

CodeWarrior™

Development Tools

PowerPlant Reference

Revised 8/12/03

Metrowerks, the Metrowerks insignia, and CodeWarrior are registered trademarks of Metrowerks Corp. in the US and/or other countries. All other trade names, trademarks and registered trademarks are the property of their respective owners.

© Copyright. 2002. Metrowerks Corp. ALL RIGHTS RESERVED.

Metrowerks reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Metrowerks does not assume any liability arising out of the application or use of any product described herein. Metrowerks software is not authorized for and has not been designed, tested, manufactured, or intended for use in developing applications where the failure, malfunction, or any inaccuracy of the application carries a risk of death, serious bodily injury, or damage to tangible property, including, but not limited to, use in factory control systems, medical devices or facilities, nuclear facilities, aircraft or automobile navigation or communication, emergency systems, or other applications with a similar degree of potential hazard.

Documentation stored on electronic media may be printed for personal use only. Except for the forgoing, no portion of this documentation may be reproduced or transmitted in any form or by any means, electronic or mechanical, without prior written permission from Metrowerks.

ALL SOFTWARE, DOCUMENTATION AND RELATED MATERIALS ARE SUBJECT TO THE METROWERKS END USER LICENSE AGREEMENT FOR SUCH PRODUCT.

How to Contact Metrowerks:

Corporate Headquarters	Metrowerks Corporation 9801 Metric Blvd. Austin, TX 78758 U.S.A.
World Wide Web	http://www.metrowerks.com
Ordering & Technical Support	Voice: (800) 377-5416 Fax: (512) 997-4901

Table of Contents

LAction	65
LAction()	66
~LAction()	66
CanRedo().	67
CanUndo()	67
Finalize()	67
GetDescription()	68
IsDone()	68
IsPostable()	68
Redo()	69
RedoSelf().	69
Undo()	69
UndoSelf()	70
operator=()	70
mStringResID	70
mStringIndex	71
mIsDone	71
 LActiveScroller	 73
LActiveScroller()	74
~LActiveScroller()	75
ActiveThumbScroll()	75
AdaptToNewThumbValue()	76
AdjustScrollBars()	76
AssignThumbProcs()	76
CalcValueFromPoint()	77
EndThumbTracking()	77
HandleThumbScroll().	78
IsTrackingThumb()	78
ListenToMessage()	78
StartThumbTracking()	79
mThumbControl	79
mTrackBarUnits	80
mTrackBarPin	80
mTrackBarSize	80

Table of Contents

mOriginalValue	80
mTrackRect	81
mValueSlop	81
mVertThumbAction	81
mHorizThumbAction.	81

LApplication **83**

LApplication()	84
~LApplication()	84
CountSubModels()	84
DoQuit()	85
FindCommandStatus()	85
GetPositionOfSubModel()	86
GetState()	87
GetSubModelByName().	87
GetSubModelByPosition().	88
HandleAppleEvent()	89
Initialize()	89
MakeMenuBar()	90
MakeModelDirector().	90
MakeSelfSpecifier().	90
ObeyCommand().	91
ProcessNextEvent().	92
Run()	92
SendAEQuit()	92
SetSleepTime().	93
ShowAboutBox().	93
Startup()	93
mState	94
mSleepTime	94

LArray **95**

LArray()	97
~LArray()	98
AddItem().	98
AdjustAllocation()	99
AdjustStorage()	99

AssignItemsAt()	100
AttachIterator()	101
BinarySearch()	101
BinarySearchByKey()	102
CopyArray()	102
DestroyArray()	103
DetachIterator()	103
FetchIndexOf()	104
FetchIndexOfKey()	104
FetchInsertIndexOf()	104
FetchInsertIndexOfKey()	105
FetchItemAt()	105
GetComparator()	106
GetCount()	106
GetItemPtr()	107
GetItemSize()	107
GetItemsHandle()	108
GrabItemRangeSize()	108
GrabItemSize()	109
InitArray()	109
InsertItemsAt()	110
InternalAdjustAllocation()	111
InternalCopyItem()	111
InternalMoveItem()	112
InternalSwapItems()	112
InvalidateSort()	113
IsKeptSorted()	113
IsSorted()	113
ItemsInserted()	114
ItemsRemoved()	114
LinearSearch()	115
LinearSearchByKey()	115
Lock()	116
MoveItem()	116
PeekItem()	117
PokeItem()	117
Remove()	118

Table of Contents

RemoveItemsAt()	118
SetComparator()	119
SetKeepSorted()	119
ShiftItems()	120
Sort()	120
StoreNewItem()	121
SwapItems()	121
Unlock()	122
ValidIndex()	122
operator=()	123
mComparator	123
mDataAllocated	123
mDataStored	124
mIsSorted	124
mItemCount	124
mItemsH	124
mItemSize	125
mIteratorListHead	125
mKeepSorted	125
mLockCount	125
mOwnsComparator	126

LArrayIterator **127**

LArrayIterator()	128
~LArrayIterator()	128
ArrayDied()	128
CalcNextIndex()	129
CalcPreviousIndex()	129
Current()	129
GetCurrentIndex()	130
GetNextIterator()	131
ItemsInserted()	131
ItemsRemoved()	131
Next()	132
Previous()	133
PtrToCurrent()	133
PtrToNext()	134

PtrToPrevious()	134
ResetTo()	135
SetNextIterator()	135
mNextIterator	136
mArray	136
mCurrIndex	136
mNextIndex	137
LAttachable	139
LAttachable()	139
~LAttachable()	140
AddAttachment()	140
ExecuteAttachments()	141
GetDefaultAttachable()	141
RemoveAllAttachments()	141
RemoveAttachment()	142
SetDefaultAttachable()	142
sDefaultAttachable	143
mAttachments	143
LAttachment	145
LAttachment()	146
~LAttachment()	146
Execute()	147
ExecuteSelf()	147
GetExecuteHost()	148
GetMessage()	148
GetOwnerHost()	149
SetExecuteHost()	149
SetMessage()	149
SetOwnerHost()	150
mOwnerHost	150
mMessage	150
mExecuteHost	150
LBeepAttachment	153
LBeepAttachment()	153
ExecuteSelf()	154

LBorderAttachment	155
LBorderAttachment()	155
ExecuteSelf().	156
mPenState.	156
mForeColor	156
mBackColor	157
 LBroadcaster	 159
LBroadcaster()	160
~LBroadcaster()	160
AddListener()	160
BroadcastMessage()	161
IsBroadcasting()	162
RemoveListener()	162
StartBroadcasting()	162
StopBroadcasting()	163
mListeners	163
mIsBroadcasting	163
 LButton	 165
LButton()	166
DrawGraphic().	167
DrawSelf()	167
HotSpotAction()	167
HotSpotResult()	167
PointIsInFrame()	168
SetGraphics()	168
SetGraphicsType()	168
mGraphicsType	169
mNormalID	169
mPushedID	169
 LCaption	 171
LCaption()	172
~LCaption()	172
DrawSelf()	173
GetDescriptor()	173
GetTextTraitsID().	173

	GetValue().	174
	SetDescriptor().	174
	SetTextTraitsID().	174
	SetValue().	175
	mText.	175
	mTxtrID	175
LCicnButton		177
	LCicnButton().	177
	~LCicnButton().	178
	DrawSelf().	178
	HotSpotAction().	179
	HotSpotResult().	179
	SetCicns().	179
	mNormalID	179
	mPushedID	180
	mNormalCicnH	180
	mPushedCicnH	180
LCleanupTask		181
	LCleanupTask().	181
	~LCleanupTask().	182
	CleanUpAtExit().	182
	DoCleanup().	182
	ETSPatch().	183
	sCleanupTaskList	183
	sOldETSRoutine	183
	sNewETSRoutine	183
LClipboard		185
	LClipboard().	186
	~LClipboard().	186
	ExecuteSelf().	186
	ExportSelf().	186
	GetClipboard().	187
	GetData().	187
	GetDataSelf().	188
	ImportSelf().	189

Table of Contents

SetData()	189
SetDataSelf()	190
sClipboard	191
mImportPending	191
mExportPending	191
LCmdEnablerAttachment	193
LCmdEnablerAttachment()	193
ExecuteSelf()	194
mCmdToEnable	194
LColorEraseAttachment	195
LColorEraseAttachment()	195
ExecuteSelf()	196
mForeColor	196
mBackColor	196
LCommander	197
LCommander()	198
~LCommander()	199
AddSubCommander()	199
AllowSubRemoval()	199
AllowTargetSwitch()	200
AttemptQuit()	200
AttemptQuitSelf()	201
BeTarget()	201
DontBeTarget()	202
FindCommandStatus()	202
GetDefaultCommander()	203
GetLatentSub()	203
GetSuperCommander()	204
GetTarget()	204
GetTopCommander()	204
GetUpdateCommandStatus()	205
HandleKeyPress()	205
InitCommander()	205
IsOnDuty()	206
IsSyntheticCommand()	206

IsTarget()	207
ObeyCommand().	207
PostAction().	208
PostAnAction()	208
ProcessCommand().	209
ProcessCommandStatus().	209
ProcessKeyPress().	210
PutChainOnDuty().	211
PutOnDuty()	211
RemoveSubCommander().	211
RestoreTarget().	212
SetDefaultCommander()	212
SetLatentSub().	213
SetSuperCommander()	213
SetTarget().	214
SetUpdateCommandStatus().	214
SwitchTarget().	214
TakeChainOffDuty()	215
TakeOffDuty().	216
sTopCommander.	216
sTarget	216
sDefaultCommander	217
sUpdateCommandStatus	217
mSuperCommander	217
mSubCommanders	217
mOnDuty	218

LCommanderPane 219

LCommanderPane()	219
~LCommanderPane().	220

LComparator 221

LComparator().	221
~LComparator()	222
Clone().	222
Compare().	222
CompareToKey().	223

Table of Contents

GetComparator().	224
IsEqualTo().	224
IsEqualToKey().	225
sComparator	225

LControl **227**

LControl().	228
~LControl().	229
BroadcastValueMessage().	229
ClickSelf().	229
FindHotSpot().	230
GetMaxValue().	230
GetMinValue().	230
GetValue().	231
GetValueMessage().	231
HotSpotAction().	231
HotSpotResult().	232
IncrementValue().	232
PointInHotSpot().	233
SetMaxValue().	233
SetMinValue().	234
SetValue().	234
SetValueMessage().	234
SimulateHotSpotClick().	235
TrackHotSpot().	235
mValueMessage	236
mValue	236
mMinValue	236
mMaxValue	237

LDataArrived **239**

LDataArrived().	239
~LDataArrived().	240
GetDataBuffer().	240
GetDataSize().	241
GetRemoteAddress().	241
mDataBuffer.	241

mDataSize	241
mRemoteAddress	242
mMustReleaseMemory	242
LDataStream	243
LDataStream()	243
~LDataStream()	244
GetBuffer()	244
SetBuffer().	245
GetBytes().	245
PutBytes().	246
operator =.	246
mBuffer	247
LDefaultOutline	249
LDefaultOutline()	249
DrawSelf()	250
LDialogBox	251
LDialogBox()	252
~LDialogBox()	253
FinishCreateSelf()	253
HandleKeyPress()	253
ListenToMessage()	253
SetCancelButton()	254
SetDefaultButton()	254
mDefaultButtonID	255
mCancelButtonID	255
mDefaultOutline	255
LDocApplication	257
LDocApplication()	258
~LDocApplication()	258
ChooseDocument().	258
CountSubModels().	259
DoAEOpenOrPrintDoc()	259
FindCommandStatus()	260
GetPositionOfSubModel().	260

Table of Contents

GetSubModelByName().	261
GetSubModelByPosition().	261
HandleAppleEvent().	262
HandleCreateElementEvent().	262
MakeNewDocument().	263
ObeyCommand().	263
OpenDocument().	263
PrintDocument().	263
SendAECreatDocument().	264
SendAEOpenDoc().	264
SetupPage().	264

LDocument **265**

LDocument().	266
~LDocument().	266
AskSaveAs().	266
AttemptClose().	267
AttemptQuitSelf().	267
Close().	268
DoAEClose().	268
DoAESave().	269
DoPrint().	269
DoRevert().	269
DoSave().	270
FindByFileSpec().	270
FindCommandStatus().	271
FindNamedDocument().	271
GetAEPProperty().	271
GetDescriptor().	272
GetDocumentList().	272
HandleAppleEvent().	273
IsModified().	273
MakeCurrent().	274
MakeSelfSpecifier().	274
ObeyCommand().	274
SendAEClose().	275
SendAESaveAs().	276

	SetModified()	276
	UsesFileSpec()	277
	sDocumentList.	277
	mPrintRecordH	277
	mIsModified.	278
	mIsSpecified.	278
LDragAndDrop		279
	LDragAndDrop().	279
	FocusDropArea().	280
	HiliteDropArea().	280
	PointInDropArea()	281
	mPane	281
LDragTask		283
	LDragTask()	283
	~LDragTask()	285
	AddFlavors()	285
	AddRectDragItem()	285
	DoDrag()	286
	DropLocationIsFinderTrash()	286
	GetDragReference().	287
	GetDragRegion().	287
	InitDragTask()	287
	MakeDragRegion()	288
	mDragRef.	288
	mDragRegion	288
	mEventRecord.	289
LDropArea		291
	LDropArea().	292
	~LDropArea()	292
	AddDropArea()	293
	DoDragDrawing()	293
	DoDragInput().	294
	DoDragReceive().	295
	DoDragSendData().	295
	DragAndDropIsPresent()	296

