

For example, result of call `StateNameToStateCode('Maryland')` will be **`MD`**.

See also
StateCodeToStateName function.

12.4.20 StateCodeToStateName

Unit
acUtils

Declaration
```
function StateNameToStateCode(const StateName: String): String;
```

Description
Function returns the state name by 2-letter state code (for USA states only)

For example, result of call `StateCodeToStateName('MD')` will be **`Maryland`**.

See also
StateNameToStateCode function.

12.4.21 StrToStrList

Unit
acUtils

Declaration
```
procedure StrToStrList(const SourceStr: String; StrList: TStrings; const
```

© 1998-2006, UtilMind Solutions®

```
  StrSeparator: String = #13#10);
```

Description
The StrToStrList procedure parses the string specified by *SourceStr* parameter, separates it by parts, and puts its parts to the string list.

12.4.22 FileReadString

Unit
acUtils

Declaration
```
function FileReadString(const Filename: String): String;
```

Description
The **FileReadString** function reads the file and returns its content as plain text string.

See also
HTTPReadString function from acHTTP unit.

12.4.23 FileWriteString

Unit
acUtils

Declaration
```
function FileWriteString(const Filename, St: String): Boolean;
```

Description
The **FileWriteString** creates the file (*FileName* parameter) and writes to it the specified string (*St* parameter).

Function returns True if succeed or False otherwise (i.e: invalid path or file write error).

See also
FileReadString function.

12.5 MySQL String routines

Unit
acUtils

Declarations
```
{ Converts string in SQL date/time format (YYYY-MM-DD HH:MM:SS) to
  TDateTime variable }
function SQLDateTimeToDateTime(Value: String): TDateTime;
{ Converts TDateTime value to SQL date/time format (YYYY-MM-DD HH:MM:SS)
  }
function DateTimeToSQLDateTime(const Value: TDateTime): String;
{ Converts TDateTime value to SQL date format (YYYY-MM-DD) }
function DateTimeToSQLDate(const Value: TDateTime): String;
{ Converts TDateTime value to SQL date format (HH:MM:SS) }
function DateTimeToSQLTime(const Value: TDateTime): String;
{ Escapes string for MySQL query and puts it to quotes }
function ESC(const Str: String): String;
```

© 1998-2006, UtilMind Solutions®

12.5.1 ESC

Unit
acUtils

Declaration
```
function ESC(const Str: String): String;
```

Description
The ESC function "escapes" the string, makes the string safe for call as SQL string parameter. It adds the slash character before all quotes and doublequotes, converts #13 character to \r and #10 to \n.

Original code
```
function ESC(const Str: String): String;
var
  I, RI: Integer;

  procedure AddToResult(Ch: Char);
  begin
    Result[RI] := '\';
    inc(RI);
    Result[RI] := Ch;
    inc(RI);
  end;

begin
  SetLength(Result, Length(Str) shl 1 + 3); // (length of original
string) * 2 + 1
  Result[1] := '''';
  RI := 2;

  for I := 1 to Length(Str) do
    case Str[I] of
      #0:   AddToResult('0');
      #13:  AddToResult('r');
      #10:  AddToResult('n');
      '\', '''', '"': AddToResult(Str[I]);
      else
        Result[RI] := Str[I];
        Inc(RI);
     end;

  Result[RI] := '''';
  SetLength(Result, RI);
end;
```

12.5.2 SQLDateTimeToDateTime

Unit
acUtils

Declaration
```
function SQLDateTimeToDateTime(Value: String): TDateTime;
```

Description
The SQLDateTimeToDateTime function converts the string in SQL date/time format to the

© 1998-2006, UtilMind Solutions®

TDateTime variable.

See also
SQLDateTimeToDateTime, DateTimeToSQLDateTime, DateTimeToSQLDate and DateTimeToSQLTime functions.

12.5.3 DateTimeToSQLDateTime

Unit
acUtils

Declaration
```
function DateTimeToSQLDateTime(const Value: TDateTime): String;
```

Description
Converts TDateTime value to SQL date/time format (YYYY-MM-DD HH:MM:SS).

See also
SQLDateTimeToDateTime, DateTimeToSQLDateTime, DateTimeToSQLDate and DateTimeToSQLTime functions.

12.5.4 DateTimeToSQLDate

Unit
acUtils

Declaration
```
function DateTimeToSQLDate(const Value: TDateTime): String;
```

Description
Converts TDateTime value to SQL date format (YYYY-MM-DD).

See also
SQLDateTimeToDateTime, DateTimeToSQLDateTime, DateTimeToSQLDate and DateTimeToSQLTime functions.

12.5.5 DateTimeToSQLTime

Unit
acUtils

Declaration
```
function DateTimeToSQLTime(const Value: TDateTime): String;
```

Description
Converts TDateTime value to SQL date format (HH:MM:SS).

See also
SQLDateTimeToDateTime, DateTimeToSQLDateTime, DateTimeToSQLDate and DateTimeToSQLTime functions.

12.6 Date/Time routines

Unit
acUtils

© 1998-2006, UtilMind Solutions®

Declarations

```
function StrToDateTimeDef(const S: String;
  DefDateTime: TDateTime): TDateTime;
function IsLeapYear(AYear: Integer): Boolean;
function DaysPerMonth(AYear: Integer; AMonth: Word): Byte;
function GMTToLocalTime: TDateTime;
function LocalTimeToGMT: TDateTime;
function GMT: TDateTime;
function InternetTimeToDateTime(InternetTime: String): TDateTime;
function DateTimeToFileTime(Time: TDateTime): TFileTime;
function FileTimeToDateTime(Time: TFileTime): TDateTime;
function SetFileTimes(const FileName: String;
  Created, Modified, Accessed: TDateTime): Boolean;
function SameDateTime(const DateTime1, DateTime2: TDateTime): Boolean;
```

12.6.1 StrToDateTimeDef

Unit

acUtils

Declaration

```
function StrToDateTimeDef(const S: String;
  DefDateTime: TDateTime): TDateTime;
```

Description

The StrToDateTimeDef function converts a string to a TDateTime value. If S does not represent a valid DateTime, StrToDateTimeDef returns the DateTime passed in DefDateTime parameter.

Used to convert TDateTime value to String without exception.

See also

InternetTimeToDateTime function.

12.6.2 IsLeapYear

Unit

acUtils

Declaration

```
function IsLeapYear(AYear: Integer): Boolean;
```

Description

The IsLeapYear function indicates whether a specified year is a leap year.

Remarks

This function used in the AppControls for Delphi 2 only. In the Delphi 3 and later this obsolete, because you can find same function in standard SysUtils unit.

12.6.3 DaysPerMonth

Unit

acUtils

Declaration

```
function DaysPerMonth(AYear: Integer; AMonth: Word): Byte;
```

© 1998-2006, UtilMind Solutions®

Description
The DaysPerMonth returns the number of days in specified month of year.

Original code
```
function DaysPerMonth(AYear: Integer; AMonth: Word): Byte;
const
  DaysInMonth: Array[1..12] of Integer = (31, 28, 31, 30, 31, 30, 31,
31, 30, 31, 30, 31);
begin
  Result := DaysInMonth[AMonth];
  if (AMonth = 2) and IsLeapYear(AYear) then Inc(Result); // leap-year
Feb is special
end;
```

12.6.4 GMTToLocalTime

Unit
acUtils

Declaration
```
function GMTToLocalTime(GMTTime: TDateTime): TDateTime;
```

Description
The GMTToLocalTime function adjusts the GMT (Greenwich Mean Time) time value to the time of local time zone.

See also
LocalTimeToGMT and GMT functions.

12.6.5 LocalTimeToGMT

Unit
acUtils

Declaration
```
function LocalTimeToGMT(LocalTime: TDateTime): TDateTime;
```

Description
The LocalTimeToGMT function converts local time to the time at Greenwich time zone.

See also
GMTToLocalTime and GMT functions.

12.6.6 GMT

Unit
acUtils

Declaration
```
function GMT: TDateTime;
```

Description
The GMT function returns the *current* time at Greenwich time zone. This is the same as Now routine but adjusted to GMT accordingly local time zone and daylight bias.

See also

© 1998-2006, UtilMind Solutions®

GMTToLocalTime and LocalTimeToGMT functions.

12.6.7 InternetTimeToDateTime

Unit
acUtils

Declaration
```
function InternetTimeToDateTime(InternetTime: String): TDateTime;
```

Description
The InternetTimeToDateTime function converts the string representation of date time to the TDateTime variable.

The *InternetTime* parameter is the string representation of the date/time returned by Internet servers in following maner:
```
day_of_week, day month year hour:min:sec offset
```
for example:
```
Sat, 19 Oct 2002 19:45:16 GMT
```
, or
```
Fri, 23 Feb 1999 6:02:37 GMT+2
```

The function will convert this string to the local time, accordingly to GMT offset (specified in the string) between client and server, and the daylight bias on local machine.

This function used in acHTTP component, when it preparing the parameters for the OnHeaderInfo event handler. It convert the date values from the header of HTTP response to the Delphi-friendly TDateTime values.

See also
StrToDateTimeDef function;
GMTToLocalTime and LocalTimeToGMT functions.

12.6.8 DateTimeToFileTime

Unit
acUtils

Declaration
```
function DateTimeToFileTime(Time: TDateTime): TFileTime;
```

Description
The DateTimeToFileTime function converts the Delphi-friendly TDateTime variable to the TFileTime record, which used in Windows file system to determinate the file creation and modification times.

The TFileTime (FILETIME) structure is a 64-bit value representing the number of 100-nanosecond intervals since January 1, 1601.

```
typedef struct _FILETIME { // ft
    DWORD dwLowDateTime;
    DWORD dwHighDateTime;
} FILETIME;
```

See also
FileTimeToDateTime and SetFileTimes routines.

© 1998-2006, UtilMind Solutions®

12.6.9 FileTimeToDateTime

Unit
acUtils

Declaration
```
function FileTimeToDateTime(Time: TFileTime): TDateTime;
```

Description
The FileTimeToDateTime function converts the TFileTime record (which used in Windows file system to determinate the file creation and modification times) to a Delphi-friendly TDateTime variable.

See also
DateTimeToFileTime and SetFileTimes routines.

12.6.10 SetFileTimes

Unit
acUtils

Declaration
```
function SetFileTimes(const FileName: String;
  Created, Modified, Accessed: TDateTime): Boolean;
```

Description
The SetFileTimes procedure used to set the time labels for the file specified in FileName parameter. You can specify the time at which the file was last modified, accessed and when the file has been created. Function returns True if succeed or False otherwise.

If you don't want to modify some of these time values, just set the parameter to -1. For example, you would like to change the time at which the file was created but leave the time of last access and modification untouched. Then simply specify the Created parameter and set Modified and Accessed parameters to -1, as shown in following example:
```
SetFileTimes('c:\autoexec.bat', Now, -1, -1);
```

See also
DateTimeToFileTime and FileTimeToDateTime routines.

12.6.11 SameDateTime

Unit
acUtils

Declaration
```
function SameDateTime(const DateTime1, DateTime2: TDateTime): Boolean;
```

Description
The SameDateTime function compares 2 TDateTime values with precision of 1 millisecond and returns True if dates/times are equal or False otherwise.

12.7 System Information

Unit
acUtils

© 1998-2006, UtilMind Solutions®

Declarations

```
function GetOS: TacOSVersion;
function IsNT: Boolean; // Win95/98/ME or NT/2000 ?
function IsXP: Boolean; // WinXP-compatible or lower ?
function IsWin2k: Boolean; // Win2000 and higher or not ?
function IsXP: Boolean; // WinXP-compatible or lower ?
function Is256ColorTrayIconAllowed: Boolean;
function GetCPUSpeed: Integer; // CPU Frequency
function IsOnline: Boolean; // Internet status - online/offline
function IsScreenSaverRunning: Boolean; // Is the screensaver running
function GetTempDir: String; // returns path to Temp directory
function GetSystemDir: String; // path to System directory
function GetWindowsDir: String; // path to Windows directory
function GetEnvironmentString(const St: String): String; // returns
     value of the environment variable
function GetSystemIconIndex(const FileName: String): Integer; // index
     of icon in system image list by FileName
```

Description

Provides system and network information

See also

acSystemInfo component.

12.7.1 GetOS

Unit

acUtils

Declaration

```
type
  TacOSVersion = (osUnknown, os95, os95OSR2, os98, os98SE, osNT3, osNT4,
os2K, osME);

function GetOS: TacOSVersion;
```

Description

The **GetOS** function determines the type of operation system on user's PC and returns according value.

GetOS is same as OSVersion property of acSystemInfo component.

See also

IsNT and IsWin2k functions; acSystemInfo component.

12.7.2 IsNT

Unit

acUtils

Declaration

```
function IsNT: Boolean; // returns True if Windows is NT/2000
```

Description

The **IsNT** function determines whether user's OS is Windows NT or compatible system (ie Windows 2000). IsNT returns True if executed on WinNT/Win2000 machine and False if executed on

© 1998-2006, UtilMind Solutions®

Win95/Win98/WinME machine.

IsNT function returns same value as WinNT property of acSystemInfo component.

See also
IsWin2k, IsXP and GetOS routines; acSystemInfo component.

12.7.3 IsWin2k

Unit
acUtils

Declaration
```
function IsWin2k: Boolean; // returns True if Win2000 or higher
```

Description
The **IsWin2k** function determines whether the current OS is Windows 2000. Funtion returns True if system is Windows 2000 or higher, and False if OS is Win95/98/NT4/ME (anything lower than Win2000).

See also
IsNT, IsXP and GetOS routines; acSystemInfo component.

12.7.4 IsXP

Unit
acUtils

Declaration
```
function IsWinXP: Boolean; // returns True if WinXP or higher
```

Description
The **IsWinXP** function determines whether the current OS is Windows XP. Funtion returns True if system is Windows 2000 or higher, and False if version number is lower than WinXP (lower than v5.1).

See also
IsNT, IsWin2k and GetOS functions; acSystemInfo component.
Windows XP themes functions.