Table of Contents

DragIsAcceptable()	296
EnterDropArea()	297
FindDropArea()	297
FocusDropArea()	298
HandleDragDrawing()	298
HandleDragInput()	299
HandleDragReceive()	300
HandleDragSendData()	301
HandleDragTracking()	301
HiliteDropArea()	302
InTrackingWindow()	302
InsideDropArea()	303
InstallHandlers()	303
ItemIsAcceptable()	304
LeaveDropArea()	304
PointInDropArea()	305
ReceiveDragItem()	305
RemoveDropArea()	306
UnhiliteDropArea()	307
mDragWindow	307
mCanAcceptCurrentDrag	307
mIsHilited	308
sDragTrackingProc	308
sDragReceiveProc	308
sDropAreaList	308
sCurrentDropArea	309
sDragHasLeftSender	309

LEditField **311**

LEditField()	312
~LEditField()	313
AdjustCursorSelf()	314
AdjustTextWidth()	314
AlignTextEditRects()	315
BeTarget()	315
ClickSelf()	315
DisableSelf()	315

DontBeTarget().	315
DrawBox()	316
DrawSelf()	316
EnableSelf()	316
FindCommandStatus()	316
FocusDraw().	317
GetDescriptor()	317
GetMacTEH()	317
GetValue().	317
HandleKeyPress()	318
HideSelf()	318
InitEditField()	318
MoveBy()	318
ObeyCommand().	319
ResizeFrameBy()	319
RestorePlace()	319
SavePlace()	319
SelectAll()	319
SetDescriptor().	320
SetKeyFilter()	320
SetMaxChars()	320
SetTextTraitsID()	321
SetValue()	321
SpendTime().	321
TooManyCharacters().	322
UseWordWrap()	322
UserChangedText().	322
mTextEditH	323
mKeyFilter	323
mTypingAction	323
mMaxChars	324
mTextTraitsID	324
mHasBox	324
mHasWordWrap	324

LEndpoint**325**

LEndpoint()	326
-----------------------	-----

Table of Contents

~LEndpoint()	326
AbortThreadOperation()	326
AckSends()	327
Bind()	327
DontAckSends()	328
DontQueueSends()	328
GetLocalAddress()	328
GetState()	329
IsAckingSends()	329
IsQueueingSends()	329
QueueSends()	330
Unbind()	330
mQueueSends	330

LEraseAttachment **331**

LEraseAttachment()	331
ExecuteSelf()	332

LEventDispatcher **333**

LEventDispatcher()	334
~LEventDispatcher()	334
AdjustCursor()	334
ClickMenuBar()	335
DispatchEvent()	335
EventActivate()	336
EventAutoKey()	336
EventDisk()	337
EventHighLevel()	337
EventKeyDown()	338
EventKeyUp()	338
EventMouseDown()	339
EventMouseUp()	339
EventOS()	339
EventResume()	340
EventSuspend()	340
EventUpdate()	341
ExecuteAttachments()	341

GetCurrentEventDispatcher()	342
UpdateMenus()	342
UseIdleTime()	343
sCurrentDispatcher.	343
mSaveDispatcher.	344
mMouseRgnH.	344
LEventSemaphore	345
LEventSemaphore()	345
~LEventSemaphore()	346
Reset()	346
Signal()	346
mPostCount.	347
LFile	349
LFile()	350
~LFile()	350
CloseDataFork()	351
CloseResourceFork()	351
CreateNewDataFile()	352
CreateNewFile()	352
EqualFileSpec()	353
GetSpecifier()	353
MakeAlias()	354
OpenDataFork()	354
OpenResourceFork()	355
ReadDataFork()	355
SetSpecifier()	356
UsesSpecifier()	356
WriteDataFork()	356
mMacFileSpec	357
mDataForkRefNum	357
mResourceForkRefNum.	357
LFileStream	359
LFileStream()	359
~LFileStream()	360
GetLength()	360

Table of Contents

SetLength()	361
GetMarker()	361
SetMarker()	361
GetBytes()	362
PutBytes()	362

LFocusBox **365**

LFocusBox()	365
AttachPane()	366
DontRefresh()	366
DrawSelf()	367
GetBoxRegion()	367
HideSelf()	367
Refresh()	368
ShowSelf()	368

LGlobalsContext **369**

LGlobalsContext()	369
~LGlobalsContext()	370
GetGlobals()	370
mSavedGlobals	370

LGrafPortView **371**

LGrafPortView()	372
~LGrafPortView()	372
Activate()	373
ApplyForeAndBackColors()	373
ClickInContent()	374
CreateGrafPortView()	374
Deactivate()	375
DispatchCommand()	375
DoIdle()	375
DoKeyPress()	376
Draw()	376
DrawSelf()	377
EstablishPort()	377
GetMacPort()	378
InitGrafPortView()	378

	InvalPortRect().	378
	InvalPortRgn().	379
	SetForeAndBackColors()	379
	UpdatePort()	380
	ValidPortRect().	380
	ValidPortRgn().	381
	mGrafPtr	381
	mForeColor	381
	mBackColor	382
LGroupBox		383
	LGroupBox()	383
	~LGroupBox()	384
	DrawSelf()	384
	DrawText()	385
	DrawBorder()	385
	CalcTextBoxFrame()	386
	mFrameColor	386
LGrowZone		387
	LGrowZone()	389
	~LGrowZone().	389
	AskListenersToFree()	390
	DoGrowZone().	390
	GetGrowZone()	391
	GiveWarning()	391
	GrowZoneCallBack()	392
	MemoryIsLow()	392
	SpendTime().	393
	UseLocalReserve()	394
	sGrowZone	394
	sGrowZoneUPP	394
	mLocalReserve.	395
	mReserveSize	395
	mGiveWarning	395
LGWorld		397
	LGWorld()	397

Table of Contents

~LGWorld()	398
BeginDrawing()	398
CopyImage()	399
EndDrawing()	399
GetBounds()	400
GetMacGWorld()	400
SetBounds()	400
mMacGWorld	401
mBounds	401
mSavePort	401
mSaveDevice	402
LHandleStream	403
LHandleStream()	403
~LHandleStream()	404
DetachDataHandle()	404
GetBytes()	404
GetDataHandle()	405
PutBytes()	405
SetDataHandle()	406
SetLength()	406
mDataH	407
LIconPane	409
LIconPane()	409
~LIconPane()	410
DrawSelf()	410
SetIconID()	410
mIconID	411
LInternetAddress	413
LInternetAddress()	413
~LInternetAddress()	414
Clone()	415
GetDNSAddress()	415
GetDNSDescriptor()	415
GetHostPort()	416
GetIPAddress()	416

GetIPAddress().	417
GetIPDescriptor()	417
InternalLookupAddress()	418
InternalLookupName()	418
MakeOTDNSAddress()	418
MakeOTIPAddress()	419
SetDNSAddress()	419
SetHostPort()	419
SetIPAddress()	420
mIPAddress	420
mDNSAddress.	420
mHostPort	421
LInternetMapper	423
LInternetMapper()	424
~LInternetMapper()	424
AbortThreadOperation()	424
AddressToName()	425
GetLocalAddress()	425
NameToAddress()	425
LInterruptSafeList	427
LInterruptSafeList()	427
~LInterruptSafeList()	428
Append()	428
IsEmpty()	428
Remove()	429
mQueue	429
mIteratorQueue	429
LKeyScrollAttachment	431
LKeyScrollAttachment().	432
ExecuteSelf().	432
mViewToScroll.	432
LLink	433
LLink()	433
~LLink()	434

Table of Contents

GetLink()	434
SetLink()	434
mLink	434

LListBox **437**

LListBox()	438
~LListBox()	439
ActivateSelf()	439
BeTarget()	439
ClickSelf()	440
DeactivateSelf()	440
DoNavigationKey().	440
DoTypeSelection()	440
DontBeTarget().	441
DrawSelf()	441
FindCommandStatus()	441
FocusDraw().	441
GetDescriptor()	442
GetDoubleClickMessage().	442
GetFocusBox()	442
GetLastSelectedCell()	442
GetMacListH().	443
GetValue().	443
HandleKeyPress()	444
HideSelf()	444
InitListBox()	444
MakeCellVisible()	445
MoveBy()	445
ObeyCommand().	445
ResizeFrameBy()	446
RestorePlace()	446
SavePlace()	446
SelectAllCells().	447
SelectOneCell()	447
SetDescriptor().	447
SetDoubleClickMessage()	448
SetValue()	448

ShowSelf().	448
UnselectAllCells()	449
mMacListH	449
mDoubleClickMessage	449
mFocusBox	449
mTextTraitsID	450
mHasGrow	450
LListener	451
LListener()	451
~LListener()	452
AddBroadcaster()	452
IsListening()	453
ListenToMessage()	453
RemoveBroadcaster()	454
StartListening()	454
StopListening()	455
mBroadcasters	455
mIsListening.	455
LLongComparator	457
LLongComparator()	457
~LLongComparator()	458
Clone()	458
Compare().	458
GetComparator().	459
IsEqualTo()	459
sLongComparator	460
LMacTCPDNSOperation	461
LMacTCPDNSOperation().	461
~LMacTCPDNSOperation()	462
GetResultProc()	462
Int_DNSCompletionProc().	462
Int_DNSCompletionProc2()	463
WaitForCompletion()	463
mOperationListMem	463
sMacTCPDNSCompletionProc.	464

sMacTCPDNSPendingOperations;	464
sDNSOperationDeleteQueue	464
LMacTCPTCPSendQueue	465
LMacTCPTCPSendQueue()	465
~LMacTCPTCPSendQueue().	466
Int_InternalSend()	466
NotifyRelease()	466
mWDS	467
mMacTCPEndpoint	467
mOperation	467
LMacTCPUDPEndpoint	469
LMacTCPUDPEndpoint()	470
~LMacTCPUDPEndpoint()	470
AbortThreadOperation()	470
AckSends()	471
Bind().	471
DontAckSends()	471
GetLocalAddress()	472
GetState()	472
Int_HandleAsyncEvent()	472
Int_UDPNotifyProc()	473
IsAckingSends()	474
ReceiveFrom()	474
SendPacketData()	475
Unbind()	475
mUDPStream	475
mLocalAddress	476
mAckSends	476
mEndpointState	476
mReceiveBuffer	476
mReceiveBufferSize	477
mMessageQueue.	477
mSharedPool	477
mSendQueue	477
sUDPNotifyUPP	478

LMacTCPUDPSendQueue	479
LMacTCPUDPSendQueue()	479
~LMacTCPUDPSendQueue()	480
Int_InternalSend()	480
NotifyRelease()	480
mWDS	481
mMacTCPUDPEndpoint	481
mOperation	481
 LMenu	 483
LMenu()	484
~LMenu()	484
CommandFromIndex()	484
FindNextCommand()	485
GetMacMenuH()	485
GetMenuID()	486
GetNextMenu()	486
IndexFromCommand()	486
InsertCommand()	487
IsInstalled()	487
IsUsed()	488
ItemIsEnabled()	488
ReadCommandNumbers()	488
RemoveCommand()	489
RemoveItem()	489
SetCommand()	490
SetInstalled()	490
SetNextMenu()	490
SetUsed()	491
SyntheticCommandFromIndex()	491
mNextMenu	492
mMacMenuH	492
mMENUId	492
mNumCommands	492
mCommandNums	493
mIsInstalled	493
mUsed	493

LMenuBar	495
LMenuBar()	495
~LMenuBar()	496
CouldBeKeyCommand()	496
FetchMenu()	496
FindCommand()	497
FindKeyCommand()	497
FindMenuItem()	498
FindNextCommand()	499
FindNextMenu()	499
GetCurrentMenuBar()	500
InstallMenu()	500
MenuCommandSelection()	500
RemoveMenu()	501
sMenuBar	501
mMenuListHead	502
 LMouseTracker	 503
LMouseTracker()	503
~LMouseTracker()	503
SpendTime()	504
 LMovieController	 505
LMovieController()	505
~LMovieController()	506
DrawSelf()	506
SpendTime()	506
mMovie	506
mMovieController	507
 LMutexSemaphore	 509
LMutexSemaphore()	509
~LMutexSemaphore()	510
Signal()	510
Wait()	510
mOwner	510
mNestedWaits	511