12.7.5 Is256ColorTrayIconAllowed

Unit
acUtils

Declaration
```
function Is256ColorTrayIconAllowed: Boolean; // returns True if WinXP or
higher
```

Description
The **Is256ColorTrayIconAllowed** function helps to determinate whether current OS allows to show 256-color icons in the system tray area.

Note
As you may know, operating systems such as Win95, Win98, WinNT and Win2000 can only display 16 colors when you write icons to the system tray, however WinME and WinXP can display

© 1998-2006, UtilMind Solutions®

256 color icons.

Original code

```
function Is256ColorTrayIconAllowed: Boolean;
begin
  Result := not (GetOS in [os95, os95OSR2, os98, os98SE, osNT3, osNT4,
os2K]);
end;
```

See also

Icon, Icon16 and ImageList properties of acTrayIcon component.
IsNT, IsWin2k and GetOS functions; acSystemInfo component.

12.7.6 GetCPUSpeed

Unit

acUtils

Declaration

```
function GetCPUSpeed: Integer; // in megaherzes
```

Description

The **GetCPUSpeed** function returns the rounded CPU frequency value for first PC processor. Return value is CPU speed in Mhz.

GetCPUSpeed returns same value as CPUFrequency property of acSystemInfo component.

See also

acSystemInfo component.

12.7.7 IsScreenSaverRunning

Unit

acUtils

Declaration

```
function IsScreenSaverRunning: Boolean;
```

Description

The **IsScreenSaverRunning** function determines is the screensaver currently running and returns True if screensaver running and False otherwise.

See also

StartScreenSaver function.

12.7.8 GetTempDir

Unit

acUtils

Declaration

```
function GetTempDir: String;
```

Description

The **GetTempDir** function returns path to the temporary directory (i.e.: "C:\WINDOWS\TEMP").

© 1998-2006, UtilMind Solutions®

See also
GetSystemDir and GetWindowsDir functions;
acSystemInfo component.

12.7.9 GetSystemDir

Unit
acUtils

Declaration
```
function GetSystemDir: String;
```

Description
The **GetSystemDir** function returns path to the system directory (i.e.: "C:\WINDOWS\SYSTEM" on Windows 95/98 or "C:\WINDOWS\SYSTEM32" on Windows NT/2000).

See also
GetTempDir and GetWindowsDir functions;
acSystemInfo component.

12.7.10 GetWindowsDir

Unit
acUtils

Declaration
```
function GetWindowsDir: String;
```

Description
The **GetWindowsDir** function returns path to the Windows directory (i.e.: "C:\WINDOWS" on Windows 95/98 or "C:\WINDOWS" on Windows NT/2000).

See also
GetTempDir and GetSystemDir functions;
acSystemInfo component.

12.7.11 GetEnvironmentString

Unit
acUtils

Declaration
```
function GetEnvironmentString(const St: String): String;
```

Description
The **GetEnvironmentString** retrieves the string value of the specified environment variable of the Windows, from the environment block of the calling process. The St parameter identifies the variable.

Example
```
{ returns the searching pathes (SET PATH="...") }
SearchPathes := GetEnvironmentString('PATH');
```

Original code
```
function GetEnvironmentString(const St: String): String;
var Len: Integer;
```

© 1998-2006, UtilMind Solutions®

```
begin
  Result := '';
  Len := GetEnvironmentVariable(PChar(St), nil, 0);
  if Len = 0 then Exit;
  SetLength(Result, Len - 1);
  GetEnvironmentVariable(PChar(St), PChar(Result), Len);
end;
```

See also
SystemPathes property of the acSystemInfo component.

12.7.12 GetSystemIconIndex

Unit
acUtils

Declaration
```
function GetSystemIconIndex(const FileName: String): Integer;
```

Description
The **GetSystemIconIndex** retrieves the icon index of sytem image list, which corresponds to specified file.

Note
Function returns -1 if specified file was not found.

See also
acSystemImageList component.

12.8 Sound routines

Unit
acUtils

Declarations
```
// PC Speaker routines
procedure Sound(Hz: Word);
procedure NoSound;
procedure Delay(MSec: Word);

// Default system sound events
procedure PlaySystemSound(SndType: TacSoundType;
   Asynchronous, Looped: Boolean);
procedure StopSystemSound;
```

Description
Plays the system sound from the internal PC speaker. The Sound, NoSound and Delay procedures represents old Turbo Pascal routines of the same name.

12.8.1 Sound

Unit
acUtils

Declaration
```
procedure Sound(Hz: Word);
```

© 1998-2006, UtilMind Solutions®

Description
The **Sound** procedure starts the internal speaker. Hz specifies the frequency of the emitted sound in hertz. The speaker continues until explicitly turned off by a call to NoSound.

Example
The following program fragment emits a 440-hertz tone for half a second:

```
uses acUtils;
begin
  Sound(440);          { Beep }
  Delay(500);          { For 500 ms }
  NoSound;             { Relief! }
end.
```

See also
NoSound and Delay procedures.

12.8.2 NoSound

Unit
acUtils

Declaration

```
procedure NoSound;
```

Description
The NoSound procedure turns off the computer's internal speaker.

Example
The following program fragment emits a 440-hertz tone for half a second:

```
uses acUtils;
begin
  Sound(440);          { Beep }
  Delay(500);          { For 500 ms }
  NoSound;             { Relief! }
end.
```

See also
Sound and Delay procedures.

12.8.3 Delay

Unit
acUtils

Declaration

```
procedure Delay(MSec: Integer);
```

Description
The **Delay** procedure delays a specified number of milliseconds. MSecs parameter specifies the number of milliseconds to wait.

The Delay procedure is the same as the **Sleep** API routine and used for compatibility with old Pascal code.

See also
Sound and NoSound procedures.

© 1998-2006, UtilMind Solutions®

12.8.4 PlaySystemSound

Unit
acUtils

Declaration
```
type
  TacSystemSoundType = (stCustom, stAsterisk, stCloseProgram,
                        stCriticalStop, stDefaultSound,
                        stExclamation, stExitWindows,
                        stMaximize, stMenuCommand, stMenuPopup,
                        stMinimize, stNewMailNotification,
                        stOpenProgram, stProgramError, stQuestion,
                        stRestoreDown, stRestoreUp, stStartWindows);

procedure PlaySystemSound(SndType: TacSystemSoundType;
   Asynchronous, Looped: Boolean);
```

Description
The PlaySystemSound procedure can be used to play default system sound event, specified in "Sounds" applet of Control Panel.

Parameters

SndType	describes the action type, to play Wav-files associated with this action.
Asynchronous	specifies whether you would like to play sound asynchronously with main application thread. Set it to True if you don't want to suspend main application thread during playing of the sound.
Looped	specifies whether you would like to play the sound continuously. Set it to False to play the sound only once. Otherwise, if you would like to replay the sound continuously — use StopSystemSound to terminate the sound.

Note
Don't use *stCustom* SndType. This type can be only used in acWavPlayer component, for playing custom Wav-file uploaded to form. Please see documentation on acWavPlayer component for more information.

See also
StopSystemSound procedure and acWavPlayer component.

12.8.5 StopSystemSound

Unit
acUtils

Declaration
```
procedure StopSystemSound;
```

Description
The StopSystemSound procedure stops ALL sounds that currently are playing through default system sound device.

See also
PlaySystemSound procedure and Stop method of acWavPlayer component.

© 1998-2006, UtilMind Solutions®

12.9 Easy Registry utilities

Unit
acUtils

Declarations
```
{ Whether the value exists in Registry }
function IsRegValueExists(const KeyName, ValueName: String; RootKey:
hKey = HKEY_CURRENT_USER): Boolean;

{ Read string from Registry. Returns empty string if failed }
function RegReadStr(const KeyName, ValueName: String; RootKey: hKey =
HKEY_CURRENT_USER): String;

{ Write string to Registry }
procedure RegWriteStr(const KeyName, ValueName, Value: String; RootKey:
hKey = HKEY_CURRENT_USER);

{ Read integer value from Registry. You can specify default value if the
key doesn't exists.}
function RegReadInt(const KeyName, ValueName: String; DefaultValue:
Integer = 0; RootKey: hKey = HKEY_CURRENT_USER): LongInt;

{ Write integer value to Registry }
procedure RegWriteInt(const KeyName, ValueName: String; Value: Integer;
RootKey: hKey = HKEY_CURRENT_USER);

{ Deletes value from Registry }
procedure RegEraseValue(const KeyName, ValueName: String; RootKey: hKey
= HKEY_CURRENT_USER);

{ Deletes the registry key and all its subkeys }
procedure RegEraseKey(const KeyName: String; const RootKey: hKey =
HKEY_CURRENT_USER; DeleteAllSubkeys: Boolean = True);

{ Reads Font settings from registry }
procedure RegReadFont(const KeyName, ValueName: String; Font: TFont;
RootKey: hKey = HKEY_CURRENT_USER);

{ Writes Font settings to registry }
procedure RegWriteFont(const KeyName, ValueName: String; Font: TFont;
RootKey: hKey = HKEY_CURRENT_USER);
```

Description
These routines gives you some easy and simple functions to work with system registry without hard Registry unit.

12.9.1 IsRegValueExists

Unit
acUtils

Declaration
```
function IsRegValueExists(KeyName, ValueName: String; RootKey: hKey):
Boolean;
```

© 1998-2006, UtilMind Solutions®

Description
The **IsRegValueExists** function returns True if the value, specified in *ValueName* parameter, is exists under registry key, specified by *RootKey* parameter.

Original code
```
function IsRegValueExists(const KeyName, ValueName: String;
  RootKey: hKey {$IFDEF D4}=HKEY_CURRENT_USER{$ENDIF}): Boolean;
begin
  Result := False;
  if RegOpen(RootKey, KeyName, False) then
   begin
    Result := RegQueryValueEx(RootKey, PChar(ValueName), nil, nil, nil,
nil) = ERROR_SUCCESS;
    RegCloseKey(RootKey);
   end;
end;
```

12.9.2 RegReadStr

Unit
acUtils

Declaration
```
function RegReadStr(KeyName, ValueName: String; RootKey: hKey): String;
```

Description
The **RegReadStr** function reads and returns the string from the registry, under specified *KeyName* and *ValueName*.

Remarks

If specified *KeyName* or *ValueName* is not exists in the registry, function will return empty string.

Example *(this code retrieves the location of Desktop folder for current user)*
```
uses RegStr, acUtils;
// ---
Result := RegReadStr(REGSTR_PATH_EXPLORER + '\Shell Folders',
  'Desktop', HKEY_CURRENT_USER);
```

See also
RegWriteStr procedure.

12.9.3 RegWriteStr

Unit
acUtils

Declaration
```
procedure RegWriteStr(KeyName, ValueName, Value: String; RootKey: hKey);
```

Description
The **RegWriteStr** procedure writes a string (*Value* parameter) to the registry, under specified *KeyName* and *ValueName*.

If specified *KeyName* is not exists in the registry, the routine will create it anyway.

See also

© 1998-2006, UtilMind Solutions®

RegReadStr function.

12.9.4 RegReadInt

Unit
acUtils

Declaration
```
function RegReadInt(KeyName, ValueName: String; DefaultValue: Integer;
RootKey: hKey): LongInt;
```

Description
The **RegReadInt** function reads and returns the integer value from the registry, under specified *KeyName* and *ValueName*. The *DefaultValue* parameter used to specify the return value if the *KeyName* or *ValueName* is not exists in the registry.

See also
RegWriteInt procedure.

12.9.5 RegWriteInt

Unit
acUtils

Declaration
```
procedure RegReadInt(KeyName, ValueName: String; Value: Integer;
RootKey: hKey);
```

Description
The **RegWriteInt** procedure writes the integer *Value* to the registry, under specified *KeyName* and *ValueName*.

If specified *KeyName* is not exists in the registry, the routine will create it anyway.

See also
RegReadInt function.

12.9.6 RegEraseValue

Unit
acUtils

Declaration
```
procedure RegEraseValue(KeyName, ValueName: String; RootKey: hKey);
```

Description
The **RegEraseValue** procedure deletes the *ValueName*, under specified *KeyName*.

See also
RegEraseKey procedure.

12.9.7 RegEraseKey

Unit
acUtils

© 1998-2006, UtilMind Solutions®

Declaration
```
procedure RegEraseKey(const KeyName: String; RootKey: hKey;
DeleteAllSubkeys: Boolean = True);
```

Description
The **RegEraseKey** procedure deletes the *KeyName*, and all its sub-keys, if *DeleteAllSubkeys* parameter is True.

See also
RegEraseValue procedure.

12.9.8 RegReadFont

Unit
acUtils

Declaration
```
procedure RegReadFont(KeyName, ValueName: String; Font: TFont; RootKey:
hKey);
```

Description
The **RegReadFont** procedure reads all properties of TFont object from the registry, under specified *KeyName* and *ValueName*.

See also
RegWriteFont procedure.

12.9.9 RegWriteFont

Unit
acUtils

Declaration
```
procedure RegWriteFont(KeyName, ValueName: String; Font: TFont; RootKey:
hKey);
```

Description
The **RegWriteFont** procedure writes all properties of TFont object to the registry, under specified *KeyName* and *ValueName*.

See also
RegReadFont procedure.

12.10 Internet utilities

Unit
acInet

Declarations
```
function IsOnline: Boolean;
function GetLocalHostName: String;
function GetLocalIP: String;
function IPToStr(IP: Integer): String;
function StrToIP(Host: String): Integer;
function ParseURL(URL: String; var Protocol, HostName, URLPath,
  Username, Password, ExtraInfo: String; var Port: Word): Boolean;
```

© 1998-2006, UtilMind Solutions®

```
function URLToHostName(URL: String): String;
function URLToFileName(URL: String): String;
function SockConnect(var SOCK: Integer; Host: String;
  Port: Integer): Boolean;
procedure SockDisconnect(var SOCK: Integer);
```

Description
Various Internet funtions.

12.10.1 IsOnline

Unit
acInet

Declaration
```
function IsOnline: Boolean;
```

Description
Determines whether user's PC connected to the Internet and returns current Internet status — True if computer connected to the Internet or False otherwise.

Remark
This function is useless on machines with Ethernet cable connection. IsOnline always returns True (online), even if user cut cable off. Unfortunately there is no convinient way to determinate whether computer is online. However, for sure you may try to ping some host or something. This function mostly good for Dial-up modem connections only.

See also
acSystemInfo component.

12.10.2 GetLocalHostName

Unit
acInet

Declaration
```
function GetLocalHostName: String;
```

Description
Retrieves the name of computer. (See "Control Panel", "Network" applet, "Identification" tab)

See also
GetLocalIP function.

12.10.3 GetLocalIP

Unit
acInet

Declaration
```
function GetLocalIP: String;
```

Description
Returns the string which represents IP address of current machine.

See also

© 1998-2006, UtilMind Solutions®

GetLocalHostName and StrToIP functions.

12.10.4 IPToStr

Unit
aclnet

Declaration
```
function IPToStr(IP: Integer): String;
```

Description
Converts integer value to the string value which represents IP address.

See also
StrToIP function.

12.10.5 StrToIP

Unit
aclnet

Declaration
```
function StrToIP(Host: String): Integer;
```

Description
Converts string which represents IP address to integer value.

See also
IPToStr function.

12.10.6 ParseURL

Unit
aclnet

Declaration
```
function ParseURL(URL: String; var Protocol, HostName, URLPath,
  Username, Password, ExtraInfo: String; var Port: Word): Boolean;
```

Description
The ParseURL procedure divides the *URL* string (in) to *Protocol*, *HostName*, *URLPath* (path and file name), *Username*, *Password*, *ExtraInfo* (GET query) and *PortNumber* values.

The URL location should be specified in following form:
```
[protocol://][username:password@]hostname[[:port]/urlpath][?extra]
```
examples:
```
http://www.appcontrols.com/download/diskcontrols_trial.exe
utilmind.com:80
https://secure.element5.com/register.html?productid=140005
file://c:/windows/desktop/page.html
http://hello:world@www.utilmind.com/locked/topsecret.zip
```

If port was not specified in the URL line, the *PortNumber* value will be recognized automatically. Defaults ports is 80 for HTTP protocol (or if protocol was not specified in the URL), 443 for HTTPS, 21 is default for FTP and 70 for Gopher.

© 1998-2006, UtilMind Solutions®

Example 1

```
ParseURL('https://secure.element5.com/register.html?productid=140005',
  Protocol, HostName, URLPath, ExtraInfo, PortNumber);
{ Results:
    Protocol = "https"
    HostName = "secure.element5.com"
    URLPath = "/register.html"
    ExtraInfo = "?productid=140005"
    PortNumber = 443 (default port for secure (SSL) connection )
}
```

Example 2

```
ParseURL('utilmind.com', Protocol, HostName, URLPath, ExtraInfo,
  PortNumber);
{ Results:
    Protocol = "http"
    HostName = "utilmind.com"
    URLPath = ""
    ExtraInfo = ""
    PortNumber = 80 (default port for HTTP protocol)
}
```

Example 3

```
ParseURL('hello:world@www.utilmind.com/cgi-bin/locked/test.cgi?test',
  Protocol, HostName, URLPath, ExtraInfo, PortNumber);
{ Results:
    Protocol = "http"
    HostName = "www.utilmind.com"
    URLPath = "/cgi-bin/locked/test.cgi"
    ExtraInfo = "?test"
    PortNumber = 80
}
```

Example 4

```
ParseURL('file://c:/windows/desktop/page.html', Protocol, HostName,
  URLPath, ExtraInfo, PortNumber);
{ Results:
    Protocol = "file"
    HostName = ""
    URLPath = "c:\windows\desktop\page.html"
    ExtraInfo = ""
    PortNumber = 0 (local file)
}
```

See also

acHTTP component.

12.10.7 URLToHostName

Unit

acInet

Declaration

```
function URLToHostName(URL: String): String;
```

Description

Retrieves the host name from specified URL.

© 1998-2006, UtilMind Solutions®

Example

```
Result :=
URLToHostName('http://www.appcontrols.com/appcontrols/overview.html');
// Result = 'www.appcontrols.com';
```

See also

URLToFileName and ParseURL functions.

12.10.8 URLToFileName

Unit

aclnet

Declaration

```
function URLToFileName(URL: String): String;
```

Description

Retrieves the file name from specified URL..

Example

```
Result :=
URLToHostName('http://www.appcontrols.com/appcontrols/overview.html');
// Result = 'overview.html';
```

See also

URLToHostName and ParseURL functions.

12.10.9 SockConnect

Unit

aclnet

Declaration

```
function SockConnect(var SOCK: Integer; const Host: String; const Port:
  Integer): Boolean;
```

Description

Connects to specified *Host* and *Port* and set the socket identifier to *SOCK* parameter. Returns True if connected successfully or False otherwise.

See also

SockDisconnect procedure.

12.10.10 SockDisconnect

Unit

aclnet

Declaration

```
procedure SockDisconnect(var SOCK: Integer);
```

Description

Disconnects from specified socket identifer and set it to 0.

See also

SockConnect function.

© 1998-2006, UtilMind Solutions®

12.11 Miscellaneous

Unit
acUtils

Declaration
```
function IsDirectoryExists(DirName: String): Boolean;
function IsDirectoryEmpty(DirName: String): Boolean;
function ExtractFileSize(FileName: String): String;
function FileSizeToStr(FileName: String;
                       [Bytes, Kb, Mb: String]): String;
function IsRegValueExists(KeyName, ValueName: String;
    RootKey: hKey = HKEY_CURRENT_USER): Boolean;
function GetKeyName(lParam: LongInt): String;
function GetShiftState: TShiftState;
function GetWorkArea: TRect;
function GetWindowTitle(Wnd: hWnd): String;
function PercentsOf(Entire, Part: Extended): Byte;
procedure StartScreenSaver;
function LoadResourceToStream(Instance: hInst; ResName, ResType: PChar;
    Stream: TStream): Boolean;
function StreamToString(Stream: TStream): String;
procedure PutStringToStream(Str: String; Stream: TStream);
procedure MsgBox(Msg: String; Flags: LongInt);
procedure InfoBox(Msg: String);
procedure ErrorBox(Msg: String);
function XORStr(Str: String; XORByte: Byte): String;
procedure ShutdownApplication;
procedure HideFromTaskList;
function IsMetricSystem: Boolean;
function IsDebug: Boolean;
procedure Wait(var WaitHandle: THandle; Timeout: DWord = 0);
procedure StopWait(var WaitHandle: THandle);
procedure LogToFile(const FileName, Str: String; AddDateTime: Boolean =
True; AddLineBreak: Boolean = True);
function ExecAndWait(const CmdLine: String; CmdShow: Word): DWord;
function ControlAtPoint(Point: TPoint; SkipInvisible: Boolean = True;
    SkipDisabled: Boolean = False; SkipTransparent: Boolean = False):
TControl;
```

Description
Micsellaneous routines.

12.11.1 IsDirectoryExists

Unit
acUtils

Declaration
```
function DirectoryExists(DirName: String): Boolean;
```

Description
The **DirectoryExists** function determinates wether a directory specified in *DirName* parameter exists. If the directory exists, the function returns True. If the directory does not exist, the function returns False.

© 1998-2006, UtilMind Solutions®

If a full path name is entered, DirectoryExists searches for the directory along the designated path. Otherwise, the *DirName* parameter is interpreted as a relative path name from the current directory.

See also
DirectoryEmpty function.

12.11.2 IsDirectoryEmpty

Unit
acUtils

Declaration
```
function DirectoryEmpty(DirName: String): Boolean;
```

Description
The **DirectoryEmpty** function determinates wether a directory specified in *DirName* parameter contains files or subdirectories or it is empty. If the directory is empty, the function returns True. If the directory contains files, the function returns False.

If a full path name is entered, DirectoryEmpty searches for the directory along the designated path. Otherwise, the *DirName* parameter is interpreted as a relative path name from the current directory.

See also
DirectoryExists function.

12.11.3 ExtractFileSize

Unit
acUtils

Declaration
```
function ExtractFileSize(FileName: String): Extended;
```

Description
The **ExtractFileSize** function returns the size (in bytes) of file specified in *FileName* parameter. The return value is number with floating point. Use FloatToStr function to convert the return value to string.

Note
Function returns -1 if specified file couldn't be found or opened.

See also
FileSizeToStr function.

12.11.4 FileSizeToStr

Unit
acUtils

Declaration
```
function FileSizeToStr(FileName: String;
                       [Bytes, Kb, Mb: String]): String;
```

Description
The **FileSizeToStr** function returns the string representation of file size. You just need to point the *FileName* (file name with full path) and function will return the file size in bytes, kilobytes or

© 1998-2006, UtilMind Solutions®

megabytes (depending on actual size).

If the file size is less than 1 kilobyte, the size will be returned in bytes, if less than 1 megabyte, the size will returned in kilobytes, and in megabytes if size is over 1 megabyte.

The number of digits after decimal points depends on file size as well. There is no digits after dicimal point for size in bytes, 1 digit in value in kilobytes and 2 digits in value in megabytes.

For example, return value for file with size of 21090 bytes will be "20.5 KB". If file size is 3867291 bytes, the return value will be "3.68 MB".

The *Bytes*, *Kb* and *Mb* is optional parameters used for localization. Use it to override standard English designations of the file size to another language.
Default values is "bytes", "KB" and "MB".

Examples

```
1. FileSizeLabel.Caption := FileSizeToStr(FileName);

2. ListItem.Caption := ExtractFileName(FileName) + '( ' +
   FileSizeToStr(FileName, 'bytes', 'KB', 'MB') + ')';

3. ListItem.Caption := ExtractFileName(FileName) + '( ' +
   FileSizeToStr(FileName, 'bytes', 'kilobytes', 'megabytes') + ')';
```

See also

ExtractFileSize function.

12.11.5 GetShiftState

Unit

acUtils

Declaration

```
function GetShiftState: TShiftState;
```

Description

The **GetShiftState** function returns current state for shift keys (Shift, Ctrl, Alt).

Original code

```
function GetShiftState: TShiftState;
begin
  Result := [];
  if GetKeyState(VK_SHIFT) < 0 then Include(Result, ssShift);
  if GetKeyState(VK_CONTROL) < 0 then Include(Result, ssCtrl);
  if GetKeyState(VK_MENU) < 0 then Include(Result, ssAlt);
end;
```

12.11.6 GetKeyName

Unit

acUtils

Declaration

```
function GetKeyName(lParam: LongInt): String;
```

Description

The **GetKeyName** function converts the virtual keyboard code to the key name. It can convert the

© 1998-2006, UtilMind Solutions®

keyboard codes returned by KeyboardProc, WM_KEYDOWN and WM_KEYDOWN messages.

Original code

```
function GetKeyName(lParam: LongInt): String;
var
  szKeyName: Array[0..$FF] of Char;
begin
  GetKeyNameText(lParam, szKeyName, SizeOf(szKeyName));
  Result := szKeyName;
end;
```

12.11.7 GetWorkArea

Unit

acUtils

Declaration

```
function GetWorkArea: TRect;
```

Description

The **GetWorkArea** function retrieves the size of the working area. The working area is the portion of the screen not obscured by the tray and application desktop toolbars.

Original code

```
function GetWorkArea: TRect;
begin
  SystemParametersInfo(SPI_GETWORKAREA, 0, @Result, 0);
end;
```

12.11.8 GetWindowTitle

Unit

acUtils

Declaration

```
function GetWindowTitle(Wnd: hWnd): String;
```

Description

The **GetWindowTitle** function retrieves the title of window by its handle.

Original code

```
function GetWindowTitle(Wnd: hWnd): String;
var
  WindowTitleLength: Integer;
begin
  WindowTitleLength := GetWindowTextLength(Wnd) + 1;
  SetLength(Result, WindowTitleLength);
  if WindowTitleLength <> 1 then
    Windows.GetWindowText(Wnd, @Result[1], WindowTitleLength);
  SetLength(Result, WindowTitleLength - 1);
end;
```

12.11.9 PercentsOf

Unit

acUtils

© 1998-2006, UtilMind Solutions®

Declaration
```
function PercentsOf(Entire, Part: Extended): Byte;
```

Description
The **PercentsOf** procedure calculates the number of percents which contains the *Part* parameter in *Entire* parameter.

Original code
```
function PercentsOf(Entire, Part: Extended): Byte;
begin
  Result := 0;
  if Entire <> 0 then
   try
     Result := Round(Part * 100 / Entire);
   except
   end;
end;
```

12.11.10 StartScreenSaver

Unit
acUtils

Declaration
```
procedure StartScreenSaver;
```

Description
The **StartScreenSaver** procedure executes default screensaver.

Alternatively you can use following command: ;-)
```
SendMessage(GetDesktopWindow, WM_SYSCOMMAND, SC_SCREENSAVE, 0);
```

See also
IsScreenSaverRunning function.

12.11.11 LoadResourceToStream

Unit
acUtils

Declaration
```
function LoadResourceToStream(Instance: hInst; const ResName, ResType:
String;
    Stream: TStream): Boolean;
```

Description
Retrieves resource, specified by name (*ResName*) and type (*ResType*) to the *Stream*.

ResName parameter specifies the name of the resource. If the first character of the string is a pound sign (#), the remaining characters represent a decimal number that specifies the integer identifier of the resource's name or type. For example, the string '#258' represents the integer identifier 258.