LNetMessageQueue	513
LNetMessageQueue(). 513
~LNetMessageQueue(). 514
SpendTime(). 514
mBroadcaster 514
 LOffscreenView	 515
LOffscreenView(). 515
~LOffscreenView(). 516
Draw(). 516
DrawOffscreen(). 516
EstablishPort(). 517
SubImageChanged(). 517
mOffscreenWorld 518
mDrawingSelf 518
 LOperationListMember	 519
LOperationListMember(). 519
~LOperationListMember(). 519
mOperation 520
 LOverlappingView	 521
LOverlappingView(). 521
~LOverlappingView(). 522
FocusDraw(). 522
 LPaintAttachment	 523
LPaintAttachment(). 523
ExecuteSelf(). 524
mPenState. 524
mForeColor 525
mBackColor 525
 LPane	 527
LPane(). 530
~LPane(). 530
Activate(). 530
ActivateSelf(). 531

Table of Contents

AdaptToNewSurroundings().	531
AdaptToSuperFrameSize().	531
AdaptToSuperScroll().	532
AdjustCursor().	532
AdjustCursorSelf()	533
ApplyForeAndBackColors()	533
CalcLocalFrameRect().	534
CalcPortFrameRect()	534
Click()	535
ClickSelf().	535
ClickTimesAreClose().	536
Contains().	536
CountPanels()	537
Deactivate()	537
DeactivateSelf()	538
Disable()	538
DisableSelf().	538
DontRefresh()	539
Draw()	539
DrawSelf()	540
Enable().	540
EnableSelf()	541
EventMouseUp().	541
FindConstPaneByID().	542
FindDeepSubPaneContaining()	542
FindPaneByID()	543
FindShallowSubPaneContaining()	543
FindSubPaneHitBy()	544
FinishCreate()	544
FinishCreateSelf()	545
FocusDraw().	545
FocusExposed()	546
GetActiveState()	546
GetClickCount()	547
GetDefaultView()	547
GetDescriptor()	547
GetEnabledState()	548

GetFrameBinding()	548
GetFrameLocation()	548
GetFrameSize()	549
GetLastPaneClicked()	549
GetLastPaneMoused()	549
GetLocalUpdateRgn()	550
GetMacPort()	550
GetPaneID()	551
GetSuperView()	551
GetUserCon()	551
GetValue()	552
GetVisibleState()	552
GlobalToPortPoint()	552
Hide()	553
HideSelf()	553
InitPane()	553
InvalPortRect()	554
InvalPortRgn()	554
IsActive()	555
IsAreaInQDSpace()	555
IsEnabled()	556
IsHitBy()	556
IsVisible()	556
LocalToPortPoint()	557
MouseEnter()	557
MouseLeave()	558
MouseWithin()	558
MoveBy()	559
PlaceInSuperFrameAt()	559
PlaceInSuperImageAt()	560
PointIsInFrame()	561
PointsAreClose()	561
PortToGlobalPoint()	562
PortToLocalPoint()	562
PrintPanel()	563
PrintPanelSelf()	563
PutInside()	564

Table of Contents

Refresh()	. 565
ResizeFrameBy()	. 565
ResizeFrameTo()	. 566
RestorePlace()	. 566
SavePlace()	. 567
ScrollToPanel()	. 567
SetDefaultView()	. 568
SetDescriptor()	. 568
SetForeAndBackColors()	. 568
SetFrameBinding()	. 569
SetLastPaneMoused()	. 569
SetPaneID()	. 570
SetRefreshAllWhenResized()	. 570
SetUserCon()	. 571
SetValue()	. 571
Show()	. 571
ShowSelf()	. 572
SuperActivate()	. 572
SuperDeactivate()	. 572
SuperDisable()	. 573
SuperEnable()	. 573
SuperHide()	. 573
SuperPrintPanel()	. 574
SuperShow()	. 574
UpdateClickCount()	. 575
UpdatePort()	. 575
ValidPortRect()	. 576
ValidPortRgn()	. 576
sDefaultView	. 576
sLastPaneClicked	. 577
sLastPaneMoused	. 577
sWhenLastMouseUp	. 577
sWhenLastMouseDown	. 577
sWhereLastMouseDown	. 578
sClickCount	. 578
mPaneID	. 578
mFrameSize	. 578

	mFrameLocation	579
	mFrameBinding	579
	mUserCon	579
	mSuperView.	579
	mVisible	580
	mActive.	580
	mEnabled	580
	mRefreshAllWhenResized.	580
LPeriodical		581
	LPeriodical().	582
	~LPeriodical()	582
	DeleteIdlerAndRepeaterQueues()	583
	DevoteTimeToIdlers().	583
	DevoteTimeToRepeaters()	584
	SpendTime().	584
	StartIdling()	585
	StartRepeating()	585
	StopIdling()	585
	StopRepeating()	586
	sIdlerQ	586
	sRepeaterQ	586
LPicture		587
	LPicture()	587
	DrawSelf()	588
	GetPictureID()	588
	InitPicture()	589
	SetPictureID()	589
	mPICTid	589
LPlaceholder		591
	LPlaceholder().	591
	~LPlaceholder()	592
	CountPanels()	592
	InstallOccupant().	593
	RemoveOccupant().	594
	ScrollToPanel().	594

Table of Contents

mOccupant	594
mOccupantSuperView	595
mOccupantPlaceH	595
mOccupantAlignment	595
LPreferencesFile	597
LPreferencesFile()	597
OpenOrCreateResourceFork()	598
LPrintout	599
LPrintout()	600
~LPrintout()	600
ApplyForeAndBackColors()	600
CountPanels()	601
CreatePrintout()	601
DoPrintJob()	601
EstablishPort()	602
FinishCreateSelf()	602
GetMacPort()	602
GetPrintJobSpecs()	602
GetPrintRecord()	603
HasAttribute()	603
InitPrintout()	604
PageToPanel()	604
PrintCopiesOfPages()	604
PrintPanel()	605
PrintPanelRange()	605
SetAttribute()	606
SetForeAndBackColors()	606
SetPrintRecord()	607
mAttributes	607
mPrintRecordH	607
mPrinterPort	607
mWindowPort	608
mHorizPanelCount	608
mVertPanelCount	608
mForeColor	608

mBackColor	609
LQueue	611
LQueue()	611
~LQueue()	612
DoForEach()	612
GetSize()	612
IsEmpty()	613
NextGet()	613
NextPut()	613
Remove()	614
operator =()	614
mFirst	614
mLast.	615
mSize.	615
LRadioGroup	617
LRadioGroup().	617
~LRadioGroup()	618
AddRadio()	618
GetCurrentRadioID()	619
ListenToMessage()	619
mCurrentRadio	619
LReentrantMemoryPool	621
LReentrantMemoryPool()	621
~LReentrantMemoryPool()	622
AddPool()	622
AllocFrom()	622
DisposePtr()	623
FreeMem()	623
GetPtrSize()	624
MakeFreeBlock()	624
NewPtr()	624
NewPtrClear()	625
TotalMem()	625
mMemoryPools	626
mFreeBlocks;	626

LScroller	627
LScroller()	628
~LScroller()	629
ActivateSelf()	629
AdjustScrollBars()	629
DeactivateSelf()	630
DrawSelf()	630
ExpandSubPane()	630
FinishCreateSelf()	630
HasHorizontalScrollBar()	631
HasVerticalScrollBar().	631
HorizSBarAction()	631
HorizScroll().	632
InstallView().	632
ListenToMessage()	633
MakeScrollBars().	634
ResizeFrameBy()	634
SubImageChanged()	635
VertSBarAction()	635
VertScroll()	636
mScrollingView	636
mVerticalBar.	637
mHorizontalBar	637
mScrollingViewID	637
 LSemaphore	 639
LSemaphore()	639
~LSemaphore()	640
BlockThread()	640
InitSemaphore()	640
Signal()	641
UnblockAll().	641
UnblockThread().	641
Wait().	642
operator =()	642
mExcessSignals0	643
mThreads	643

LSendQueue	645
LSendQueue()	. 646
~LSendQueue()	. 646
Append()	. 646
GetBlockingDataSize()	. 647
GetMaxPendingRelease()	. 647
Int_InternalSend()	. 647
Int_SendComplete()	. 648
InternalClearReleaseQueue()	. 648
IsBusy()	. 648
KillQueue()	. 649
NotifyRelease()	. 649
Run()	. 649
SetBlockingDataSize()	. 650
SetMaxPendingRelease()	. 650
WaitingDataSize()	. 650
mWaitingQueue	. 651
mPendingQueue	. 651
mReleaseQueue	. 651
mContinue	. 651
mDeleteData	. 652
mEndpointDead	. 652
mBusy	. 652
mReleaseWaiting	. 652
mEndpoint	. 653
mThread	. 653
mMaxPendingRelease	. 653
mBlockingDataSize	. 653
 LSharable	 655
LSharable()	. 655
~LSharable()	. 655
AddUser()	. 656
GetUseCount()	. 656
NoMoreUsers()	. 656
RemoveUser()	. 657
mUseCount	. 657

LSharableModel	659
LSharableModel()	659
~LSharableModel().	660
AddUser().	660
Finalize().	660
SuperDeleted().	661
 LSharedMemoryPool	 663
LSharedMemoryPool().	663
~LSharedMemoryPool().	664
AddPoolUser().	664
GetSharedPool().	664
RemovePoolUser().	665
sSharedMemoryPool	665
 LSharedQueue	 667
LSharedQueue().	667
~LSharedQueue().	668
DoForEach().	668
Next().	668
NextPut().	669
Remove().	669
mAvailable	669
 LSingleDoc	 671
LSingleDoc().	671
~LSingleDoc().	672
AllowSubRemoval().	672
GetDescriptor().	672
GetFile().	672
GetWindow().	672
MakeCurrent().	673
UsesFileSpec().	673
mWindow.	673
mFile	674
 LSIOUXAttachment	 675
LSIOUXAttachment().	675

	ExecuteSelf().	676
LStdButton		677
	LStdButton().	677
	HotSpotResult().	678
LStdCheckBox		679
	LStdCheckBox().	679
	HotSpotResult().	680
LStdControl		681
	LStdControl().	682
	~LStdControl().	684
	AlignControlRect().	684
	CalcBigValue().	684
	CalcSmallValue().	685
	CreateFromCNTL().	685
	DisableSelf().	686
	DrawSelf().	686
	EnableSelf().	687
	FindHotSpot().	687
	FocusDraw().	687
	GetDescriptor().	687
	GetMacControl().	688
	GetTrackingControl().	688
	HideSelf().	688
	HotSpotAction().	688
	HotSpotResult().	689
	InitStdControl().	689
	MoveBy().	690
	PointInHotSpot().	690
	ResizeFrameBy().	690
	SetActionProc().	690
	SetDescriptor().	691
	SetMaxValue().	691
	SetMinValue().	691
	SetStdMinAndMax().	691
	SetThumbFunc().	692

Table of Contents

SetValue()	693
ShowSelf()	693
TrackHotSpot()	693
ValueIsInStdRange()	693
mMacControlH	694
mThumbFunc	694
mControlKind	694
mTextTraitsID	694
mUsingBigValues	695
sTrackingControl	695

LStdPopupMenu **697**

LStdPopupMenu()	697
~LStdPopupMenu()	699
DrawSelf()	699
GetMacMenuH()	699
InitStdPopupMenu()	700

LStdRadioButton **701**

LStdRadioButton()	701
HotSpotResult()	703
SetValue()	703

LStr255 **705**

LStr255()	705
LStr255(const LStr255&)	706
LStr255(const LString&)	706
LStr255(const unsigned char*)	706
LStr255(unsigned char)	707
LStr255(char)	707
LStr255(const char*)	707
LStr255(const void*,unsigned char)	708
LStr255(char**)	708
LStr255(short,short)	708
LStr255(long)	709
LStr255(long double,signed char,short)	709
LStr255(unsigned long)	710
operator=()	710

mString	711
LStream	713
LStream()	714
~LStream()	714
AtEnd()	714
GetBytes()	715
GetLength()	715
GetMarker()	715
PeekData()	716
PutBytes()	716
ReadBlock()	717
ReadCString()	717
ReadData()	718
ReadHandle()	718
ReadPString()	719
ReadPtr()	719
SetLength()	719
SetMarker()	720
WriteBlock()	720
WriteCString()	721
WriteData()	721
WriteHandle()	722
WritePString()	722
WritePtr()	722
LStream& operator << (<i>TypeParameter</i>)	723
LStream& operator >> (<i>TypeParameter</i>)	723
LStream& operator =	724
mMarker	724
mLength	725
LString	727
LString()	728
Append (<i>TypeParameters</i>)	728
LSubOverlapView	747
LSubOverlapView()	747
~LSubOverlapView()	748

	FocusDraw().	748
LTabGroup		749
	LTabGroup().	750
	~LTabGroup().	750
	BeTarget().	750
	GetOnDutySub().	750
	HandleKeyPress().	751
	RotateTarget().	751
LTable		753
	LTable().	754
	~LTable().	754
	ActivateSelf().	754
	ClickCell().	755
	ClickSelf().	755
	DeactivateSelf().	755
	DrawCell().	755
	DrawSelf().	756
	EqualCell().	756
	FetchCellDataIndex().	756
	FetchCellHitBy().	757
	FetchLocalCellFrame().	757
	GetCellData().	758
	GetSelectedCell().	758
	GetTableSize().	758
	HiliteCell().	759
	InitTable().	759
	InsertCols().	760
	InsertRows().	760
	IsValidCell().	761
	RemoveCols().	761
	RemoveRows().	762
	SelectCell().	762
	SetCellData().	763
	SetCellDataSize().	763
	SetColWidth().	763