ResType parameter is the string which identifies the type of resource. For standard resource types, this parameter can be one of the following values:

Value	Meaning
RT_ACCELERATOR	Accelerator table

© 1998-2006, UtilMind Solutions®

RT_ANICURSOR	Animated cursor
RT_ANIICON	Animated icon
RT_BITMAP	Bitmap resource
RT_CURSOR	Hardware-dependent cursor resource
RT_DIALOG	Dialog box
RT_FONT	Font resource
RT_FONTDIR	Font directory resource
RT_GROUP_CURSOR	Hardware-independent cursor resource
RT_GROUP_ICON	Hardware-independent icon resource
RT_ICON	Hardware-dependent icon resource
RT_MENU	Menu resource
RT_MESSAGETABLE	Message-table entry
RT_RCDATA	Application-defined resource (raw data)
RT_STRING	String-table entry
RT_VERSION	Version resource

Example

```
// Demonstrates custom implementation over LoadResourceToStream
procedure LoadJPEG(ResourceID: Integer; Picture: TPicture);
var
  JPG: TJPEGImage;
  Stream: TMemoryStream;
begin
  JPG := TJPEGImage.Create;
  try
    Stream := TMemoryStream.Create;
    try
      LoadResourceToStream(hInstance,
         '#' + IntToStr(ResourceID), 'JPG', Stream);
      Stream.Seek(0, soFromBeginning);
      JPG.LoadFromStream(Stream);
      Picture.Assign(JPG);
    finally
      Stream.Free;
    end;
  finally
    JPG.Free;
  end;
end;
```

See also

ExtractResourceToFile function.

12.11.12 ExtractResourceToFile

Unit

acUtils

Declaration

```
function ExtractResourceToFile(Instance: hInst; const ResName, ResType,
FileName: String;
   MarkAsTemporaryFile: Boolean = False): Boolean;
```

Description

The **ExtractResourceToFile** retrieves the resource, specified by *ResName* and *ResType* paremters to the *FileName*.

© 1998-2006, UtilMind Solutions®

Example

```
// we are using this routine in AutoUpgrader
procedure TacAutoUpgrader.Relaunch(const ParamStr: String);
begin
  // writting Upgrader.exe (self upgrading module)
  if ExtractResourceToFile(hInstance, UpgraderModuleName,
UpgraderModuleName, UpgraderFileName, True) then
   begin
    // start self-upgrading mechanism
    WinExec(PChar(UpgraderFileName + ' ' + Application.ExeName + ' ' +
ParamStr), SW_SHOWNORMAL);

    FRestarting := True;
    if Assigned(FOnDoOwnCloseAppMethod) then
      FOnDoOwnCloseAppMethod(Self)
    else
      ShutdownApplication;
   end;
end;
```

See also

LoadResourceToStream function.

12.11.13 StreamToString

Unit

acUtils

Declaration

```
function StreamToString(Stream: TStream): String;
```

Description

The StreamToString function converts the content of the stream to a string. It determines the size of future string by stream size and put full content of stream to a string.

Original code

```
function StreamToString(Stream: TStream): String;
var
  Len: DWord;
begin
  with Stream do
   begin
    Len := Size - Position;
    SetLength(Result, Len);
    Read(Result[1], Len);
   end;
end;
```

See also

PutStringToStream and GetStringFromStream routines.

12.11.14 PutStringToStream

Unit

acUtils

Declaration

© 1998-2006, UtilMind Solutions®

```
procedure PutStringToStream(Str: String; Stream: TStream);
```

Description

The PutStringToStream procedure puts the string to stream (to current position of stream). To retrieve the string from stream — use GetStringFromStream function.

Original code

```
procedure PutStringToStream(const Str: String; Stream: TStream);
var
  Len: DWord;
begin
  Len := Length(Str);
  with Stream do
   begin
    Write(Len, SizeOf(Len));
    Write(Str[1], Len);
   end;
end;
```

See also

GetStringFromStream function.

12.11.15 GetStringFromStream

Unit

acUtils

Declaration

```
function GetStringFromStream: String;
```

Description

The GetStringFromStream function retrieves the string from current position of stream. It determines the size of string by first 4 bytes read from stream. This function should be used to get the string, if the string was put to stream with PutStringToStream procedure.

Original code

```
function GetStringFromStream(Stream: TStream): String;
var
  Len: DWord;
begin
  with Stream do
   try
     Read(Len, SizeOf(Len));
     { check whether the stream really contain the line with such length
}
     if Size - Position >= Len then
      begin
       SetLength(Result, Len);
       Read(Result[1], Len);
      end
     else Result := '';
   except
   end;
end;
```

See also

© 1998-2006, UtilMind Solutions®

GetStringFromStream function.

12.11.16 MsgBox

Unit
acUtils

Declaration
```
function MsgBox(Msg: String; Flags: LongInt);
```

Description
Shows the message box with specified text (*Msg* parameter) and flags (*Flags* parameter).

Original code
```
function MsgBox(Msg: String; Flags: LongInt): Integer;
begin
  Result := Application.MessageBox(PChar(Msg), PChar(Application.Title),
Flags);
end;
```

See also
InfoBox and ErrorBox procedures.

12.11.17 InfoBox

Unit
acUtils

Declaration
```
function InfoBox(Msg: String);
```

Description
Shows the "information box" with "I" icon and text, specified in *Msg* parameter.

Original code
```
procedure InfoBox(Msg: String);
begin
  MsgBox(Msg, MB_OK or MB_ICONINFORMATION);
end;
```

See also
MsgBox and ErrorBox procedures.

12.11.18 ErrorBox

Unit
acUtils

Declaration
```
function ErrorBox(Msg: String; Flags: LongInt);
```

Description
Shows the "error box" with "STOP" icon and text, specified in *Msg* parameter.

Original code
```
procedure ErrorBox(Msg: String);
begin
```

© 1998-2006, UtilMind Solutions®

```
  MsgBox(Msg, MB_OK or MB_ICONSTOP);
end;
```

See also
MsgBox and InfoBox procedures.

12.11.19 XORStr

Unit
acUtils

Declaration
```
function XORStr(Str: String; XORByte: Byte);
```

Description
XORStr is VERY simple crypting routine. It just XOR's every character of string, specified in *Str* parameter, to the byte, specified in *XORByte* parameter.

Original code
```
function XORStr(Str: String; XORByte: Byte): String;
var
  I: Integer;
begin
  Result := Str;

  I := Length(Str);
  if I <> 0 then
   for I := 1 to I do
    Result[I] := Char(Byte(Str[I]) xor XORByte);
end;
```

12.11.20 ShutdownApplication

Unit
acUtils

Declaration
```
procedure ShutdownApplication;
```

Description
Very quickly close an application. (First close main form then terminate whole application.)

Original code
```
procedure ShutdownApplication;
begin
  if Assigned(Application.MainForm) then
    Application.MainForm.Close;
  Application.Terminate;
end;
```

12.11.21 HideFromTaskList

Unit
acUtils

Declaration
```
procedure HideFromTaskList;
```

© 1998-2006, UtilMind Solutions®

Description
The HideFromTaskList procedure hides the application from the task list of Windows 95/98/ME.

This routine do NOTHING on NT-family machines (NT4/2000/XP etc).

12.11.22 IsMetricSystem

Unit
acUtils

Declaration
```
function IsMetricSystem: Boolean;
```

Description
Checks the system, whether it uses Metric or old British measurement system.

Original code
```
function IsMetricSystem: Boolean;
begin
  Result := GetLocaleChar(GetThreadLocale, LOCALE_IMEASURE, #0) = '0';
end;
```

12.11.23 IsDebug

Unit
acUtils

Declaration
```
function IsDebug: Boolean;
```

Description
The IsDebug function returns True if program executed under standard debugger, or False if it is not in debug mode.

Original code
```
function IsDebug: Boolean;
begin
  Result := DebugHook <> 0;
end;
```

12.11.24 Wait

Unit
acUtils

Declaration
```
procedure Wait(var WaitHandle: THandle; Timeout: DWord = 0);
```

Description
The Wait procedure used to wait some time in that line of code which called Wait procedure, until the *WaitHandle* (system event) is signalled (StopWait procedure is called), OR the *Timeout* is expired. After calling the StopWait procedure, or expiration of the *Timeout*, the program will continue from the next line of your code.

The Wait and StopWait routines should be used instead "`while something do Application.ProcessMessages;`", to not utilize CPU time for long time, until some operation

© 1998-2006, UtilMind Solutions®

will be completed.

Remarks

When *Timeout* parameter is set to 0, the function considering it as *Timeout* does not specified, so procedure will wait infinitely, until the *WaitHandle* is signalled (or StopWait procedure will be called).

Example

```
TMyForm = class(TForm)
private
  FWaitEvent: THandle;
public
  procedure CallSomethingAndWait;
  procedure DestroySomethingAndStopWait(Sender: TObject);
end;

procedure TMyForm.CallSomethingAndWait;
begin
  // ...call something

  Wait(FWaitEvent); // ...and wait result...

  // next line will be executed after StopWait will be called OR the
application will be terminated
end;

procedure TMyForm.DestroySomethingAndStopWait(Sender: TObject);
begin
  // this code should be executed when the some long operation completed
and the program should stop awaiting
  StopWait(FWaitEvent);
end;
```

Original code

```
procedure Wait(var WaitHandle: THandle; Timeout: DWord {$IFDEF D4} = 0
{$ENDIF});
begin
  if Timeout = 0 then Timeout := INFINITE;
  WaitHandle := CreateEvent(nil, True, False, nil);  // manual reset,
start non-signaled
  try
    while not Application.Terminated and
      (MsgWaitForMultipleObjects(1, WaitHandle, False, Timeout,
QS_ALLINPUT) = WAIT_OBJECT_0 + 1) do
      Application.ProcessMessages;
  finally
    CloseHandle(WaitHandle);
    WaitHandle := INVALID_HANDLE_VALUE;
  end;
end;
```

See also

StopWait procedure.

12.11.25 StopWait

Unit

acUtils

© 1998-2006, UtilMind Solutions®

Declaration

```
procedure StopWait(var WaitHandle: THandle);
```

Description

The StopWait procedure used to send a signal to *WaitHandle*, so the program will continue its execution from next line after the call of Wait procedure.

The Wait and StopWait routines should be used instead "**while** *something* **do** `Application.ProcessMessages;`", to not utilize CPU time for long time, until some operation will be completed.

Original code

```
procedure StopWait(var WaitHandle: THandle);
begin
  if (WaitHandle <> INVALID_HANDLE_VALUE) and (WaitHandle <> 0) then
    SetEvent(WaitHandle);
end;
```

See also

Wait procedure.

12.11.26 LogToFile

Unit

acUtils

Declaration

```
procedure LogToFile(const FileName, Str: String; AddDateTime: Boolean =
True; AddLineBreak: Boolean = True);
```

Description

The LogToFile procedure adds a new string (*Str* parameter) to the file specified by *FileName* parameter.

Optionally you can specify whether each line which about to be written should contain the current date and time (*AddDateTime* parameter), and whether you want to separate lines with CRLF characters (*AddLineBreak* parameter).

Example

```
procedure TMainForm.Log(const St: String);
begin
  if OutputLog1.Checked and not (csDestroying in ComponentState) then
    LogMemo.Lines.Add(DateTimeToStr(Now) + ' ' + St);

  if LogToFile1.Checked and not (csDestroying in ComponentState) then
    LogToFile('C:\Logs\MyLog.log', St);
end;
```

Original code

```
procedure LogToFile(const FileName, Str: String;
  AddDateTime: Boolean {$IFDEF D4} = True {$ENDIF};
  AddLineBreak: Boolean {$IFDEF D4} = True {$ENDIF});
var
  FileHandle: hFile;
  Dummy: DWord;
  DTSt: String;
```

© 1998-2006, UtilMind Solutions®

```
begin
  FileHandle := CreateFile(PChar(FileName), GENERIC_WRITE,
FILE_SHARE_READ or FILE_SHARE_WRITE, nil,
                           OPEN_ALWAYS, FILE_ATTRIBUTE_NORMAL, 0);
  if FileHandle <> INVALID_HANDLE_VALUE then
   begin
    SetFilePointer(FileHandle, 0, nil, FILE_END);
    if AddDateTime then
     begin
      DTSt := '[' + DateTimeToStr(Now) + '] ';
      WriteFile(FileHandle, DTSt[1], Length(DTSt), Dummy, nil);
     end;
    WriteFile(FileHandle, Str[1], Length(Str), Dummy, nil);
    if AddLineBreak then
      WriteFile(FileHandle, CRLF, 2, Dummy, nil);
    FlushFileBuffers(FileHandle);
    CloseHandle(FileHandle);
   end;
end;
```

See also

Wait procedure.

12.11.27 ExecAndWait

Unit

acUtils

Declaration

```
function ExecAndWait(const CmdLine: String; CmdShow: Word): DWord;
function ExecAndWait(const FileName, Params: String; CmdShow: Word =
SW_SHOWNORMAL): DWord;
```

Description

The ExecAndWait function starts an executable file and waits before process next line of code until the executed file will be terminated.

Example

```
ExecAndWait('C:\Utils\MyTool.exe', SW_SHOWNORMAL);
```

12.11.28 ControlAtPoint

Unit

acUtils

Declaration

```
function ControlAtPoint(Point: TPoint;
    SkipInvisible: Boolean = True;
    SkipDisabled: Boolean = False;
    SkipTransparent: Boolean = False): TControl;
```

Description

The ControlAtPoint retreives the control that contains the specified point.

The main advantage of this function is that it can find even disabled and hidden controls/windows. If no control exists in specified point, function returns nil (NULL).

© 1998-2006, UtilMind Solutions®

12.12 Russian code tables

Unit
acRusCodeTables

Declaration
```
function DOSToKOI(St: String): String;
function DOSToWin(St: String): String;
function WinToDOS(St: String): String;
function WinToKOI(St: String): String;
function KOIToDOS(St: String): String;
function KOIToWin(St: String): String;
```

Description
These functions converts the text in three major russian code tables: DOS, KOI8-R and CP-1251.

12.12.1 DOSToKOI

Unit
acRusCodeTables

Declaration
```
function DOSToKOI(St: String): String;
```

Description
The **DOSToKOI** function converts the string from the DOS code table to KOI8-R.

See also
KOIToDOS function.

12.12.2 DOSToWin

Unit
acRusCodeTables

Declaration
```
function DOSToWin(St: String): String;
```

Description
The **WinToDOS** function converts the string from the DOS code table to Windows (CP-1251).

See also
WinToDOS function.

12.12.3 WinToDOS

Unit
acRusCodeTables

Declaration
```
function WinToDOS(St: String): String;
```

Description
The **WinToDOS** function converts the string from the Windows (CP-1251) code table to DOS.

See also

© 1998-2006, UtilMind Solutions®

DOSToWin function.