SetRowHeight()	.764
UnhiliteCell()	.764
mRows	.765
mCols	.765
mRowHeight	.765
mColWidth	.766
mCellData.	.766
mSelectedCell	.766
LTableArrayStorage	767
LTableArrayStorage()	.767
~LTableArrayStorage()	.768
FindCellData()	.768
GetCellData()	.768
GetStorageSize()	.768
InsertCols()	.768
InsertRows()	.769
RemoveCols()	.769
RemoveRows()	.769
SetCellData()	.769
mdataArray	.769
mOwnsArray	.769
LTableGeometry	771
LTableGeometry()	.771
~LTableGeometry()	.772
GetColHitBy()	.772
GetColWidth()	.772
GetImageCellBounds()	.773
GetRowHeight()	.773
GetRowHitBy()	.774
GetTableDimensions()	.774
InsertCols()	.774
InsertRows()	.775
RemoveCols()	.775
RemoveRows()	.776
SetColWidth()	.776

Table of Contents

SetRowHeight()	. 777
mTableView	. 777

LTableMonoGeometry **779**

LTableMonoGeometry()	. 779
~LTableMonoGeometry()	. 780
GetColHitBy()	. 780
GetColWidth()	. 780
GetImageCellBounds()	. 780
GetRowHeight()	. 780
GetRowHitBy()	. 781
GetTableDimensions()	. 781
SetColWidth()	. 781
SetRowHeight()	. 781
mColWidth	. 781
mRowHeight	. 781

LTableMultiGeometry **783**

LTableMultiGeometry()	. 783
~LTableMultiGeometry()	. 784
GetColHitBy()	. 784
GetColWidth()	. 784
GetImageCellBounds()	. 784
GetRowHeight()	. 785
GetRowHitBy()	. 785
GetTableDimensions()	. 785
InsertCols()	. 785
InsertRows()	. 785
RemoveCols()	. 785
RemoveRows()	. 786
SetColWidth()	. 786
SetRowHeight()	. 786
mRowHeights	. 786
mColWidths	. 786
mDefaultRowHeight	. 787
mDefaultColWidth	. 787

LTableMultiSelector	789
LTableMultiSelector()	789
~LTableMultiSelector()	790
CellsSelected()	790
ClickSelect()	790
DragSelect()	790
GetFirstSelectedCell().	790
GetFirstSelectedRow()	790
InsertCols()	791
InsertRows().	791
RemoveCols()	791
RemoveRows().	791
SelectAllCells().	791
SelectCell()	791
SelectCellBlock()	792
UnselectAllCells()	792
UnselectCell()	792
mSelectionRgn.	792
mAnchorCell	793
 LTableSelector	 795
LTableSelector()	795
~LTableSelector().	796
CellsSelected()	796
ClickSelect()	796
DragSelect()	797
GetFirstSelectedCell().	797
GetFirstSelectedRow()	797
InsertCols()	798
InsertRows().	798
RemoveCols()	799
RemoveRows().	799
SelectAllCells().	799
SelectCell()	800
UnselectAllCells()	800
UnselectCell()	800
mTableView	801

LTableSingleSelector	803
LTableSingleSelector().	803
~LTableSingleSelector().	804
CellsSelected().	804
ClickSelect().	804
DragSelect().	804
GetFirstSelectedCell().	804
GetFirstSelectedRow().	804
InsertCols().	805
InsertRows().	805
RemoveCols().	805
RemoveRows().	805
SelectAllCells().	805
SelectCell().	805
UnselectAllCells().	806
UnselectCell().	806
mSelectedCell	806
 LTableStorage	 807
LTableStorage().	807
~LTableStorage().	808
FindCellData().	808
GetCellData().	808
GetStorageSize().	809
InsertCols().	809
InsertRows().	810
RemoveCols().	810
RemoveRows().	811
SetCellData().	811
mTableView	812
 LTableView	 813
LTableView().	815
~LTableView().	815
ActivateSelf().	815
AdjustImageSize().	816
CellsSelected().	816

CellToIndex()	816
ClickCell()	817
ClickSelect()	817
ClickSelf()	818
DeactivateSelf()	818
DrawCell()	818
DrawSelf()	819
FetchIntersectingCells()	819
FindCellData()	819
GetCellData()	820
GetCellHitBy()	821
GetColWidth()	821
GetCustomHilite()	822
GetFirstSelectedCell()	822
GetHiliteRgn()	822
GetImageCellBounds()	823
GetLocalCellRect()	823
GetNextCell()	824
GetNextSelectedCell()	825
GetPreviousCell()	825
GetPreviousSelectedCell()	826
GetRowHeight()	826
GetTableGeometry()	827
GetTableSelector()	827
GetTableSize()	827
GetTableStorage()	828
HiliteCell()	828
HiliteCellActively()	828
HiliteCellInactively()	829
HiliteSelection()	829
IndexToCell()	830
InitTable()	830
InsertCols()	831
InsertRows()	831
IsValidCell()	832
IsValidCol()	832
IsValidRow()	833

Table of Contents

PointsAreClose()	833
RefreshCell()	834
RefreshCellRange()	834
RemoveAllCols()	835
RemoveAllRows()	835
RemoveCols()	835
RemoveRows()	836
ScrollCellIntoFrame()	836
SelectAllCells()	837
SelectCell()	837
SelectionChanged()	838
SetCellData()	838
SetColWidth()	838
SetCustomHilite()	839
SetDeferAdjustment()	839
SetRowHeight()	840
SetTableGeometry()	840
SetTableSelector()	841
SetTableStorage()	841
SetUseDragSelect()	842
UnselectAllCells()	842
UnselectCell()	842
mRows	843
mCols	843
mTableGeometry	843
mTableSelector	843
mTableStorage	844
mUseDragSelect	844
mCustomHilite	844
mDeferAdjustment	844

LTCPEndpoint

845

LTCPEndpoint()	846
~LTCPEndpoint()	846
AbortiveDisconnect()	846
AcceptIncoming()	847
AcceptRemoteDisconnect()	847

Connect()	847
Disconnect()	848
GetAmountUnread()	848
GetRemoteHostAddress()	848
Listen()	849
Receive()	849
ReceiveChar()	850
ReceiveData()	850
ReceiveDataUntilMatch()	851
ReceiveLine()	852
RejectIncoming()	852
Send()	853
SendCStr()	853
SendData()	853
SendDisconnect().	854
SendHandle()	854
SendPStr().	855
SendPtr()	855
LTEClearAction	857
LTEClearAction().	857
~LTEClearAction()	857
RedoSelf().	858
LTECutAction	859
LTECutAction()	859
~LTECutAction().	860
RedoSelf().	860
LTETextAction	863
LTETextAction()	863
LTETextAction	861
LTETextAction().	861
~LTETextAction()	862
RedoSelf().	862
UndoSelf()	862
mPastedTextH	862

Table of Contents

~LTETextAction()	864
CanRedo()	864
CanUndo()	865
Redo()	865
Undo()	865
UndoSelf()	865
mTextCommander	865
mTextPane	865
mMacTEH	866
mActionCommand	866
mDeletedTextH	866
mDeletedTextLen	866
mSelStart	867
mSelEnd	867

LTETypingAction **869**

LTETypingAction()	869
~LTETypingAction()	870
BackwardErase()	870
ForwardErase()	870
InputCharacter()	871
RedoSelf()	871
Reset()	871
UndoSelf()	872
mTypedTextH	872
mTypingStart	872
mTypingEnd	872

LTextButton **875**

LTextButton()	876
~LTextButton()	876
DrawSelf()	876
GetDescriptor()	876
HotSpotAction()	877
HotSpotResult()	877
SetDescriptor()	877
SetValue()	877

mText.	877
mTextTraitsID	878
mSelectedStyle.	878
LTextEditView	879
LTextEditView()	880
~LTextEditView().	881
AdjustCursorSelf()	881
AdjustImageToText()	882
AlignTextEditRects()	882
BeTarget()	882
CalcTEHeight()	883
ChangeFontSizeBy()	883
ClickSelf().	883
ClickAutoScroll().	884
DontBeTarget().	884
DrawSelf()	884
FindCommandStatus()	884
FocusDraw().	885
ForceAutoScroll()	885
GetAlignment()	885
GetAttributes().	885
GetColor().	885
GetDescriptor()	886
GetFont()	886
GetMacTEH()	886
GetSelection()	887
GetSize()	887
GetStyle()	887
GetTextHandle()	887
GetTextTraitsID().	888
GetValue().	888
HandleKeyPress()	888
HasAttribute()	888
HideSelf().	889
InitTextEditView()	889
Insert()	889

Table of Contents

MoveBy()	890
MyClickLoop()	890
MyTEClick()	890
ObeyCommand()	890
ResizeFrameBy()	891
RestorePlace()	891
SavePlace()	891
SaveStateForUndo()	892
ScrollImageBy()	892
SelectAll()	892
SetAlignment()	892
SetAttributes()	893
SetClickLoop()	893
SetColor()	893
SetDescriptor()	894
SetFont()	894
SetSize()	894
SetStyle()	894
SetTextHandle()	895
SetTextPtr()	895
SetTextTraitsID()	896
SpendTime()	896
TETooBig()	896
ToggleAttribute()	896
UserChangedText()	897
mTextEditH	897
mTextTraitsID	897
mTextAttributes	898

LToggleButton

899

LToggleButton()	900
DrawGraphic()	901
DrawSelf()	901
HotSpotAction()	901
HotSpotResult()	902
PointIsInFrame()	902
SetGraphics()	902

SetGraphicsType()	903
SetValue()	903
mGraphicsType	903
mOnID	904
mOnClickID	904
mOffID	904
mOffClickID	905
mTransitionID	905
LUDPEndpoint		907
LUDPEndpoint()	907
~LUDPEndpoint()	908
ReceiveFrom()	908
SendPacketData()	908
.	909
LUndoer		911
LUndoer()	911
~LUndoer()	912
ExecuteSelf()	912
FindUndoStatus()	913
PostAction()	913
ToggleAction()	914
operator=()	914
mAction	914
LVariableArray		917
LVariableArray()	918
~LVariableArray()	918
AdjustAllocation()	919
AdjustStorage()	919
AssignItemsAt()	919
GetItemOffset()	919
GetItemPtr()	920
GetItemSize()	920
GetOffsetsHandle()	921
GrabItemRangeSize()	921
GrabItemSize()	921

Table of Contents

InternalAdjustAllocation().	921
InternalCopyItem().	922
PokeItem().	922
ShiftItems().	922
Sort().	922
StoreNewItem().	922
mItemOffsetsH	923
mItemsAllocated.	923

LView

925

LView().	927
~LView().	928
Activate().	928
AdaptToNewSurroundings().	928
AdaptToSuperFrameSize().	928
AddSubPane().	928
AdjustCursor().	929
AutoScrollImage().	929
CalcPortOrigin().	929
CalcRevealedRect().	930
Click().	931
CountPanels().	931
Deactivate().	931
DeleteAllSubPanels().	931
Disable().	932
DontRefresh().	932
Draw().	932
Enable().	932
EstablishPort().	932
ExpandSubPane().	933
FindConstPaneByID().	933
FindDeepSubPaneContaining().	933
FindPaneByID().	934
FindShallowSubPaneContaining().	934
FindSubPaneHitBy().	934
FinishCreate().	934
FocusDraw().	934

FocusExposed()	935
GetDescriptorForPanelID()	935
GetImageLocation()	935
GetImageSize()	936
GetInFocusView()	936
GetLocalUpdateRgn()	937
GetPortOrigin()	937
GetRevealedRect()	937
GetScrollPosition()	938
GetScrollUnit()	938
GetSubPanels()	938
GetValueForPanelID()	939
Hide()	939
ImagePointIsInFrame()	939
ImageRectIntersectsFrame()	940
ImageToLocalPoint()	940
InitView()	941
LocalToImagePoint()	941
LocalToPortPoint()	942
MoveBy()	942
OrientAllSubPanels()	942
OrientSubPanel()	943
OutOfFocus()	943
PortToLocalPoint()	944
PrintPanel()	944
ReconcileFrameAndImage()	944
Refresh()	945
RemoveSubPanel()	945
ResizeFrameBy()	946
ResizeImageBy()	946
ResizeImageTo()	947
RestorePlace()	947
SavePlace()	947
ScrollBits()	948
ScrollImageBy()	948
ScrollImageTo()	949
ScrollPinnedImageBy()	950

Table of Contents

ScrollToPanel()	950
SetDescriptorForPaneID()	951
SetReconcileOverhang()	951
SetScrollUnit()	952
SetValueForPaneID()	952
Show()	953
SubImageChanged()	953
SuperActivate()	953
SuperDeactivate()	953
SuperDisable()	954
SuperEnable()	954
SuperHide()	954
SuperPrintPanel()	954
SuperShow()	954
sInFocusView	955
mImageSize	955
mImageLocation	955
mScrollUnit	955
mPortOrigin	956
mSubPanels	956
mRevealedRect	956
mUpdateRgnH	956
mReconcileOverhang	957

LWindow **959**

LWindow()	961
~LWindow()	961
Activate()	962
ActivateSelf()	962
ApplyForeAndBackColors()	962
AttemptClose()	962
CalcStandardBounds()	963
CalcStandardBoundsForScreen()	963
ClearAttribute()	964
ClickInContent()	964
ClickInDrag()	964
ClickInGoAway()	965

ClickInGrow()	965
ClickInZoom()	966
CreateWindow()	966
Deactivate()	967
DeactivateSelf()	967
DoClose()	967
DoSetBounds()	968
DoSetPosition()	968
DoSetZoom()	969
DrawSelf()	969
DrawSizeBox()	969
Enable()	970
EstablishPort()	970
ExpandSubPane()	970
FetchWindowObject()	971
FindCommandStatus()	971
FindWindowByID()	971
GetAEProperty()	972
GetAEWindowAttribute()	972
GetDescriptor()	973
GetGlobalBounds()	973
GetMacPort()	973
GetMinMaxSize()	974
GetStandardSize()	974
GlobalToPortPoint()	974
HandleAppleEvent()	975
HandleClick()	975
HasAttribute()	976
HideSelf()	976
InitWindow()	977
InvalPortRect()	977
InvalPortRgn()	977
MakeMacWindow()	977
MakeSelfSpecifier()	978
MoveWindowBy()	978
MoveWindowTo()	979
ObeyCommand()	979

Table of Contents

PortToGlobalPoint()	980
ResizeFrameBy()	980
ResizeWindowBy()	980
ResizeWindowTo()	981
Resume()	981
Select()	981
SendAESetBounds()	982
SendAESetPosition()	982
SendAESetZoom()	983
SetAEPProperty()	983
SetAttribute()	984
SetDescriptor()	984
SetForeAndBackColors()	985
SetMinMaxSize()	985
SetStandardSize()	985
Show()	986
ShowSelf()	986
Suspend()	986
UpdatePort()	986
ValidPortRect()	987
ValidPortRgn()	987
mMacWindowP	987
mMinMaxSize	987
mStandardSize	987
mUserBounds	988
mAttributes	988
mForeColor	988
mBackColor	988
mMoveOnlyUserZoom	989

LYieldAttachment 991

LYieldAttachment()	991
ExecuteSelf()	992
mQuantum	992
mNextTicks	992

StAsyncOperation	993
StAsyncOperation()	. 993
~StAsyncOperation()	. 994
AbortOperation()	. 994
GetThreadOperation()	. 994
Int_AsyncResume()	. 995
WaitForResult()	. 995
mThread	. 995
mResult	. 996
sPendingOperations	. 996
 StCritical	 997
StCritical()	. 997
~StCritical()	. 997
 StCursor	 999
StCursor()	. 999
~StCursor()	1000
mRestoreID	1000
 StDialogHandler	 1001
StDialogHandler()	1002
~StDialogHandler()	1002
AllowSubRemoval()	1002
DoDialog()	1002
FindCommandStatus()	1003
GetDialog()	1003
ListenToMessage()	1003
SetSleepTime()	1004
mDialog	1004
mMessage	1004
mSleepTime	1004
 StMacTCPOperation	 1005
StMacTCPOperation()	1005
~StMacTCPOperation()	1006
AsyncRun()	1006
GetCompletionProc()	1007

Table of Contents

GetParamBlock()	1007
Int_TCPCompletionProc()	1007
Run()	1008
mTCPParamBlock	1008
sMacTCPCompletionProc	1008
sTCPParamBlockDeleteQueue	1009
StMacTCPUDPOperation	1011
StMacTCPUDPOperation()	1011
~StMacTCPUDPOperation()	1012
GetCompletionProc()	1012
GetParamBlock()	1012
Int_UDPCompletionProc()	1013
Run()	1013
mUDPPParamBlock	1013
sMacTCPUDPCompletionProc	1014
sUDPPParamBlockDeleteQueue	1014
StMutex	1015
StMutex()	1015
~StMutex()	1015
mMutex.	1016
StOpenTptOperation	1017
StOpenTptOperation()	1017
~StOpenTptOperation()	1018
GetCookie()	1018
GetEventCode()	1019
GetResultCode()	1019
Int_TimerProc()	1019
OperationTimedOut().	1020
SetEventCode()	1020
WaitForCompletion()	1020
WaitForResult()	1021
mNotifHandler	1021
mEventCode.	1021
mCookieTest.	1022
mTestCookie.	1022

	mResultCode	1022
	mCookie	1022
	mOperationTimeout	1023
	sOTOpTimerUPP.	1023
StSetupGlobals		1025
	StSetupGlobals()	1025
	~StSetupGlobals()	1026
	mOldGlobals	1026
TArray		1027
	TArray()	1027
	~TArray()	1028
	AddItem().	1028
	AssignItemsAt()	1029
	FetchIndexOf().	1029
	FetchInsertIndexOf()	1029
	FetchItemAt()	1029
	FetchItemPtr()	1029
	InsertItemsAt().	1030
	Remove()	1030
	Operator[]	1030
UCursor		1031
	GetCurrentID()	1031
	InitTheCursor()	1032
	InAnimatedCursor()	1032
	SetArrow()	1032
	SetCurrentID().	1033
	SetCross()	1033
	SetInAnimatedCursor()	1033
	SetIBeam()	1034
	SetPlus()	1034
	SetTheCursor().	1034
	SetWatch().	1035
	sCurrentID	1035
	sInAnimatedCursor	1035

UDNSCache	1037
AddToDNSCache()	1037
CheckCache()	1038
CreateDNSCacheElem()	1038
GetAddressFromCache()	1038
GetNameFromCache()	1039
sOTDNSNameCache	1039
sOTDNSAddressCache	1040
 UMacTCPSupport	 1041
GetMacTCPRefNum()	1041
HasMacTCP()	1041
OpenMacTCPDriver()	1042
sMacTCPRefNum	1042
sCloseResolverTask.	1042
 UNetworkFactory	 1043
CreateInternetMapper()	1043
CreateTCPEndpoint()	1044
CreateUDPEndpoint()	1044
HasMacTCP()	1044
HasOpenTransport()	1045
HasTCP()	1045
 UOpenTptSupport	 1047
GetOTGestalt().	1047
HasOpenTransport()	1048
HasOpenTransportTCP()	1048
OTAddressToPPAddress().	1048
StartOpenTransport()	1049
sOTGestaltTested.	1049
sCloseOpenTptTask	1049
sOTGestaltResult.	1049
 UReanimator	 1051
CreateView()	1051
LinkListenerToBroadcasters()	1052
LinkListenerToControls()	1053

	ObjectsFromStream()	1053
	ReadObjects()	1054
UTextDrawing		1055
	DrawWithJustification()	1055

Table of Contents

LAction

LAction is a PowerPlant class that helps manage the do, undo, and redo operations. It maintains the data required to restore the previous state of menu items.

Methods The methods in this class are:

LAction()	~LAction()
CanRedo()	CanUndo()
Finalize()	GetDescription()
IsDone()	IsPostable()
Redo()	RedoSelf()
Undo()	UndoSelf()
operator=()	

Data Members The data members in this class are:

mStringResID	mStringIndex
mIsDone	

Operation When you implement Undo capability in your program, you will want to modify the text of the Undo menu item of the Edit Menu in your application. For example, if the last operation of your program was to clear some text from a document window, you want to change the Undo menu item to Undo Clear. If you then chose Undo Clear from the Edit Menu, you would then want to change the menu item to Redo Clear.

See Also [LUndoer](#)

Source files (Action Classes)
[LAction.h](#)
[LAction.cp](#)

LAction

LAction()

Purpose	Create or copy LAction objects.		
Access	Public		
Prototypes	<pre>LAction(ResIDT inStringResID, SInt16 inStringIndex, Boolean inAlreadyDone); LAction(ResIDT inStringResID, SInt16 inStringIndex, Boolean inAlreadyDone); LAction(const LAction &inOriginal);</pre>		
Parameters	The parameters for these constructors are:		
	ResIDT	inStringResID	The resource ID of the Redo menu item text (a 'STR#' resource). The Undo text menu item must have a STR# resource ID that is one higher.
	SInt16	inStringIndex	The index number in the STR# resource for the Redo menu item.
	Boolean	inAlreadyDone	A value indicating whether the action is already done.
	const LAction&	inOriginal	A reference to an LAction object.

~LAction()

Purpose	Destroy LAction objects.
Access	Public
Prototype	<code>~LAction();</code>

Parameters	None
------------	------

CanRedo()

Purpose	This method tells whether an action is currently undone.
Access	Virtual, Public
Prototype	<code>virtual Boolean CanRedo() const;</code>
Parameters	None
Return	Returns a Boolean true if the action is undone, false otherwise.

CanUndo()

Purpose	This method tells whether an action is currently done.
Access	Virtual, Public
Prototype	<code>virtual Boolean CanUndo() const;</code>
Parameters	None
Return	Returns a Boolean true if the action is done, false otherwise.

Finalize()

Purpose	This method is called by the framework before deleting an action posted to an Undoer. This allows you to do something with the action before it is lost. For example, you may want to preserve the action if you are implementing multiple Undo.
Access	Virtual, Public
Prototype	<code>virtual void Finalize();</code>
Parameters	None

LAction

Return	None
--------	------

GetDescription()

Purpose	This method allows you to retrieve the strings for Undo and Redo menu items.
Access	Virtual, Public
Prototype	<code>virtual void GetDescription(Str255 outRedoString, Str255 outUndoString) const;</code>
Parameters	None
Return	None

IsDone()

Purpose	This method returns the value of mIsDone .
Access	Public
Prototype	<code>Boolean IsDone() const;</code>
Parameters	None
Return	A Boolean that reflects the value of mIsDone .

IsPostable()

Purpose	Return whether an Action is postable, meaning that it affects the Undo state. This method always returns true. Override this method to return false for actions that are not undoable.
Access	Virtual, Public
Prototype	<code>virtual Boolean IsPostable() const;</code>

Parameters	None
Return	A Boolean indicating whether the Action affects the Undo state.

Redo()

Purpose	This method is a wrapper which calls RedoSelf() if the Action can be redone. Note that the first time this function is called, "Redo" really means "Do" the Action (unless mIsDone is set to true when creating the Action).
Access	Virtual, Public
Prototype	<code>virtual void Redo();</code>
Parameters	None
Return	None

RedoSelf()

Purpose	This method is a pure virtual method that you must supply an implementation for in your class that inherits from LAction.
Access	Virtual, Public
Prototype	<code>virtual void RedoSelf() = 0;</code>
Parameters	None
Return	None

Undo()

Purpose	This method is a wrapper to call UndoSelf() .
Access	Virtual, Public

LAction

Prototype	<code>virtual void Undo();</code>
Parameters	None
Return	None

UndoSelf()

Purpose	This method is a pure virtual method that you must supply an implementation for in your class that inherits from LAction.
Access	Virtual, Public
Prototype	<code>virtual void UndoSelf() = 0;</code>
Parameters	None
Return	None

operator=()

Purpose	The assignment operator copies the values of the mStringResID , mStringIndex and mIsDone data members to the assigned object.
Access	Public
Prototype	<code>LAction& operator=(const LAction &inOriginal);</code>
Parameters	The parameter for this operator is a <code>const</code> reference to an LAction object that you wish to make a copy of.
Return	Returns an LAction reference.

mStringResID

Purpose	This member contains the resource ID of the 'STR#' resource that has the text for the redo menu items.
---------	--

Access	Protected
Prototype	ResIDT mStringResID;

mStringIndex

Purpose	This member contains an index number into the 'STR#' resource for the redo item related to an action.
Access	Protected
Prototype	SInt16 mStringIndex;

mIsDone

Purpose	This member is true if an action is done, and false if an action is undone.
Access	Protected
Prototype	Boolean mIsDone;

LActiveScroller

Overview LActiveScroller is a PowerPlant class that is used for scrolling views. Scroller views implement all the functionality of scroll bars, including creation, resizing, moving, hiding, showing, enabling and other operations. Sometimes you will see scrolling views referred to simply as “scrollers” in the PowerPlant documentation.

The only functional difference between [LActiveScroller](#) and [LScroller](#) is that LActiveScroller supports dynamic scrolling of the view.

Methods The methods in this class are:

LActiveScroller()	~LActiveScroller()
ActiveThumbScroll()	AdaptToNewThumbValue()
AdjustScrollBars()	AssignThumbProcs()
CalcValueFromPoint()	EndThumbTracking()
HandleThumbScroll()	IsTrackingThumb()
ListenToMessage()	StartThumbTracking()

Data Members The data members in this class are:

mThumbControl	mTrackBarUnits
mTrackBarPin	mTrackBarSize
mOriginalValue	mTrackRect
mValueSlop	mVertThumbAction
mHorizThumbAction	

Operation The most important feature of a scroller view is that it has only one subpane, which is a view. The subview may contain an arbitrary number of panes and views. The net effect is that the scroller view may contain any number of panes of any type.