12.12.4 WinToKOI

Unit
acRusCodeTables

Declaration
```
function WinToKOI(St: String): String;
```

Description
The **WinToKOI** function converts the string from the Windows (CP-1251) code table to KOI8-R.

See also
KOIToWin function.

12.12.5 KOIToDOS

Unit
acRusCodeTables

Declaration
```
function KOIToDOS(St: String): String;
```

Description
The **KOIToDOS** function converts the string from the KOI80R code table to DOS.

See also
DOSToKOI function.

12.12.6 KOIToWin

Unit
acRusCodeTables

Declaration
```
function KOIToWin(St: String): String;
```

Description
The **KOIToWin** function converts the string from the KOI8-R code table to Windows (CP-1251).

See also
WinToKOI function.

12.13 WindowsXP Themes

Unit
acXPThemes

Declarations
```
// True when Windows XP theme active
function IsThemeActive: Boolean;
// True when the theme is active for current process
function IsAppThemed: Boolean;
// True if possible to use XP theme for controls
function IsThemeEnabled: Boolean;
```

© 1998-2006, UtilMind Solutions®

```
// WinXP Theme API
function OpenThemeData(Wnd: hWnd; pszClassList: lpCWStr): hTheme;
function CloseThemeData(Theme: hTheme): hResult;
function DrawThemeBackground(Theme: hTheme; DC: hDC; iPartId, iStateId:
Integer; const Rect: TRect; pClipRect: PRect): hResult;
function DrawThemeParentBackground(Wnd: hWnd; DC: hDC; prc: PRect):
hResult;

// Custom WinXP Theme functions
procedure DrawThemedControl(Wnd: hWnd; DC: hDC; ClassList: lpCWStr;
iPartId, iStateId: Integer; const Rect: TRect);
```

Description

This unit provides some functions of Windows XP Theme API + one custom routine (DrawThemedControl), which allows to quickly draw themed control even without knowing general specifications of theme API.

See also

DrawThemedControl and IsXP functions.

12.13.1 IsThemeEnabled

Unit

acXPThemes

Declaration

```
function IsThemeEnabled: Boolean; // returns True if application can use
WinXP themes
```

Description

The **IsThemeEnabled** function determines whether the application can or can not use Windows XP themes to draw the interface.

See also

IsXP, IsAppThemed and IsThemeActive functions.

12.13.2 IsThemeActive

Unit

acXPThemes

Declaration

```
function IsThemeActive: Boolean; // returns True if WinXP theme active
```

Description

The **IsThemeActive** function can be used to test if a system theme is active for the current user session.

See also

IsXP, IsAppThemed and IsThemeEnabled functions.

12.13.3 IsAppThemed

Unit

acXPThemes

© 1998-2006, UtilMind Solutions®

Declaration

```
function IsAppThemed: Boolean; // returns True if Theme active
```

Description

The **IsAppThemed** function a theme is active and available to the current process.

See also

IsXP, IsThemeActive and IsThemeEnabled functions.

12.13.4 OpenThemeData

Unit

acXPThemes

Declaration

```
function OpenThemeData(Wnd: hWnd; pszClassList: lpCWStr): hTheme;
```

Description

The **OpenThemeData** function opens the theme data for the specified window (the window handle should be specified in *Wnd* parameter) and semi-colon separated list of class names (*pszClassList* parameter).

Parameters

Wnd	Window handle of the control/window to be themed;
pszClassList	Class name (or list of names separated by semi-colon) to match to theme data section.

Return value

OpenThemeData tries to match each class, one at a time, to a class data section in the active theme. If a match is found, an associated HTHEME handle is returned. If no match is found NULL (nil) is returned.

Remarks

The *pszClassList* parameter contains a list, not just a single name, to provide the class an opportunity to get the best match between the class and the current visual style. For example, a button might pass L"OkButton, Button" if its ID=ID_OK. If the current visual style has an entry for OkButton, that is used, otherwise no visual style is applied.

See also

IsXPThemeEnabled, CloseThemeData, DrawThemeBackground, DrawThemeParentBackground and DrawThemedControl functions.

12.13.5 CloseThemeData

Unit

acXPThemes

Declaration

```
function CloseThemeData(Theme: hTheme): hResult;
```

Description

The **CloseThemeData** function closes the theme data handle, previously opened by OpenThemeData function.

The *Theme* parameter is a handle to a window's specified theme data.

© 1998-2006, UtilMind Solutions®

Function returns S_OK if successful, or an error value otherwise.

Remarks

The CloseThemeData function should be called when a window that has a visual style applied is destroyed. Also this function is used when a window receives a WM_THEMECHANGED message followed by an attempt to create a new theme data handle.

See also

IsXPThemeEnabled, OpenThemeData, DrawThemeBackground, DrawThemeParentBackground and DrawThemedControl functions.

12.13.6 DrawThemeBackground

Unit

acXPThemes

Declaration

```
function DrawThemeBackground(Theme: hTheme; DC: hDC; iPartId, iStateId:
Integer; const Rect: TRect; pClipRect: PRect): hResult;
```

Description

The **DrawThemeBackground** function draws the background image defined by the visual style for the specified control part.

Paramters

Theme	Handle to a window's specified theme data. Use OpenThemeData to create an HTHEME.
DC	Handle to a device context (HDC) used for drawing the theme-defined background image.
iPartId	Value of type int that specifies the part to draw.
iStateId	Value of type int that specifies the state of the part to draw.
Rect	A TRect structure that contains the rectangle, in logical coordinates, in which the background image is drawn.
pClipRect	Pointer to a RECT structure that contains a clipping rectangle. This parameter may be set to NULL.

Function returns S_OK if successful, or an error value otherwise.

Remarks

Drawing operations are scaled to fit and not exceed the rectangle specified in pRect. Your application should not draw outside the rectangle specified by pClipRect.

The background defined by the visual style can be based on a bitmap file, a border, or fill pattern.

See also

IsXPThemeEnabled, OpenThemeData, CloseThemeData, DrawThemeParentBackground and DrawThemedControl functions.

12.13.7 DrawThemeParentBackground

Unit

acXPThemes

Declaration

```
function DrawThemeParentBackground(Wnd: hWnd; DC: hDC; prc: PRect):
```

© 1998-2006, UtilMind Solutions®

```
hResult;
```

Description

The **DrawThemeParentBackground** function used by partially-transparent or alpha-blended child controls to draw the part of their parent that they appear in front of.

Paramters

Wnd	Handle of the child control;
DC	Graphic descriptor of the child control;
prc	Optional TRect structure that defines the area to be drawn (CHILD coordinates).

Function returns S_OK if successful, or an error value otherwise.

See also

IsXPThemeEnabled, OpenThemeData, CloseThemeData, DrawThemeBackground and DrawThemedControl functions.

12.13.8 DrawThemedControl

Unit

acXPThemes

Declaration

```
procedure DrawThemedControl(Wnd: hWnd; DC: hDC; ClassList: lpCWStr;
iPartId, iStateId: Integer; const Rect: TRect);
```

Description

The **DrawThemedControl** function helps to quickly and easily draws the control without opening/closing the theme and painting the parent's background. It is helper for standard WinXP theme API.

Parameters

Wnd	Window handle of the control/window to be themed
DC	Handle to a device context (HDC) used for drawing the theme-defined background image;
ClassList	Class name (or list of names separated by semi-colon) to match to theme data section;
iPartId	Value of type int that specifies the part to draw;
iStateId	Value of type int that specifies the state of the part to draw;
Rect	A TRect structure that contains the rectangle, in logical coordinates, in which the background image is drawn.

Original code

```
procedure DrawThemedControl(Wnd: hWnd; DC: hDC; ClassList: lpCWStr;
iPartId, iStateId: Integer; const Rect: TRect);
var
  Theme: hTheme;
begin
  Theme := OpenThemeData(Wnd, ClassList);
  if Theme <> 0 then
   begin
    DrawThemeParentBackground(Wnd, DC, @Rect);
    DrawThemeBackground(Theme, DC, iPartId, iStateId, Rect, nil);
    CloseThemeData(Theme);
```

© 1998-2006, UtilMind Solutions®

```
    end;
  end;
```

See also

IsXPThemeEnabled, OpenThemeData, CloseThemeData, DrawThemeBackground and DrawThemeParentBackground functions.

© 1998-2006, UtilMind Solutions®

Index

- A -

acAccurateTimer 83
acAnimationEffect 85
acApartPopupMenu 523
acAppAutoRun 89
acAppBar 100
acAppCursors 123
acAppEvents 130
acAppGlobalAlert 149
acAppGlobalSwitch 151
acAppStartCounter 154
acAutoUpgrader 158
acAwayTimer 196
acBitBtn 523
acButton 527
acButtonsPanel 530
acCaptionButton 200
acCheckBox 543
acCheckGroupBox 548
acClipboard 214
acComboDatePicker 551
acConnectionList 221
acControlKeyState 558
acCronJob 240
acCurrencyEdit 561
acCursorLocker 243
acDateTimeCalc 561
acDBListView 564
acDesktopWallpaper 565
acDragPanel 568
acEdit 569
acEmbeddedForm 244
acExtAssociation 246
acExtAssociation example 252
acFileStorage 253
acFlashWindow 912
acFormBackground 262
acFormHelp 266
acFormHints 288
acFormHook 294
acFormMagnet 301
acFormPlacementSaver 306
acFormResizeGrip 312
acFormSizeRestrictions 321
acFormSystemMenu 325
acFormTopmost 327
acGraphics unit 909
acHeader 588
acHeaderPanel 603
acHostnameResolver 620
acHTMLViewer 629
acHTTP 335
acHTTPDownloadQueue 383
acIESniffer 383
acIESnifferAutoFillUserInfo 407
acImagesComboBox 634
acInfoBox 413
acIPEdit 645
acLabel 656
acLinksPanel 678
acListView 681
acMakeOpaqueWindow 911
acMakeTranslucentWindow 910
acMRUComboBox 710
acNumberEdit 726
acOnlyOne 421
acPicturePanel 746
acProcessList 424
acProportionalImage 755
acQuickAboutBox 762
acRadioButton 770
acRichLabel 776
acRSSFeed 437
acRSSFeedContainer 438
acSendMail 438
acShortcut 459
acSimpleTCPClient 781
acSimpleTCPServer 784
acSplitterPanel 789
acSQLQueryThread 797
acStatusControl 801
acSystemImageList 818
acSystemInfo 819
acTextTemplateConverter 838
acThread 464
acThreadSafeLog 474
acTrackBar 845
acTrayIcon 488
acTreeComboBox 845

© 1998-2006, UtilMind Solutions®

acTreeView 874
acUnixCrypt 888
acURLLabel 889
acUtils unit 906
acWavPlayer 512
acWebImage 894
acWin2kEffects 515
Additional colors 903
Additional cursors 904
Advanced GDI routines 909
Advanced Visible Controls 55
Application Controls 49
AutoUpgrader 158

- B -

BinToDec 917
Bonus routines 906

- C -

CaseAnsiCompareText 933
CloseThemeData 974
Components Overview 1 49
Components Overview 2 55
Contact Information 83
ControlAtPoint 970

- D -

Date/Time routines 937
DateTimeToFileTime 940
DateTimeToSQLDate 937
DateTimeToSQLDateTime 937
DateTimeToSQLTime 937
DaysPerMonth 938
DecToBin 916
DecToHex 916
DecToRoman 917
Delay 947
DOSToKOI 971
DOSToWin 971
DrawThemeBackground 975
DrawThemedControl 976
DrawThemeParentBackground 975
dsgnintf.dcu problem 77

- E -

ErrorBox 965
ESC 936
Example of tags of acFormHelp component 286
Example of using acFileStorage 261
ExcludeLeadingChar 931
ExcludeTrailingBackslash 930
ExecAndWait 970
ExploreFolder 920
ExtractFileSize 958
ExtractResourceToFile 962

- F -

FileReadString 935
FileSizeToStr 958
FileStorage - accessing stored files at run-time 261
FileTimeToDateTime 941
FileWriteString 935
FormHelp Designer 287
FullParamStr 933

- G -

GetCPUSpeed 944
GetEnvironmentString 945
GetExecutableByExtension 919
GetKeyName 959
GetLocalHostName 953
GetLocalIP 953
GetOS 942
GetShiftState 959
GetStringFromStream 964
GetSystemDir 945
GetSystemIconIndex 946
GetSystemImageList 919
GetTempDir 944
GetWindowsDir 945
GetWorkArea 960
GMT 939
GMTToLocalTime 939

© 1998-2006, UtilMind Solutions®

- H -

HasParamStr 932
Help designer 287
HexToDec 916
HideFromTaskList 966
Hint property editor 287
History 60
How to install new file association? 252
HTTP status codes 379
HTTPReadString 382

- I -

IncludeLeadingChar 931
IncludeTrailingBackslash 930
InfoBox 965
InfoFile Designer for AutoUpgrader 195
Info-File example 191
Information File example 191
Installation Instructions 77
Installation Problems 77
Int2x32ToInt64 917
Internet Utilities 952
InternetTimeToDateTime 940
Introduction 48
IPToStr 954
Is256ColorTrayIconAllowed 943
IsAppThemed 973
IsDebug 967
IsDirectoryEmpty 958
IsDirectoryExists 957
IsLeapYear 938
IsMetricSystem 967
IsMouseOverAppBar 117
  TacAppBar 117
IsNT 942
IsOnline 953
IsRegValueExists 949
IsScreenSaverRunning 944
IsThemeActive 973
IsThemeEnabled 973
IsValidEmail 930
IsWin2k 943
IsXP 943

- K -

KOIToDOS 972
KOIToWin 972

- L -

LeftPart 926
LeftPartR 927
License Agreement 81
LoadResourceToStream 961
LocalTimeToGMT 939
LogToFile 969

- M -

MinimizeAll 920
Miscellaneous routines 957
MsgBox 965
MySQL String routines 935

- N -

NoSound 947
Numerical Conversion routines 915

- O -

OnActivate 118
  TacAppBar 118
OnClick 118
  TacAppBar 118
OnDblClick 118
  TacAppBar 118
OnDeactivate 119
  TacAppBar 119
OnDocked 119
  TacAppBar 119
OnDocking 119
  TacAppBar 119
OnDockMove 120
  TacAppBar 120
OnExitSizeMove 120
  TacAppBar 120
OnFullScreenApp 120

© 1998-2006, UtilMind Solutions®

OnFullScreenApp 120
TacAppBar 120
OnHiding 120
TacAppBar 120
OnOtherAppBarPosChanged 121
TacAppBar 121
OnUnableToAutoHide 121
TacAppBar 121
OnUnhiding 121
TacAppBar 121
OnWindowArrange 122
TacAppBar 122
OpenThemeData 974
OpenURL 920
OpenWithDlg 921
Overview (Part 1) 49
Overview (Part 2) 55

- P -

ParseURL 954
PercentsOf 960
Playing .WAV from acFileStorage (example) 261
PlaySystemSound 948
PosR 928
PutStringToStream 963

- R -

RegEraseKey 951
RegEraseValue 951
Registration Information 80
Registry routines 949
RegReadFont 952
RegReadInt 951
RegReadStr 950
RegWriteFont 952
RegWriteInt 951
RegWriteStr 950
RepaintScreen 921
RestoreAll 921
Rich-formatting tags in context-sensitive help 286
Rich-text in acFormHelp component 286
RightPart 926
RightPartR 928
RunControlPanel 922
RunCPL 923

- S -

SameDateTime 941
Self-upgrading mechanism 192
SendMail 458
SetFileTimes 941
Shell utilities routines 918
ShowHiddenAppBar 117
TacAppBar 117
ShutdownApplication 966
SockConnect 956
SockDisconnect 956
Sound 946
Sound routines 946
SplitFileNameAndParams 929
SplitStr 924
SQLDateTimeToDateTime 936
StartScreenSaver 961
StopSystemSound 948
StopWait 968
StreamToString 963
String / Filename routines 924
StringReplace 931
StrToDateTimeDef 938
StrToIP 954
StrToStrList 934
System Information routines 941

- T -

TacACaptionButton 91
BtnOrder 91
Cursor 92
CursorDown 92
Enabled 93
Hint 93
SeparatorWidth 93
ShowHint 94
Visible 94
TacAccurateTimer 83
Active 83
Interval 84
OnTimer 85
Reset 84
TacAnimationEffect 85
AnimateOnHide 86

© 1998-2006, UtilMind Solutions®

TacAnimationEffect 85
AnimateOnShow 85
Close 88
Delay 86
EffectType 86
FireworkBlasts 87
Frames 87
OnHideEffect 89
OnShowEffect 88
Open 87
Steps 87
TacApartPopupMenu 523
OnCloseUp 523
PopupFromCursor 523
TacAppAutoRun 89
AutoRun 90
AutoStarted 91
CaptionButton 91
LogonNotice 94
MenuItem 96
OnAutoRun 99
OnButtonClick 100
RunFor 96
RunOnce 96
SystemMenu 97
TacAppBar 100
AllowedEdges 102
AlwaysOnTop 103
AutoHide 103
Docking 104
DontSlideOnTaskBarWhenAutoHide 106
FAQ 122
FloatRestrictions 107
IsMouseOverAppBar 117
LastDockingPlace 109
OnActivate 118
OnClick 118
OnDblClick 118
OnDeactivate 119
OnDocked 119
OnDocking 119
OnDockMove 120
OnExitSizeMove 120
OnFullScreenApp 120
OnHiding 120
OnOtherAppBarPosChanged 121
OnUnableToAutoHide 121
OnUnhiding 121
OnWindowArrange 122
Placement 109
RegistrySaver 110
ShowHiddenAppBar 117
Sizing 114
Sliding 115
TaskIcon 116
TitleBar 116
UpdateBar 117
TacAppBarDocking 104
Enabled 104
Height 105
MaxHeight 105
MaxWidth 105
MinHeight 106
MinWidth 106
Width 105
TacAppBarFloatRestrictions 107
Enabled 107
MaxHeight 108
MaxWidth 107
MinHeight 108
MinWidth 108
TacAppBarRegistrySaver 110
Enabled 110
IniFileName 111
IniSection 111
Options 111
RegKey 112
RegLocation 112
RestoreProperties 113
Storage 113
TacAppBarSizing 114
Enabled 114
HorizontalIncrement 114
VerticalIncrement 115
TacAppBarSliding 115
Enabled 115
SlideTime 116
TacAppCursors 123
BorderBottom 124
BorderBottomLeft 124
BorderBottomRight 124
BorderLeft 125
BorderNonResize 125
BorderRight 125
BorderTop 126
BorderTopLeft 126
BorderTopRight 126
BtnClose 126

© 1998-2006, UtilMind Solutions®

TacAppCursors 123
BtnHelp 127
BtnMaximize 127
BtnMinimize 127
Caption 127
Client 128
ClientNonVCL 128
MainMenu 128
MenuItem 129
OnMouseOverObject 130
ScrollHorz 129
ScrollVert 129
SystemMenu 129
Transparent 130
TacAppEvents 130
AnimateIcon 131
AnimateInterval 131
AppPriorityClass 132
FixedMinimizeRestoreAnimation 132
HideHidePause 133
HintColor 133
HintFont 133
HintPause 133
HintShortPause 134
Icon 134
ImageIndex 134
ImageList 135
IsWindowsShutdowned 135
MouseDragImediate 135
MouseDragThreshold 136
OnActionExecute 138
OnActionUpdate 138
OnActivate 139
OnActiveControlChange 139
OnActiveFormChange 140
OnAppKeyDown 140
OnAppKeyUp 141
OnAppMouseDown 142
OnAppMouseMove 142
OnAppMouseUp 143
OnDeactivate 143
OnException 144
OnHelp 144
OnHint 145
OnIdle 145
OnMessage 146
OnMinimize 146
OnRestore 147
OnSettingsChanged 147
OnShortCut 147
OnShowHint 148
OnWindowsShutdown 148
ShowHint 136
ShowMainForm 136
ShowTaskIcon 137
UpdateFormatSettings 137
UpdateMetricSettings 137
TacAppGlobalAlert 150
Alert 150
GlobalAlert 149
OnAlert 150
TacAppGlobalSwitch 149, 151
GlobalSwitch 151
OnSwitch 154
RegistrySaver 152
Value 153
Enabled 152
RegKey 153
RegLocation 153
RegValue 153
TacAppStartCounter 154
Counter 155
InstallTime 155
LifeTime 156
OnFirstStart 157
RegKey 156
RegLocation 156
ResetCounter 157
TacAttachedLabel 571
Caption 572
Color 572
Cursor 573
Enabled 573
FocusOnClick 574
Font 574
Hint 574
ParentColor 575
ParentFont 575
ParentShowHint 576
PopupMenu 576
Position 576
ShowAccelChar 577
ShowHint 577
Spacing 578
SpacingKind 578
Transparent 579
Visible 579
WordWrap 580

© 1998-2006, UtilMind Solutions®

TacAutoRunLogonNotice 94
Caption 95
Enabled 95
Text 95
TacAutoUpgrader 158
Abort 176
AutoCheck 159
AutoCheckDelay 160
CacheOptions 160
CheckUpdate 177
DoOwnCloseAppMethod 181
HTTPPassword 161
HTTPUsername 161
InfoFile 162
InfoFile Designer 195
InfoFile example 191
InfoFileURL 164
InternetOptions 165
MiscFilesUpgrade 165
OnAborted 178
OnAfterRestart 178
OnBeginUpgrade 179
OnConnLost 180
OnEndUpgrade 181
OnFileDone 182
OnFileStart 183
OnHostUnreachable 183
OnLaterUpgrade 184
OnLostFile 185
OnNoInfoFile 186
OnNoUpdateAvailable 186
OnPasswordRequest 187
OnProgress 188
OnProxyAuthenticationRequest 190
Proxy 166
RestartApplication 177
RestartParams 169
Screenshot1 194
Screenshot2 194
Screenshot3 195
Screenshot4 195
ShowMessages 170
ThreadPriority 171
TransferBufferSize 171
VersionControl 172
VersionDate 172
VersionDateAutoSet 173
VersionNumber 173
Wizard 174
TacAutoUpgraderInfo 162
Files 162
UpgradeMethod 162
UpgradeMsg 163
TacAutoUpgraderWizard 174
Enabled 174
HideFileLocation 174
Language 175
Pic118x218 175
StayOnTop 176
TacAwayTimer 196
Active 197
AwayTime 197
MonitorInterval 198
OnActiveTimer 199
OnScreenSaverEnd 199
OnScreenSaverStart 199
OnSleep 200
ResetAway 198
SleepInterval 198
TacBackground 588
Active 589
GradientColor1 590
GradientColor2 590
GradientDirection 590
Kind 591
Picture 591
PictureStyle 592
TacBitBtn 523
Align 524
Caption 525
Color 525
OnMouseEnter 527
OnMouseLeave 527
ParentColor 526
WordWrap 526
TacButton 527
Align 528
Caption 529
OnMouseEnter 529
OnMouseLeave 530
WordWrap 529
TacButtonsPanel 530
Align 531
Alignment 531
ApplyEnabled 532
BtnApply 532
BtnCancel 533
BtnCustom 534

© 1998-2006, UtilMind Solutions®

TacButtonsPanel 530
BtnHelp 535
BtnOK 536
ButtonHeight 537
ButtonSpacing 537
ButtonWidth 537
CustomButtonPosition 538
Images 538
MarginHorz 539
MarginVert 539
OnAnyBtnClick 540
OnApplyEnabledChanged 541
OnApplySettings 541
OnBtnApplyClick 541
OnBtnCancelClick 542
OnBtnCustomClick 542
OnBtnHelpClick 542
OnBtnOKClick 542
ShowAccelChar 539
ShowDivider 539
ShowSizeGrip 540
TacCaptionButton 200
AllowPress 202
BtnOrder 202
Cursor 203
CursorDown 203
Down 203
Enabled 204
Glyph 204
GlyphDown 204
GlyphTransparent 205
Hint 205
OnButtonClick 212
OnCaptionDblClick 212
OnClick 212
OnMenuClick 213
OnPressed 213
OnReleased 213
PopupMenu 206
RefreshButton 211
RefreshNCArea 211
SepearatorWidth 206
ShowHint 206
ShowParentForm 211
Sign 207
SignFont 207
SignLeft 207
SignTop 208
SystemMenu 208
Visible 210
TacCheckBox 543
Align 544
AlignmentHorz 545
AlignmentVert 545
BoxAlignment 546
Ctl3D 546
OnMouseEnter 547
OnMouseLeave 548
ParentCtl3D 547
WordWrap 547
TacCheckGroupBox 548
BoxAlignment 549
Caption 549
Checked 550
Ctl3D 550
OnCheckBoxClick 551
State 550
TacClipboard 214
Active 214
AvailableFormats 215
Clear 217
ClipboardBitmap 215
ClipboardFiles 216
ClipboardText 216
GetFormatIndex 218
GetFormatName 219
GetStream 219
GetString 219
HasFormat 217
OnChange 221
PutStream 220
PutString 220
TacComboDatePicker 551
Date 552
DateFormat 552
Day 553
Labels 553
MaxYear 555
MinYear 556
Month 556
OnChange 558
SpacingBetweenCombos 556
WidthDayMonthBoxes 557
WidthYearBox 557
Year 557
TacComboDatePickerLabels 553
Position 554
Spacing 554

© 1998-2006, UtilMind Solutions®

TacComboDatePickerLabels 553
StrDay 554
StrMonth 555
StrYear 555
Visible 555
TacConnection 234
Data 235
IsIncoming 240
LocalHostname 236
LocalIP 235
LocalIPLong 236
LocalIPStr 236
LocalPort 237
Proto 239
ProtoName 239
RemoteHostname 238
RemoteIP 237
RemoteIPLong 237
RemoteIPStr 237
RemotePort 238
State 238
StateStr 239
TacConnectionList 221
AutoResolve 222
CacheSize 223
ConnectionList 223
Count 224
FindConnectionByLocalIPStr 230
FindConnectionByLocalPort 229
FindConnectionByRemoteIPStr 231
FindConnectionByRemotePort 230
FindLocalIPStrByLocalPort 231
FindLocalIPStrByRemotePort 231
FindRemoteIPStrByLocalPort 232
FindRemoteIPStrByRemotePort 232
IsOnline 224
IsPortListening 233
Items 225
LocalHostname 226
LocalIP 226
MonitorActive 227
MonitorInterval 227
OnHostnameRetrieved 233
OnOnlineStatusChanged 234
Priority 227
Refresh 229
RegistrySaver 228
RetrieveTCP 228
RetrieveUDP 229
TacControlKeyState 558
CapsLock 559
Control 559
Insert 559
NumLock 559
OnCapsLock 560
OnControl 560
OnInsert 560
OnNumLock 560
OnScrollLock 561
OnShift 561
ScrollLock 559
Shift 560
TacCronJob 240
Active 241
AlertsCount 241
CronMask 242
LastTime 243
OnAlert 243
TacCurrencyEdit 561
OnChange 561
Value 561
TacCursorLocker 243
Active 244
Control 244
TacDateTimeCalc 561
BeginTime 562
Days 563
EndTime 562
Hours 563
Minutes 564
Months 563
MSeconds 564
Seconds 564
Years 562
TacDBListView 564
DataSource 565
TacDesktopWallpaper 565
AutoFormat 566
RemoveWallpaper 567
SetDesktopWallpaper 567
Stretch 566
Tile 566
Wallpaper 567
TacDragPanel 568
DragCursor 568
DragObject 568
TacEdit 569
Align 569