Source files (Pane Classes)

LActiveScroller.h

LActiveScroller.cp

- Ancestors
- [LListener](#)
 - [LPane](#)
 - [LScroller](#)
 - [LView](#)
-

LActiveScroller()

Purpose The constructor creates objects from the passed-in parameters.

Access Public

Prototype

```
LActiveScroller();
LActiveScroller( LStream *inStream );
LActiveScroller( const LScroller &inOriginal );
LActiveScroller( const SPaneInfo &inPaneInfo,
const SViewInfo &inViewInfo,
SInt16 inHBLeftIndent,
SInt16 inHBRightIndent,
SInt16 inVBTopIndent,
SInt16 inVBBottomIndent,
LView *inScrollingView);
```

Parameters These constructors have the following parameters:

const LScroller&	inOriginal	A reference to the LScroller object you want to copy.
const SPaneInfo&	inPaneInfo	A reference to the SPaneInfo object that is the super view.
const SViewInfo&	inViewInfo	A reference to the SViewInfo object that contains information about the SuperView.
SInt16	inHBLeftIndent	The indentation to use on the left side for the horizontal scrolling. A good initial value for this parameter is 15.

SInt16	inHBRightIndent	The indentation to use on the right side for the horizontal scrolling. A good initial value for this parameter is 15.
SInt16	inVBTOPIndent	The indentation to use on the top for the vertical scrolling. A good initial value for this parameter is 15.
SInt16	inVBBottomIndent	The indentation to use on the bottom for the vertical scrolling. A good initial value for this parameter is 15.
LView	inScrollingView	A pointer to the view that corresponds to this Scroller.
LStream*	inStream	A pointer to a stream object that contains the information to create the LScroller object.

~LActiveScroller()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LActiveScroller();</code>

ActiveThumbScroll()

Purpose	This is the static callback for the scroll bars when the thumb is tracked. The LScroller implementation embeds a pointer to the scroller object in the control reference field of the control record.
Access	Static, Protected
Prototype	<code>static pascal void LActiveThumbScroll();</code>
Parameters	None
Return	None

AdaptToNewThumbValue()

Purpose	This method is where to do the scroll of the scrolling view based on the new value for the tracking control.	
Access	Virtual, Protected	
Prototype	<pre>virtual void AdaptToNewThumbValue(SInt32 inNewValue);</pre>	
Parameters	This method has the following parameter:	
	SInt32	inNewValue This is the new value to set for the scroller.
Return	None	

AdjustScrollBars()

Purpose	<p>This method gets called as a result of the view scrolling. We want this to function the same as the LScroller implementation, except while we're tracking. We maintain the scroll bar during the track, so there's no need to calculate the value and draw the control twice per scroll.</p> <p>This method is an override of AdjustScrollBars() in the LScroller class.</p>
---------	---

AssignThumbProcs()

Purpose	<p>Create new UPP's (Universal Procedure Pointers) and assign them to the thumbs of each of the scroll bars (if they exist).</p> <p>To learn more about UPP's, refer to <i>Inside Macintosh: PowerPC System Software</i>, published by Addison-Wesley.</p>
---------	--

Access	Virtual, Protected
Prototype	<code>virtual void AssignThumbProcs();</code>
Parameters	None
Return	None

CalcValueFromPoint()

Purpose	Given a point, calculate the value for the control. The instance variables used in this calculation are initialized in the StartThumbTracking() method. The returned value is adjusted for slop.		
Access	Virtual, Protected		
Prototype	<code>virtual SInt32 CalcValueFromPoint(Point inPoint);</code>		
Parameters	This method has the following parameter:		
	Point	inPoint	This is the point coordinates from which the control value is calculated.
Return	SInt32 indicating the new value for the control.		

EndThumbTracking()

Purpose	Register the fact that we're done tracking. The most important thing is to NULL out the instance variable that keeps track of the current tracking control. When that variable is nil we're assumed to not be tracking.		
Access	Virtual, Protected		
Prototype	<code>virtual void EndThumbTracking();</code>		
Parameters	None		
Return	None		

HandleThumbScroll()

Purpose	This method is where the actual thumb action is handled.	
Access	Virtual, Protected	
Prototype	<pre>virtual void HandleThumbScroll(LStdControl *inWhichControl);</pre>	
Parameters	This method has the following parameter:	
	LStdControl*	inWhichControl This is the pointer to the standard control.
Return	None	
Remarks	We could eliminate the click loop if we could guarantee to get called if the drag didn't change the value of the control. Currently, if the value doesn't change, no notification is issued, therefore there's no way for us to reset the tracking state.	

IsTrackingThumb()

Purpose	Correct way to check to see if we're currently in the tracking loop.	
Access	Virtual, Public	
Prototype	<pre>virtual Boolean IsTrackingThumb();</pre>	
Parameters	None	
Return	Boolean indicating whether the tracking loop is currently executing.	

ListenToMessage()

Purpose	In this method we want to do exactly like we do in the LScroller implementaion except when we have finished tracking. Usually the
---------	---

msg_ThumbDragged message causes the scrolling view to adjust, but since we did live tracking, we're already scrolled to the correct location. So we just ignore that particular message.

This method is an override of [ListenToMessage\(\)](#) in the [LListener](#) class.

StartThumbTracking()

Purpose	This method sets up the tracking instance variables. We also calculate the value slop constant so that clicks in the thumb of a scroll bar with a large scroll unit will not cause unnecessary jumps in the view.		
Access	Virtual, Protected		
Prototype	virtual void StartThumbTracking(LStdControl *inWhichControl);		
Parameters	This method has the following parameter:		
	LStdControl*	inWhichControl	This is the pointer to the standard control.
Return	None		

mThumbControl

Purpose	This data member is the pointer to the thumb standard control.
Access	Protected
Prototype	LStdControl *mThumbControl;

mTrackBarUnits

Purpose	This data member is the value of the track bar units.
Access	Protected
Prototype	<code>SInt32 mTrackBarUnits;</code>

mTrackBarPin

Purpose	This data member is the value of the track bar pin.
Access	Protected
Prototype	<code>SInt32 mTrackBarPin;</code>

mTrackBarSize

Purpose	This data member is the value of the track bar size.
Access	Protected
Prototype	<code>SInt32 mTrackBarSize;</code>

mOriginalValue

Purpose	This data member is the original value of the control before any changes.
Access	Protected
Prototype	<code>SInt32 mOriginalValue;</code>

mTrackRect

Purpose	This data member stores the Rect value for tracking.
Access	Protected
Prototype	<code>Rect mTrackRect;</code>

mValueSlop

Purpose	This data member stores the slop factor to apply to the value.
Access	Protected
Prototype	<code>SInt32 mValueSlop;</code>

mVertThumbAction

Purpose	This data member stores the vertical thumb action UPP.
Access	Protected
Prototype	<code>ThumbActionUPP mVertThumbAction;</code>

mHorizThumbAction

Purpose	This data member stores the horizontal thumb action UPP.
Access	Protected
Prototype	<code>ThumbActionUPP mHorizThumbAction;</code>

LApplication

Overview	LApplication is a PowerPlant class that is used for encapsulating the common behaviors of a typical Mac OS application.	
Methods	The methods in this class are:	
	LApplication()	~LApplication()
	CountSubModels()	DoQuit()
	FindCommandStatus()	GetPositionOfSubModel()
	GetState()	GetSubModelByName()
	GetSubModelByPosition()	HandleAppleEvent()
	Initialize()	MakeMenuBar()
	MakeModelDirector()	MakeSelfSpecifier()
	ObeyCommand()	ProcessNextEvent()
	Run()	SendAEQuit()
	SetSleepTime()	ShowAboutBox()
	StartUp()	
Data Members	The data members in this class are:	
	mState	mSleepTime
Operation	You create a single object of this class in your application. This object manages the execution of the application program. It handles initialization, destruction, the main event loop, application-level events using LEventDispatcher , Apple Events, updating of the menu bar, cursor adjustment, and printing of documents.	
Source files	(Commander Classes)	
	LApplication.h	
	LApplication.cp	
See also	LAttachable	
	LCommander	

LApplication

[LDocApplication](#)

[LEventDispatcher](#)

[LMenuBar](#)

LApplication()

Purpose	The constructor initializes parameters such as the default event loop sleep time, Color QuickDraw, and some PowerPlant-specific parameters.
Access	Public
Prototype	<code>LApplication();</code>
Parameters	None

~LApplication()

Purpose	The destructor destroys the application object.
Access	Virtual, Public
Prototype	<code>virtual ~LApplication();</code>
Parameters	None

CountSubModels()

Purpose	This method counts the number of windows or documents open in the application.
Access	Virtual, Public
Prototype	<code>virtual SInt32 CountSubModels(DescType inModelID) const;</code>

Parameters	The parameter for this method is:		
	DescType	inModelID	This is a resource descriptor parameter, which you can set to cWindow to count the number of windows open.
Return	SInt32 indicating the number of open windows (if inModelID == cWindow) or open documents.		

DoQuit()

Purpose	Quit the Application, if the conditions specified by AttemptQuit() in LCommander() are met.		
Access	Virtual, Public		
Prototype	<code>virtual void DoQuit(SInt32 inSaveOption);</code>		
Parameters	The parameter for this method is:		
	SInt32	inSaveOption	This is a parameter passed on to AttemptQuit() in LCommander() .
Return	None		

FindCommandStatus()

Purpose	Pass back status information on whether a Command is enabled and/or marked in a Menu.		
Access	Virtual, Public		
Prototype	<code>virtual void FindCommandStatus(CommandT inCommand, Boolean &outEnabled, Boolean &outUsesMark,</code>		

LApplication

```
USInt16 &outMark,  
Str255 outName);
```

Parameters The parameters for this method are:

Command T	inCommand	This is the command type being passed in.
Boolean	outEnabled	This passed-out value indicates whether the command is enabled.
Boolean	outUsesMark	This passed-out value indicates whether the command has a check mark.
USInt16&	outMark	The character to use as a mark.
Str255	outName	A passed-out string containing the menu command text.

Return None

GetPositionOfSubModel()

Purpose Return the position (1 means the first) of an Apple Event SubModel within an Application.

Access Virtual, Public

Prototype

```
virtual SInt32 GetPositionOfSubModel(  
DescType inModelID,  
const LModelObject *inSubModel) const;
```

Parameters The parameters for this method are:

	DescType	inModelID	A resource descriptor that describes the SubModel, or you can set it to cWindow to find the window index using GetMacPort().
	const LModelObject*	inSubModel	This passed-in pointer is the address of the LModeObject to search for.
Return	SInt32 containing the position of the Apple Event SubModel.		

GetState()

Purpose	This method returns the value of the mState data member.
Access	Inline, Public
Prototype	EProgramState GetState() const;
Parameters	None
Return	EProgramState containing the value of mState .

GetSubModelByName()

Purpose	Pass back a token to an Apple Event SubModel specified by name.
Access	Virtual, Public
Prototype	virtual void GetSubModelByName(DescType inModelID, Str255 inName, AEDesc &outToken) const;
Parameters	The parameters for this method are:

DescType	inModelID	A resource descriptor that describes the SubModel, or you can set it to cWindow to search for a window by name.
Str255	inName	A string pointer containing the name of the SubModel to search for.
AEDesc&	outToken	This passed-out reference is the AEDesc for the specified SubModel.

Return None

GetSubModelByPosition()

Purpose Get the token of an Apple Event SubModel specified by its position.

Access Virtual, Public

Prototype `virtual void GetSubModelByPosition(
DescType inModelID,
SInt32 inPosition,
AEDesc &outToken) const;`

Parameters The parameters for this method are:

DescType	inModelID	This is a resource descriptor parameter that describes the SubModel, or you can set it to cWindow to search for a window by its position.
SInt32	inPosition	The position of the SubModel to retrieve a token for, 1 means first position.
AEDesc&	outToken	This passed-out reference is the AEDesc for the specified SubModel.

Return None

HandleAppleEvent()

Purpose	This method handles the Apple Events that are sent to the application.
Access	Virtual, Public
Prototype	<pre>virtual void HandleAppleEvent(const AppleEvent &inAppleEvent, AppleEvent &outAEReply, AEDesc &outResult, long inAENumber);</pre>
Parameters	The parameters for this method are:

const AppleEvent&	inAppleEvent	This is a reference to the AppleEvent passed-in for the application to handle.
AppleEvent&	outAEReply	This is a reference to the reply received after the AppleEvent.
AEDesc&	outResult	This passed-out reference is the AEDesc for the specified SubModel.
long	inAENumber	This is the passed-in Apple Event number, something like ae_Clone or ae_Move.

Initialize()

Purpose	Calling this method provides a last chance to initialize the application before event processing begins.
Access	Virtual, Protected
Prototype	<pre>virtual void Initialize();</pre>
Parameters	None

LApplication

Return	None
--------	------

MakeMenuBar()

Purpose	Create the MenuBar object for the application. You should override this to use a class other than LMenuBar .
Access	Protected
Prototype	<code>virtual void MakeMenuBar();</code>
Parameters	None
Return	None

MakeModelDirector()

Purpose	Create ModelDirector (an AppleEvent handle) object for the application. You should override this if you want to use a class other than LModelDirector.
Access	Virtual, Protected
Prototype	<code>virtual void MakeModelDirector();</code>
Parameters	None
Return	None

MakeSelfSpecifier()

Purpose	Make an Object Specifier for the application.
Access	Virtual, Protected
Prototype	<code>virtual void MakeSelfSpecifier(AEDesc& inSuperSpecifier, AEDesc& outSelfSpecifier) const;</code>

Parameters	The parameters for this method are:		
	AEDesc&	inSuperSpecifier	This passed-in reference is an AEDesc, and is not needed in this method implementation.
	AEDesc&	outSelfSpecifier	This is the passed-out AEDesc that contains the Object Specifier created.
Return	None		

ObeyCommand()

Purpose	Process commands that are generated when running the application.		
Access	Virtual, Public		
Prototype	virtual Boolean ObeyCommand(CommandT inCommand, void *ioParam);		
Parameters	The parameters for this method are:		
	Command T	inCommand	This is the command type being passed in.
	void*	ioParam	This is a pointer to a data block that accompanies the command.
Return	A Boolean containing true if the command was handled by this call, and false otherwise.		

ProcessNextEvent()

Purpose	Retrieve and handle the next event in the application event queue. This includes letting Attachments process the event, updating menu item status, and calling Repeaters.
Access	Virtual, Public
Prototype	<code>virtual void ProcessNextEvent();</code>
Parameters	None
Return	None

Run()

Purpose	Run the Application by processing events until quitting. This includes doing some menu management, cursor updating, and other housekeeping.
Access	Virtual, Public
Prototype	<code>virtual void Run();</code>
Parameters	None
Return	None
Remarks	You should catch all exceptions in your code. If an exception occurs when calling ProcessNextEvent() , a signal will occur and the application will continue running.

SendAEQuit()

Purpose	Sends a Quit AppleEvent to this application.
Access	Virtual Public
Prototype	<code>virtual void SendAEQuit();</code>

Parameters	None
Return	None

SetSleepTime()

Purpose	This method sets the value of the mSleepTime data member.	
Access	Inline, Public	
Prototype	<code>void SetSleepTime(SInt32 inSleepTime);</code>	
Parameters	The parameter for this method is:	
	SInt32 inSleepTime	This is the value to set for mSleepTime .
Return	None	

ShowAboutBox()

Purpose	Display the About Box for the application. This basic implementation just puts up an Alert Box. You should override this if you wish to display a more elaborate About Box.	
Access	Virtual, Public	
Prototype	<code>virtual void ShowAboutBox();</code>	
Parameters	None	
Return	None	

StartUp()

Purpose	This method is meant to provide a place to perform actions at application start up when launched without any documents. You
---------	---

LApplication

should override this if you wish to instead perform some default action, such as creating a new, untitled document.

Access	Virtual, Protected
Prototype	<code>virtual void StartUp();</code>
Parameters	None
Return	None

mState

Purpose	This data member contains a value signifying the state of the program, such as <code>programState_StartingUp</code> .
Access	Protected
Prototype	<code>EProgramState mState;</code>

mSleepTime

Purpose	This data member contains the overridable value of the sleep time parameter for the main event loop. This member is set to a default value by the constructor.
Access	Protected
Prototype	<code>long mSleepTime;</code>

LArray

Description LArray is a PowerPlant class that is used for implementing an ordered collection of fixed-size items. The first item is at index value 1. The index value 0 is used to indicate a nonexistent item.

Methods The methods in this class are:

<u>LArray()</u>	<u>~LArray()</u>
<u>AddItem()</u>	<u>AdjustAllocation()</u>
<u>AdjustStorage()</u>	<u>AssignItemsAt()</u>
<u>AttachIterator()</u>	<u>BinarySearch()</u>
<u>BinarySearchByKey()</u>	<u>CopyArray()</u>
<u>DestroyArray()</u>	<u>DetachIterator()</u>
<u>FetchIndexOf()</u>	<u>FetchIndexOfKey()</u>
<u>FetchInsertIndexOf()</u>	<u>FetchInsertIndexOfKey()</u>
<u>FetchItemAt()</u>	<u>GetComparator()</u>
<u>GetCount()</u>	<u>GetItemPtr()</u>
<u>GetItemSize()</u>	<u>GetItemsHandle()</u>
<u>GrabItemRangeSize()</u>	<u>GrabItemSize()</u>
<u>InitArray()</u>	<u>InsertItemsAt()</u>
<u>InternalAdjustAllocation()</u>	<u>InternalCopyItem()</u>
<u>InternalMoveItem()</u>	<u>InternalSwapItems()</u>
<u>InvalidateSort()</u>	<u>IsKeptSorted()</u>
<u>IsSorted()</u>	<u>ItemsInserted()</u>
<u>ItemsRemoved()</u>	<u>LinearSearch()</u>
<u>LinearSearchByKey()</u>	<u>Lock()</u>
<u>MoveItem()</u>	<u>PeekItem()</u>
<u>PokeItem()</u>	<u>Remove()</u>
<u>RemoveItemsAt()</u>	<u>SetComparator()</u>

LArray

[SetKeepSorted\(\)](#)

[ShiftItems\(\)](#)

[Sort\(\)](#)

[StoreNewItems\(\)](#)

[SwapItems\(\)](#)

[Unlock\(\)](#)

[ValidIndex\(\)](#)

[operator=\(\)](#)

Data Members The data members in this class are:

[mComparator](#)

[mDataAllocated](#)

[mDataStored](#)

[mIsSorted](#)

[mItemCount](#)

[mItemsH](#)

[mItemSize](#)

[mIteratorListHead](#)

[mKeepSorted](#)

[mLockCount](#)

[mOwnsComparator](#)

Operation Index values are signed, 32-bit integers. When specifying an item, you pass a pointer to the item data as a parameter. The array stores a copy of the data, or returns a copy of the data to you.

You can insert, remove, get a value, assign a value, swap, move items, and get data about the array. PowerPlant defines the constants `index_First` and `index_Last` so that you can easily insert items at the beginning or end of the array.

Source files (Array Classes)

`LArray.h`

`LArray.cp`

See Also [LArrayIterator](#)

[LComparator](#)

LArray()

Purpose The constructors create the object with the passed-in parameters.

Access Public

Prototypes `LArray();`
`LArray(const LArray &inOriginal);`

These constructors create an array with space pre-allocated for the specified number of items of the specified size. If `nItemCount` is not specified, then the array will be empty after creation.

```
LArray( UInt32 inItemCount,
        UInt32 inItemSize,
        LComparator *inComparator = nil,
        Boolean inKeepSorted = false );
LArray( UInt32 inItemSize,
        LComparator *inComparator = nil,
        Boolean inKeepSorted = false );
LArray( UInt32 inItemSize,
        Handle inItemsHandle,
        LComparator *inComparator = nil,
        Boolean inIsSorted = false,
        Boolean inKeepSorted = false );
```

Parameters The parameters for this constructor are:

UInt32	<code>inItemCount</code>	The number of items to put in the array.
UInt32	<code>inItemSize</code>	The size of one item.
Handle	<code>inItemsHandle</code>	A handle to items for the array.
LComparator*	<code>inComparator</code>	The Comparator object compares array items and must be allocated in the heap via "new". The array assumes ownership of the Comparator and is responsible for deleting it. The default is <code>nil</code> .

LArray

Boolean	inKeepSorted	This specifies whether to keep the array sorted when items are inserted or assigned new values. The default is <code>false</code> .
<code>const LArray &</code>	inOriginal	A reference to the object to copy.

~LArray()

Purpose	The destructor destroys the array.
Access	Virtual, Public
Prototype	<code>virtual ~LArray();</code>

AddItem()

Purpose	Add one item to array. If the array is not sorted, add the item to the end of the array.		
Access	Virtual, Public		
Prototype	<code>virtual ArrayIndexT AddItem(const void *inItem, USInt32 inItemSize);</code>		
Parameters	This method takes the following parameters:		
	<code>const void*</code>	<code>inItem</code>	The item to add to the array.
	<code>USInt32</code>	<code>inItemSize</code>	The size of the item in bytes.
Return	Returns the index point at which the item was inserted.		
Remarks	For unsorted Arrays, this function is a faster version of InsertItemsAt() since it doesn't have to bother with checking/adjusting the count and insertion index.		

AdjustAllocation()

Purpose	This method is a wrapper call to InternalAdjustAllocation() .	
Access	Virtual, Public	
Prototype	<pre>virtual void AdjustAllocation(UInt32 inExtraItems, UInt32 inExtraData);</pre>	
Parameters	This method takes the following parameters:	
	UInt32 inExtraItems	The item to add to the array.
	UInt32 inExtraData	The size of the item in bytes. This has a default parameter value of zero and is ignored.
Return	None	
Remarks	For fixed-size items, the number of items determines the amount of data stored.	

AdjustStorage()

Purpose	This method is called internally when the number of bytes used by items in the array changes.	
Access	Virtual, Protected	
Prototype	<pre>virtual void AdjustStorage(SInt32 inDeltaItems, SInt32 inDeltaData);</pre>	
Parameters	This method takes the following parameters:	
	SInt32 inDeltaItems	The change in the number of items in the array.
	SInt32 inDeltaData	Ignored, since the number of items determines the amount of data stored for fixed-size item arrays.

LArray

Return	None
Remarks	<p>If the current allocation is too small, this implementation sets the internal allocation size to:</p> $\text{current_alloc} + \max(\text{current_alloc}, \text{delta_bytes})$ <p>For small adjustments (adding less bytes than what's already allocated), this doubles the allocation.</p> <p>For large adjustments (adding more bytes than what's already allocated), this increases the allocation by the number of bytes added.</p>

AssignItemsAt()

Purpose	Assign the same value to items in the array starting at the specified index.		
Access	Virtual, Public		
Prototype	<pre>virtual ArrayIndexT AssignItemsAt(UInt32 inCount, ArrayIndexT inAtIndex, const void *inValue, UInt32 inItemSize)</pre>		
Parameters	This method takes the following parameters:		
	UInt32	inCount	The number of items to make the same.
	ArrayIndexT	inAtIndex	The starting index.
	const void*	inValue	A pointer to the item data. The array makes and stores a copy of the item data.
	UInt32	inItemSize	The size of the array item.
Return	Returns index of first "assigned" item. This may be different from <code>inAtIndex</code> if the array is sorted. Returns <code>LArray::index_Bad</code> if <code>inAtIndex</code> is out of range.		
Remarks	Does nothing if <code>inAtIndex</code> is out of range.		

AttachIterator()

Purpose	Associate an iterator with an array.		
Access	Protected		
Prototype	<pre>void AttachIterator(LArrayIterator *inIterator) const;</pre>		
Parameters	This method takes the following parameter:		
	LArrayIterator*	inIterator	The iterator to associate with the array.
Return	None		

BinarySearch()

Purpose	This method returns the index of the specified item using a binary search. It assumes that the array is sorted.		
Access	Protected		
Prototype	<pre>ArrayIndexT BinarySearch(const void *inItem, UInt32 inItemSize) const;</pre>		
Parameters	This method takes the following parameters:		
	const void *	inItem	A pointer to the item to search for.
	UInt32	inItemSize	The size of the array item.
Return	The index at which the item was found. If not found, the value returned will be <code>index_Bad</code> .		

BinarySearchByKey()

Purpose	Return the index of the item with the specified key, using a binary search. It assumes that the array is sorted.	
Access	Protected	
Prototype	<pre>ArrayIndexT BinarySearchByKey(const void *inKey) const;</pre>	
Parameters	This method takes the following parameter:	
	<pre>const void * inKey</pre>	A pointer to the key to search for.
Return	The index at which the item was found. If not found, the value returned will be <code>index_Bad</code> .	

CopyArray()

Purpose	Creates a deep copy by duplicating the items in the Array. However, if the items in the Array are pointers to other objects, those other objects aren't duplicated.	
Access	Private	
Prototype	<pre>void CopyArray(const LArray &inOriginal);</pre>	
Parameters	This method takes the following parameter:	
	<pre>const LArray& inOriginal</pre>	A reference to the original object to copy.
Return	None	
Remarks	<ul style="list-style-type: none">• Also duplicates the Comparator if the original owns its Comparator. NOTE: If you have an array which owns a custom Comparator, you MUST override Clone() for your subclass of LComparator in order for this array copy constructor to work properly.• Copy is always unlocked, even if original is locked. This ensures that Lock() and Unlock() calls always balance. If the copy were	

locked (because the original is locked), then to change the copy you would have to unlock it without ever having called Lock(). And tracking the number of times the copy is locked would be a big problem.

- Copy does NOT duplicate original's iterators, since there would be no way to access such iterators. The list of iterators is an internal implementation detail.