© 1998-2006, UtilMind Solutions®

TacEdit 569
Alignment 570
AttachedLabel 571
AutoSelect 580
Button 580
Color 584
ColorDisabled 584
CursorBorder 584
OnButtonClick 586
OnButtonLeftMouseDown 586
OnLabelClick 586
OnLabelDblClick 587
OnMouseEnter 587
OnMouseLeave 587
SetSelection 585
ValidateChars 585
ValidChars 585
TacEditButton 580
Cursor 581
Flat 581
Glyph 581
Hint 583
Kind 583
NumGlyphs 582
Visible 583
Width 583
TacEmailParams 890
Address 891
BlindCarbonCopy 891
CarbonCopy 891
Message 892
Name 892
Subject 892
TacEmbeddedForm 244
FormName 245
HolderControl 246
TacExtAssociation 246
AccessMode 247
ExecutableFile 248
ExtDescription 248
EXTENSION 248
FileDescription 249
IconIndex 249
IconLocation 249
InstallExtension 251
LargeIcon 250
MIMEType 250
ParamString 250
SmallIcon 251
UninstallExtension 252
TacExtAssociationDefCommandName 247
TacFileStorage 253
AutoExtract 253
Count 258
DataSize 258
Extract 259
Files 258
OnExtract 260
OnExtracted 260
TacFormBackground 262
Active 263
GradientColor1 263
GradientColor2 263
GradientDirection 264
Kind 264
Picture 265
PictureStyle 265
TacFormHelp 266
Active 267
AdjustPopupWidth 268
CaptionButton 268
Color 272
Cursor 272
CursorHelp 272
DelayInterval 272
Font 273
InvokeHelpPointer 282
Margins 273
OnButtonClick 284
OnHide 284
OnShow 284
OnWhatsThis 285
PopupWidth 274
ShadowColor 274
ShowHelp 282
ShowHelpFromControl 283
ShowHelpFromPoint 283
ShowShadow 275
SystemMenu 275
TextStyle 277
UseF1 279
WhatsThis 279
TacFormHelpMargins 273
Horizontal 273
Vertical 273
TacFormHints 288
Active 289
BiDiMode 289

© 1998-2006, UtilMind Solutions®

TacFormHints 288
ColorBackground 290
ColorBorder 290
ColorShadow 290
Direction 290
Font 291
MaxWidth 291
OnShowHint 294
Rounded 291
RoundRadius 292
ShowArrow 292
ShowShadow 292
TextAlignment 293
TransparencyRate 293
Transparent 293
TacFormHook 294
AcceptFiles 295
ActivateFormOnAcceptFiles 295
Bottommost 295
CloseButton 296
DragByCaption 296
DragByClient 296
NoCaption 297
NoPaintBackground 297
OnCaptionChanged 298
OnDropFiles 298
OnFormCloseButtonClick 298
OnFormRecreate 299
OnFormShown 299
OnMessageAfter 300
OnMessageBefore 300
Topmost 297
TacFormMagnet 301
Active 301
DoMagnet 305
FormDragable 302
FormTopmost 302
Glue 302
MagnetType 303
OnDragMove 305
OnGlue 305
OnMagnet 306
Options 302
Power 303
ScreenMagnet 303
TacFormPlacementSaver 306
Enabled 307
FlickerFreeMaximize 307
Load 309
MaximizeIfNoInfoFound 307
OnLoadSettings 310
OnRegKeyNotFound 310
OnSaveSettings 311
Options 307
RegKey 308
RegLocation 308
RestoreFormSize 309
Save 309
TacFormResizeGrip 312
Visible 312
TacFormRoller 312
Animate 313
AnimateSpeed 313
CaptionButton 314
HintRollDown 317
HintRollUp 318
OnRollerChanged 321
RemainHeight 318
RollUp 318
SystemMenu 319
TacFormSizeRestrictions 321
Enabled 322
MaxClientHeight 322
MaxClientWidth 322
MaxHeight 323
MaxWidth 324
MinClientHeight 323
MinClientWidth 323
MinHeight 324
MinWidth 324
TacFormSystemMenu 325
Active 325
InsertPosition 325
Items 326
SysItems 326
SystemMenuHandle 327
TacFormTopmost 327
CaptionButton 328
MenuItem 332
OnTopmostChanged 335
SystemMenu 332
Topmost 334
TacHeader 588
AutoSize 588
Background 588
CaptionMenu 592
Image 593
ImageAlignment 593

© 1998-2006, UtilMind Solutions®

TacHeader 588
ImageIndex 594
ImageList 594
ImageMenu 594
ImageTransparent 594
OnCaptionClick 601
OnImageClick 601
OnMouseEnter 601
OnMouseLeave 602
OnPaint 602
OnRightButton2Click 603
OnRightButtonClick 602
RightButton 595
RightButton2 598
RightCaption 598
RightCaptionFont 598
TextStyle 599
TacHeaderPanel 603
Background 603
BackgroundHeader 607
CaptionMenu 611
Flat 611
FlatBorderColor 612
HeaderHeight 612
Image 612
ImageAlignment 612
ImageIndex 612
ImageList 612
ImageMenu 612
ImageTransparent 612
OnHeaderCaptionClick 617
OnHeaderClick 617
OnHeaderImageClick 617
OnHeaderMouseEnter 618
OnHeaderMouseLeave 618
OnMouseEnter 618
OnMouseLeave 618
OnPaint 619
OnPaintHeader 619
OnRightButton2Click 620
OnRightButtonClick 620
OnRolled 620
PopupMenuHeader 612
RightButton 613
RightButton2 613
RightCaption 613
RightCaptionFont 614
RollOnDblClick 614
RollRegistrySaver 614
ShowHeader 614
ShowSizeGrip 614
Text 614
TextAlignment 614
TextFont 615
TextLeft 617
TextStyle 615
TextTop 617
WordWrap 617
TacHeaderRightButton 595
AutoRoll 595
Cursor 595
Enabled 596
Glyph 596
Hint 596
Kind 596
ShowEdges 597
ShowHint 597
UseCaptionColor 597
Visible 597
TacHostnameResolver 620
Abort 627
AutoResolve 621
CacheSize 621
Execute 627
IPList 621
IsOnline 622
LocalHostname 623
LocalIP 623
MonitorActive 623
MonitorInterval 624
OnHostnameRetrieved 628
OnOnlineStatusChanged 629
Priority 624
RegistrySaver 625
ResolveCachedIP 628
TacHostnameResolverRegistrySaver 625
Enabled 625
RegKey 626
RegLocation 626
TacHTMLViewer 629
BorderColor 630
BorderStyle 630
Color 631
HTML 631
Margins 631
NoScroll 633
NoStdPopupMenu 633
PopupMenu 633

© 1998-2006, UtilMind Solutions®

TacHTMLViewer 629
WebBrowser 634
TacHTMLViewerMargins 631
Bottom 632
Left 632
Right 632
Top 632
TacHTTP 335
Abort 356
AcceptTypes 337
AddHeaders 338
Agent 339
Busy 339
CacheOptions 339
FileName 340
HostName 341
InternetOptions 341
IsGlobalOffline 357
OnAborted 361
OnAnyError 362
OnBeforeSendRequest 362
OnConnLost 362
OnDone 363
OnDoneInterrupted 365
OnHeaderInfo 365
OnHostUnreachable 367
OnHTTPError 367
OnOutputFileError 369
OnPasswordRequest 370
OnProgress 371
OnProxyAuthenticationRequest 373
OnRedirected 373
OnUploadCGITimeoutFailed 374
OnUploadFieldRequest 374
OnUploadProgress 377
OnWaitTimeoutExpired 379
OutputFileAttributes 342
OutputFileName 343
Password 344
Pause 357
POSTData 344
Proxy 345
Range 348
Read 358
ReadRange 358
Referer 349
RequestMethod 349
Resume 359
ShowGoOnlineMessage 350
Suspended 351
Thread 353
ThreadPriority 353
Timeouts 351
TransferBufferSize 354
Upload 360
UploadByFieldNames 360
URL 354
Username 354
WaitThread 355
WaitTimeout 355
WorkOffline 356
TacHTTPDownloadQueue 383
TacHTTPProxy 345
AccessType 346
ProxyBypass 346
ProxyPassword 346
ProxyPort 347
ProxyServer 347
ProxyUsername 347
TacHTTPRange 348
EndRange 348
StartRange 349
TacHTTPTimeouts 351
ConnectTimeout 352
ReceiveTimeout 352
SendTimeout 352
TacIESniffer 383
Active 385
AddBrowser 390
AutoFill 408
AutoHighlight 408
ClearMarks 390
CloseBrowsers 391
Fields 409
Fill 412
FillTokens 410
FormAutoFill 386
HighlightColor 411
HighlightTextColor 411
IEList 386
MarkText 391
MonitorInterval 387
OnURLChange 393
OnWBBeforeNavigate2 394
OnWBCommandStateChange 395
OnWBDocumentComplete 396
OnWBDownloadBegin 397
OnWBDownloadComplete 397

© 1998-2006, UtilMind Solutions®

TacIESniffer 383
OnWBFileDownload 399
OnWBFullScreen 399
OnWBMenuBar 400
OnWBNavigateComplete2 400
OnWBNewWindow2 401
OnWBProgressChange 402
OnWBPropertyChange 402
OnWBQuit 402
OnWBStatusBar 403
OnWBStatusTextChanged 403
OnWBTheaterMode 404
OnWBTitleChange 404
OnWBToolBar 405
OnWBVisible 405
OnWindowLoad 406
OnWindowUnload 407
Refresh 392
RegistrySaver 411
ReplaceText 393
SearchBar 388
SearchRedirect 388
SniffWithHTTPPrefixOnly 389
URLs 389
TacIESnifferAutoFillUserInfo 407
CustomFields 409
Save 413
TacImagesComboBox 634
AttachedLabel 635
Images 644
Items 644
TacInfoBox 413
ButtonCaptions 414
CheckBoxCaption 414
Execute 420
Font 415
Icon 415
InfoBoxButtons 415
OnCheckBoxChecked 420
Picture 415
PictureResID 415
Position 416
RegKey 417
RegLocation 417
RegValue 417
ShowCheckBox 417
Text 418
TextStyle 418
Title 420
TacIPEdit 645
Align 646
Alignment 647
AttachedLabel 647
AutoSelect 652
AutoSize 653
Color 653
ColorDisabled 653
IP1 654
IP2 654
IP3 654
IP4 654
IPLong 655
OnChange 655
Text 655
TacJPEGProperties 757
CompressionQuality 758
Grayscale 758
Performance 758
PixelFormat 759
ProgressiveDisplay 759
ProgressiveEncoding 760
Scale 760
Smoothing 761
TacLabel 656
Blinking 656
CursorClick 657
FontClick 658
FontHover 660
Highlight 662
OnMouseEnter 666
OnMouseLeave 667
Options 666
Shadow 664
WordWrap 666
TacLabelBlinking 656
DelayInvisible 657
DelayVisible 657
Enabled 657
TacLabeledComboBox 677
OnLabelClick 677
OnLabelDblClick 677
OnMouseEnter 678
OnMouseLeave 678
SetSelection 677
TacLabeledComboBox.AttachedLabel 667
TacLabelEffect 662
Color 662
ColorClick 662

© 1998-2006, UtilMind Solutions®

TacLabelEffect 662
ColorHover 663
Depth 663
Direction 663
Enabled 663
TacLinksPanel 678
FontClick 678
FontHover 679
ImageList 679
ImageSpacing 679
Items 679
LinksProperties 679
MarginLeft 680
MarginTop 680
OnLinkClick 680
Spacing 680
TacListView 681
BestFit 698
CheckOnClick 684
Clear 699
CopySelectionToClipboard 699
CountPerPage 699
Cursors 684
DeleteSelected 700
EmptyListMessage 686
EmptyListMessageColor 686
ExportToHTML 700
ExportToTXT 701
FindCaption 701
GetColumnAt 702
GetItemAt 702
HeaderColor 687
HeaderFont 687
HeaderMenu 687
HeaderStyle 690
ImportFromTXT 703
IsScrollBarsVisible 703
ItemPopupMenu 691
OnAlignmentChanged 704
OnCheckBoxClick 705
OnColumnDragged 705
OnColumnResizeBegin 706
OnColumnResizeEnd 706
OnCustomDrawSubItemEx 707
OnDrawHeaderSection 707
OnMouseEnter 708
OnMouseLeave 708
OnSettingsChanged 708
OnToolTip 709
RegistrySaver 691
SelectAll 704
SetColumnOrders 704
ShowEmptyListMessage 693
ShowHeaderMenu 693
ShowScrollTips 693
ShowSortMark 694
ShowToolTips 694
SortColumn 695
SortDirection 695
SortMarkAlign 696
SortMarkImageAsc 696
SortMarkImageDesc 697
SortMarkKind 697
ToolTipOptions 698
TacListViewCursors 684
Border 684
Divider 685
Header 685
HotCursor 685
ScrollHorz 685
ScrollVert 686
TacListViewHeaderMenu 687
AlignCenterItem 689
AlignLeftItem 688
AlignRightItem 688
AscendingItem 689
BestFitItem 690
DescendingItem 689
PopupMenu 690
TacListViewRegistrySaver 691
Enabled 691
RegKey 692
RegLocation 692
TacMRUComboBox 710
Align 711
AttachedLabel 712
AutoFill 721
AutoFillCaseSensitive 721
AutoFillDelay 722
ClearMRU 724
MRUEnabled 722
MRUMax 722
MRURegKey 722
MRURegLocation 723
MRUSaveOnExit 723
OnMouseEnter 725
OnMouseLeave 725
OnSubmit 726

© 1998-2006, UtilMind Solutions®

TacMRUComboBox 710
Save 724
SetSelection 725
SubmitButton 724
TacNumberEdit 726
Align 727
Alignment 728
AttachedLabel 728
AutoSelect 737
Button 738
Color 741
ColorDisabled 741
CursorBorder 742
Kind 742
MaxValue 742
OnButtonClick 743
OnButtonLieftMouseDown 744
OnChange 744
OnLabelClick 744
OnLabelDblClick 745
OnMouseEnter 745
OnMouseLeave 745
SetSelection 743
Value 743
TacOnlyOne 421
CloseFirst 422
CloseFirstInstance 423
CloseSecond 422
OnAlreadyExist 424
SwitchToFirst 422
TacPicturePanel 746
Background 746
BevelStyle 750
Image 750
ImageAlignment 751
ImageIndex 751
ImageList 751
ImageMargin 751
ImageTransparent 751
Margin 751
OnMouseEnter 754
OnMouseLeave 755
OnPaint 755
ShowBevel 752
Text 752
TextAlignment 752
TextStyle 752
TacProcess 430
AppType 431
AppTypeName 432
Data 432
Description 433
ExeName 433
Kill 437
ProcID 434
SystemIcon16Handle 434
SystemIcon32Handle 434
SystemIconIndex 435
Windows 436
WindowTitle 436
TacProcessList 424
Count 425
FindDescriptionByPID 429
FindDescriptionByWnd 429
FindExeNameByPID 428
FindExeNameByWnd 429
FindProcessByPID 428
FindProcessByWnd 428
FindWindowTitleByPID 430
FindWindowTitleByWnd 430
Items 425
ProcessList 427
Refresh 427
TacProportionalImage 755
Alignment 755
ControlCanvas 756
DrawRect 756
FixedStretching 757
JPEGProperties 757
Proportional 761
TacQuickAboutBox 762
AnimationDelay 762
AnimationEffect 762
Caption 763
Copyright 763
Cursors 763
Description 765
Email 765
Execute 770
GradientBackground 765
Icon 766
Localization 767
Registered 768
RegisteredTo 768
RegistrationURL 769
Shareware 769
ShowBevels 769
ShowHints 769

© 1998-2006, UtilMind Solutions®

TacQuickAboutBox 762
WebSite 770
Width 770
TacQuickAboutBoxFormCursors 763
Border 764
BtnClose 764
Buttons 764
Caption 763
Client 764
Links 764
TacQuickAboutBoxGradientBackground 765
Color1 765
Color2 766
Direction 766
Enabled 766
TacQuickAboutBoxLocalization 767
stAbout 767
stOK 767
stRegister 767
stRegisteredTo 768
stUnregistered 768
TacRadioButton 770
Align 771
AlignmentHorz 772
AlignmentVert 773
BoxAlignment 773
Ctl3D 774
OnMouseEnter 775
OnMouseLeave 775
ParentCtl3D 774
WordWrap 775
TacRichLabel 776
Caption 777
MarginHorz 777
MarginVert 777
OnMouseEnter 780
OnMouseLeave 781
ShadowColor 777
ShowFrame 778
ShowShadow 778
TextStyle 778
Transparent 780
WordWrap 780
TacRSSFeed 437
TacRSSFeedContainer 438
TacScreenMagnet 303
Area 304
Edges 304
ScreenPower 305
TacSendMail 438
Abort 453
AddHeaders 439
Agent 439
Attach 440
AttachType 440
Authentication 441
Busy 442
CantAttach 454
FromAddress 443
FromName 443
FromOrganization 443
MsgBody 443
MsgContentCharset 444
MsgContentType 444
MsgDateTime 445
MsgPriority 445
MsgSubject 445
OnAborted 453
OnAnyError 454
OnConnLost 454
OnHostUnreachable 455
OnProgress 455
OnResponse 456
OnSMTPError 457
OnSuccess 457
OnWaitTimeoutExpired 457
ReplyTo 446
SaveToFile 452
Send 453
SMTPHost 446
SMTPPort 446
Suspended 447
TemplateConverter 447
Thread 447
Threaded 448
ThreadPriority 448
ToAddr 449
ToBCC 450
ToCC 451
WaitThread 451
WaitTimeout 452
TacSendMailAuthentication 441
Enabled 441
Password 442
Username 442
TacShortcut 459
FILENAME 460
HotKey 460

© 1998-2006, UtilMind Solutions®

TacShortcut 459
IconIndex 461
IconLocation 461
LargeIcon 461
LinkTarget 461
LoadFromFile 463
Parameters 462
RunAs 462
SaveToFile 464
ShortDescription 462
SmallIcon 463
WorkingDirectory 463
TacSimpleTCPClient 781
AllowChangeHostAndPortOnConnection 781
AutoReconnect 781
AutoReconnectInterval 781
Connected 782
Host 782
OnConnected 783
OnDataAvailable 783
OnDisconnected 783
OnError 784
OnRead 784
OnSelfDisconnected 784
Port 782
Receive 783
ReceiveStream 783
Send 782
SendStream 783
TacSimpleTCPServer 784
AllowChangeHostAndPortOnConnection 785
Broadcast 787
BroadcastStream 787
Listen 785
LocalHostName 785
LocalIP 786
OnAccept 787
OnClientConnected 788
OnClientDataAvailable 788
OnClientDisconnected 788
OnClientRead 789
OnError 789
Port 786
Receive 787
ReceiveStream 787
Send 786
SendStream 786
TacSplitterPanel 789
Align 790
Background 790
ClipSplitRect 794
CursorHorz 794
CursorVert 794
MinSize 794
OnPaint 796
OnSplitterPosChanged 797
ProportionalPanes 795
RegistrySaver 795
SplitterPos 796
SplitterStyle 795
SplitterWidth 796
TacSQLQueryThread 797
Abort 800
OnDone 800
OnException 801
OnWaitTimeoutExpired 801
Priority 797
Query 798
RunSQL 799
WaitThread 798
WaitTimeout 799
TacStatusControl 801
Control 802
PanelIndex 802
StatusBar 802
TacStringListContainer 808
Add 808
AddStrings 809
Clear 810
Count 803
Delete 810
DeleteFirstItem 810
Duplicates 803
Exchange 810
GetAndDeleteItem 811
GetFIFO 811
IndexOf 811
IndexOfName 812
IndexOfValue 812
Insert 813
List 804
LoadFromFile 813
LoadFromRegistry 814
MaxSize 804
Names 806
NVSeparator 804
NVSeparatorMainSide 805
OnChange 817

© 1998-2006, UtilMind Solutions®

TacStringListContainer 808
OnChanging 817
SaveToFile 815
Sort 815
Sorted 805
StoreToRegistry 816
Strings 807
Values 807
TacSystemImageList 818
ImageSize 818
TacSystemInfo 819
Allow_Edit 820
ComputerDescription 820
ComputerName 820
ComputerWorkgroup 821
CPUArchitecture 821
CPUFrequency 821
CPUIdentifier 822
CPUsNumber 822
CPUType 822
CPUVendor 822
CurrentUserName 823
DefaultBrowser 823
DirAppData 823
DirAppDataAll 823
DirCache 824
DirCommonFiles 824
DirCookies 824
DirDesktop 824
DirFavorites 825
DirFonts 825
DirHistory 825
DirMedia 825
DirMyPictures 826
DirNetHood 826
DirPersonal 826
DirPrintHood 826
DirProgramFiles 827
DirPrograms 827
DirRecent 827
DirSendTo 827
DirStartMenu 828
DirStartup 828
DirStartupAll 828
DirSystem 829
DirTemp 829
DirTemplates 829
DirWindows 829
DiskCSerialNumber 830
DiskCSerialNumberStr 830
InternetStatus 830
MemAvailPhisical 831
MemAvailVirtual 831
MemLoad 831
MemTotalPhisical 831
MemTotalVirtual 832
OSVersion 832
OwnerCompany 833
OwnerName 833
ScreenColorMode 833
ScreenResolution 834
ScreenSaver 834
SystemPathes 834
SystemUpTime 835
WinDirectXVersion 835
WinNT 835
WinProductID 835
WinProductKey 836
WinProductName 836
WinVersion 836
WinVersionBuild 837
WinVersionMajor 837
WinVersionMinor 837
WinVersionNumber 837
WinVersionSubNumber 837
TacSystemMenu 208
ApplyToMenu 209
Caption 209
Position 210
Separators 210
TacTextStyle 277
PlainText 278
TagClose 278
TagOpen 279
TacTextTemplateConverter 838
Convert 841
ConvertStrings 842
ConvertToFile 843
IgnoreCaseOfParams 839
OnParamChanged 845
OnTemplateChanged 845
ParamByName 843
ParamByValue 844
Params 840
Stream 842
Template 841
TacThread 464
Execute 469

© 1998-2006, UtilMind Solutions®

TacThread 464
Handle 465
OnException 472
OnExecute 472
OnTerminate 474
OnWaitTimeoutExpired 474
Priority 465
Resume 469
ReturnValue 466
Running 466
Suspend 469
Suspended 467
Synchronize 470
SynchronizeEx 470
Terminate 471
Terminated 467
ThreadID 467
WaitFor 472
WaitThread 468
WaitTimeout 468
TacThread.ThreadHandleExceptions 465
TacThreadSafeLog 474
ClearLog 484
DateTimeFormat 475
EmailLastRecords 486
FileName 477
Flush 485
FlushBuffers 477
GetLastRecords 486
Log 485
LogToFile 478
LogToMemo 478
MaxSize 479
Memo 479
OnCustomPrefix 486
OnFileWriteError 487
OnMaxSizeReached 487
Options 479
SendMail 480
Limit 481
MsgContentCharset 481
MsgFromAddress 481
MsgFromName 482
MsgPriority 482
MsgSubject 482
MsgToAddress 483
MsgToName 483
SMTPHost 483
SMTPPort 483
Threaded 484
TacThreadSafeLogSendMailProperties 480
TacTrackBar 845
TacTrayIcon 488
Animate 489
AnimateInterval 490
CaptionButton 490
HideInfoBalloon 507
Hint 493
Icon 494
Icon16 494
ImageIndex 495
ImageList 495
IsMinimized 496
MinimizeToTray 505
OnAnimate 507
OnBalloonHide 508
OnBalloonShow 508
OnBalloonTimeOut 508
OnBalloonUserClick 509
OnCaptionDblClick 507
OnClick 508
OnDblClick 509
OnHide 509
OnMinimize 510
OnMouseDown 510
OnMouseMove 510
OnMouseUp 511
OnPopupMenu 511
OnRestore 512
OnShow 512
Options 496
PopupMenu 497
PopupMenuRestoreItem 497
PopupMenuTrayMinimizeItem 498
PopupMenuWhatsThisItem 498
RestoreFromTray 506
ShowHint 499
ShowInfoBalloon 506
StartMinimized 499
Visible 502
WhatsThisHint 502
Win2kInfoBalloon 503
TacTreeComboBox 845
AcceptOnDblClick 846
AddPath 867
AttachedLabel 847
AutoComplete 856
AutoCompleteDelay 856

© 1998-2006, UtilMind Solutions®

TacTreeComboBox 845
AutoCompleteIgnoreCase 857
Button 857
ColorDisabled 860
CursorBorder 861
DropDownCount 861
DropDownWidth 861
EmptyItemImageIndex 862
EmptyItemText 862
ExportToTreeNodes 868
Images 862
ImportFromTreeNodes 869
ImportFromTreeView 869
ItemIndex 863
Items 863
OnAfterDropDown 870
OnButtonClick 870
OnButtonLeftMouseDown 871
OnCanSelectNode 871
OnCloseUp 871
OnDropDown 872
OnLabelClick 872
OnLabelDblClick 873
OnMouseEnter 873
OnMouseLeave 873
SelectedNode 864
SelectNodeByText 869
SetSelection 870
ShowTreePathInEdit 864
StateImages 865
TreeOptions 865
TreePathSeparator 866
TreeView 867
TacTreeView 874
AlwaysExpandRoot 875
AlwaysFullExpand 875
AutoExpand 875
BoldItem 876
Cursors 882
ExpandTopLevelNodes 885
FindAbsoluteIndexByNode 885
FindNode 885
HotTrack 883
ItemPopupMenu 883
OnBoldItemBeforeSwitch 886
OnBoldItemSwitched 887
OnMouseEnter 887
OnMouseLeave 887
RowSelect 884
ToolTips 884
TacTreeViewBoldItem 876
BoldNode 877
ColorBack 877
ColorFrame 878
ColorText 878
ImageIndex 878
ImageList 879
ImageSpacing 879
ItemIndex 880
ItemIndexIs 880
ShowFrame 881
SwitchSelectionOnBold 881
SwitchToBoldOnDblClick 881
Text 882
TacTreeViewCursors 882
Border 882
ScrollHorz 883
ScrollVert 883
TacUnixCrypt 888
Key 889
Result 889
Salt 889
TacURLLabal 889
TacURLLabel 890
Email 890
Kind 893
URL 893
URLInNewWindow 893
TacWavPlayer 512
Asynchronous 513
Looped 513
OnAfterPlay 515
OnBeforePlay 515
Play 514
SoundType 513
Stop 514
WaveSound 514
TacWebImage 894
Alignment 894
CacheOptions 894
ControlCanvas 895
DrawRect 895
FixedStretching 895
HTTPAgent 895
HTTPPassword 895
HTTPProxy 896
HTTPUsername 896
InternetOptions 899

© 1998-2006, UtilMind Solutions®

TacWebImage 894
JPEGProperties 895
OnAborted 901
OnDone 901
OnDownloadProgress 901
OnError 901
OnPasswordRequest 901
OnProxyAuthenticationRequest 902
OnUnsupportedFormat 903
ThreadPriority 900
URL 900
TacWhatsThis 279
Caption 280
Enabled 280
MenuItem 281
ToolbarButton 281
TacWin2kAnimation 518
Enabled 518
Flags 518
Time 519
TacWin2kEffects 515
AnimateOnHide 516
AnimateOnShow 518
Flash 521
FlashBackground 522
FlashCount 520
FlashTaskIcon 520
OnHideEffect 522
OnShowEffect 522
Translucent 520
TranslucentRate 521
TacWin2kInfoBalloon 503
Enabled 503
Icon 504
NoSound 504
Text 504
Timeout 505
Title 505
Tags of acFormHelp component 286
TAutoExtract 253
Enabled 254
ExtractTo 254
Overwrite 257
RemoveOnTerminate 257
TExtractTo 254
CreatePath 255
SubDirectory 255
ToplevelDir 256
TStyleFont 660
Color 660
Style 661
Used 662

- U -

UpdateBar 117
TacAppBar 117
Upgrade Information File 191
URLDecode 934
URLEncode 933
URLToFileName 956
URLToHostName 955
Using tags in context-sensitive help 286

- V -

Version History 60

- W -

Wait 967
Welcome 48
WinToDOS 971
WinToKOI 972

- X -

XORStr 966

© 1998-2006, UtilMind Solutions®