DestroyArray()

Purpose	Destroy the internal data an array.
Access	Private
Prototype	<code>void DestroyArray();</code>
Parameters	None
Return	None

DetachIterator()

Purpose	Remove the association of an iterator from an array.		
Access	Protected		
Prototype	<code>void DetachIterator(LArrayIterator *inIterator) const;</code>		
Parameters	This method takes the following parameter:		
	<code>LArrayIterator*</code>	<code>inIterator</code>	A pointer to the iterator.
Return	None		

FetchIndexOf()

Purpose	Returns the index of the specified item within the array.	
Access	Virtual, Public	
Prototype	<pre>virtual ArrayIndexT FetchIndexOf(const void *inItem, UInt32 inItemSize)</pre>	
Parameters	This method takes the following parameters:	
	<pre>const void *</pre>	<pre>inItem</pre> A pointer to the item to search for.
	<pre>UInt32</pre>	<pre>inItemSize</pre> The size of the array item.
Return	Returns <code>index_Bad</code> if the item is not in the array.	

FetchIndexOfKey()

Purpose	Return the index of the item with the specified Key value.	
Access	Virtual, Public	
Prototype	<pre>virtual ArrayIndexT FetchIndexOfKey(const void *inKey)</pre>	
Parameters	This method takes the following parameter:	
	<pre>const void *</pre>	<pre>inKey</pre> A pointer to the key to search for.
Return	Returns the index of the item.	

FetchInsertIndexOf()

Purpose	Return the index at which the specified item would be inserted.
---------	---

Access	Virtual, Public		
Prototype	<pre>virtual ArrayIndexT FetchInsertIndexOf(const void *inItem, UInt32 inItemSize)</pre>		
Parameters	This method takes the following parameters:		
	const void *	inItem	A pointer to the item to search for.
	UInt32	inItemSize	The size of the array item.
Return	Returns <code>index_Last</code> if the Array is not sorted or if the item is <code>nil</code> .		

FetchInsertIndexOfKey()

Purpose	Return the index at which an item with the specified Key would be inserted.		
Access	Virtual, Public		
Prototype	<pre>virtual ArrayIndexT FetchInsertIndexOfKey(const void *inKey)</pre>		
Parameters	This method takes the following parameter:		
	const void *	inKey	A pointer to the key to search for.
Return	Returns <code>index_Last</code> if the Array is not sorted or if the item is <code>nil</code> .		

FetchItemAt()

Purpose	Pass back the item at the specified index.
Access	Virtual, Public
Prototype	<pre>virtual Boolean FetchItemAt(ArrayIndexT inAtIndex,</pre>

LArray

```
void *outItem,  
USInt32 &ioItemSize ) const;  
virtual Boolean FetchItemAt(  
ArrayIndexT inAtIndex,  
void *outItem ) const;
```

Parameters	This method takes the following parameters:		
	ArrayIndexT	inAtIndex	The index to retrieve from.
	void*	outItem	The pointer to the retrieved item. Caller must make sure that this points a buffer large enough to hold the item data.
	USInt32&	ioItemSize	The size of the item.
Return	Returns true if an item exists at inIndex (and sets outItem). Returns false if inIndex is out of range (and leaves outItem unchanged).		

GetComparator()

Purpose	This method returns the value of the mComparator data member.
Access	Public
Prototype	<code>LComparator* GetComparator() const;</code>
Parameters	None
Return	A pointer to the LComparator .

GetCount()

Purpose	This method returns the value of the mItemCount data member.
Access	Public
Prototype	<code>USInt32 GetCount() const;</code>

Parameters	None
Return	USInt32 indicating the value of mItemCount .

GetItemPtr()

Purpose	Returns a pointer to the start of an item's data within the internal storage Handle.			
Access	Virtual, Public			
Prototype	<pre>virtual void* GetItemPtr(ArrayIndexT inAtIndex) const;</pre>			
Parameters	<p>This method takes the following parameter:</p> <table><tr><td>ArrayIndexT</td><td>inAtIndex</td><td>The index to retrieve from.</td></tr></table>	ArrayIndexT	inAtIndex	The index to retrieve from.
ArrayIndexT	inAtIndex	The index to retrieve from.		
Return	A pointer to the start of the item data.			
Remarks	<p>WARNING: The return pointer references information inside a relocatable block. This pointer will become invalid if the Handle block moves. Call Lock() and then Unlock() where necessary.</p> <p>WARNING: For sorted arrays, be careful when changing the data using the pointer. If your changes alter the sorting order, call InvalidateSort() so that the array's internal flags correctly reflect the sorting state. Then call Sort() afterwards if you still want the array to be sorted.</p>			

GetItemSize()

Purpose	This method returns the value of the mItemSize data member.
Access	Virtual, Public
Prototype	<pre>virtual USInt32 GetItemSize(ArrayIndexT inIndex) const;</pre>
Parameters	This method takes the following parameter:

LArray

	ArrayIndexT	inIndex	The index to retrieve the size of.
Return	The value of mItemSize in a USInt32.		

GetItemsHandle()

Purpose	Return Handle used to store data for array items.
Access	Public
Prototype	<code>Handle GetItemsHandle() const;</code>
Parameters	None
Return	The handle to the array items.

GrabItemRangeSize()

Purpose	This method returns the total size in bytes of the items in the array between (inclusive) the specified start and end indices.		
Access	Inline, Virtual, Protected		
Prototype	<code>virtual USInt32 GrabItemRangeSize(ArrayIndexT inStartIndex, ArrayIndexT inEndIndex) const;</code>		
Parameters	This method takes the following parameters:		
	ArrayIndexT	inStartIndex	The starting index.
	ArrayIndexT	inEndIndex	The ending index.
Return	A size in bytes stored in a USInt32.		

GrabItemSize()

Purpose	This data member returns the value of the mItemSize data member.	
Access	Virtual, Protected	
Prototype	<pre>virtual UInt32 GrabItemSize(ArrayIndexT inIndex) const;</pre>	
Parameters	This method takes the following parameter: ArrayIndexT inIndex The index (unused).	
Return	UInt32 containing the value of the mItemSize member.	

InitArray()

Purpose	This is an internal method which initializes the data members for an array.	
Access	Private	
Prototype	<pre>void InitArray(UInt32 inItemSize, LComparator *inComparator, Boolean inIsSorted, Boolean inKeepSorted);</pre>	
Parameters	This method takes the following parameters: UInt32 inItemSize The size of the items for the array. LComparator* inComparator The pointer to the comparator object.	

LArray

	Boolean	inIsSorted	A value indicating whether the array is sorted or not.
	Boolean	inKeepSorted	A value indicating whether to keep the array sorted or not.
Return	None		

InsertItemsAt()

Purpose	Insert items at the specified position in an array.		
Access	Virtual, Public		
Prototype	<pre>virtual ArrayIndexT InsertItemsAt(UInt32 inCount, ArrayIndexT inAtIndex, const void *inItem, UInt32 inItemSize);</pre>		
Parameters	This method takes the following parameters:		
	UInt32	inCount	The number of items to insert.
	ArrayIndexT	inAtIndex	The index to begin inserting at.
	const void*	inItem	A pointer to the item to insert.
	UInt32	inItemSize	The size of the item in bytes.
Return	The index at which items were inserted. This can differ from the input value of inAtIndex as described in the Remarks.		
Remarks	<p>All items are set to the same value, as specified by inItem. inItem may be nil, in which case the data for the inserted items is unspecified (but space is allocated).</p> <p>inAtIndex is adjusted if necessary:</p> <ul style="list-style-type: none">• > to sorted position if array is kept sorted• > to after last item if inAtIndex is too big		

-
- > to 1 if inAtIndex is too small
-

InternalAdjustAllocation()

Purpose	Called internally to change the size of the storage used.							
Access	Virtual, Protected							
Prototype	<pre>virtual void InternalAdjustAllocation(UInt32 inItemAllocation, UInt32 inDataAllocation);</pre>							
Parameters	This method takes the following parameter:							
	<table> <tr> <td>UInt32</td><td>inItemAllocation</td><td>The the number of items to allocate space for.</td></tr> <tr> <td>UInt32</td><td>inDataAllocation</td><td>The the amount of data to allocate space for.</td></tr> </table>	UInt32	inItemAllocation	The the number of items to allocate space for.	UInt32	inDataAllocation	The the amount of data to allocate space for.	
UInt32	inItemAllocation	The the number of items to allocate space for.						
UInt32	inDataAllocation	The the amount of data to allocate space for.						
Remarks	Fixed-size item Array only stores data, so inItemAllocation is ignored.							

InternalCopyItem()

Purpose	Set value of destination item to that of the source item.							
Access	Virtual, Protected							
Prototype	<pre>virtual void InternalCopyItem(ArrayIndexT inSourceIndex, ArrayIndexT inDestIndex);</pre>							
Parameters	This method takes the following parameters:							
	<table> <tr> <td>ArrayIndexT</td><td>inSourceIndex</td><td>The source index.</td></tr> <tr> <td>ArrayIndexT</td><td>inDestIndex</td><td>The destination index.</td></tr> </table>	ArrayIndexT	inSourceIndex	The source index.	ArrayIndexT	inDestIndex	The destination index.	
ArrayIndexT	inSourceIndex	The source index.						
ArrayIndexT	inDestIndex	The destination index.						
Return	None							

InternalMoveItem()

Purpose	Move an item from one position to another in an array. The net result is the same as removing the item and inserting at a new position.	
Access	Virtual, Protected	
Prototype	<pre>virtual void InternalMoveItem(ArrayIndexT inFromIndex, ArrayIndexT inToIndex, void *inBuffer);</pre>	
Parameters	This method takes the following parameters:	
	void*	inBuffer A pointer to the data buffer.
	ArrayIndexT	inSourceIndex The source index.
	ArrayIndexT	inDestIndex The destination index.
Return	None	

InternalSwapItems()

Purpose	Swap the values of the Items at the specified indices. This is an internal method for this class.	
Access	Virtual, Protected	
Prototype	<pre>virtual void InternalSwapItems(ArrayIndexT inIndexA, ArrayIndexT inIndexB, void *inBuffer);</pre>	
Parameters	This method takes the following parameters:	
	void*	inBuffer A pointer to the data buffer.
	ArrayIndexT	inIndexA The first index.
	ArrayIndexT	inIndexB The second index.

Return	None
--------	------

InvalidateSort()

Purpose	This method sets the mIsSorted data member to false.
Access	Public
Prototype	<code>void InvalidateSort();</code>
Parameters	None
Return	None

IsKeptSorted()

Purpose	This method returns the value of the mKeepSorted data member.
Access	Inline, Public
Prototype	<code>Boolean IsKeptSorted() const;</code>
Parameters	None
Return	Boolean true or false indicating the value of mKeepSorted .

IsSorted()

Purpose	This method returns the value of the mIsSorted data member.
Access	Inline, Public
Prototype	<code>Boolean IsSorted() const;</code>
Parameters	None
Return	Boolean true or false indicating the value of mIsSorted .

ItemsInserted()

Purpose	Notify Iterators associated with an array that items have been inserted.		
Access	Protected		
Prototype	<pre>void ItemsInserted(UInt32 inCount, ArrayIndexT inAtIndex);</pre>		
Parameters	This method takes the following parameters:		
	UInt32	inCount	The number of items that have been inserted.
	ArrayIndexT	inAtIndex	The index of the insertion.
Return	None		

ItemsRemoved()

Purpose	Notify Iterators associated with an Array that items have been removed		
Access	Protected		
Prototype	<pre>void ItemsRemoved(UInt32 inCount, ArrayIndexT inAtIndex);</pre>		
Parameters	This method takes the following parameters:		
	UInt32	inCount	The number of items that have been removed.
	ArrayIndexT	inAtIndex	The index of the removal.
Return	None		

LinearSearch()

Purpose	Return the index of the specified item, searching linearly from the start of the array.	
Access	Protected	
Prototype	<pre>ArrayIndexT LinearSearch(const void *inItem, UInt32 inItemSize) const;</pre>	
Parameters	This method takes the following parameters:	
	UInt32	inItemSize The size of the item to search for.
	const void*	inItem The pointer to the item to search for.
Return	The index at which the item is located, <code>index_Bad</code> if not found.	

LinearSearchByKey()

Purpose	Return the index of the item with the specified key, searching linearly from the start of the array.	
Access	Protected	
Prototype	<pre>ArrayIndexT LinearSearchByKey(const void *inKey) const;</pre>	
Parameters	This method takes the following parameter:	
	const void*	inKey A pointer to the key to search for.
Return	The index at which the item is located, <code>index_Bad</code> if not found.	

LArray

Lock()

Purpose	Lock the Handle that stores the data for the items in the array.
Access	Public
Prototype	<code>void Lock() const;</code>
Parameters	None
Return	None
Remarks	This class maintains a lock count, so each call to Lock() should be balanced by a corresponding call to Unlock() .

MoveItem()

Purpose	Move an item from one position to another in an array. The net result is the same as removing the item and inserting at a new position.		
Access	Virtual, Public		
Prototype	<code>virtual void MoveItem(ArrayIndexT inFromIndex, ArrayIndexT inToIndex);</code>		
Parameters	This method takes the following parameters:		
	ArrayIndexT	inFromIndex	The source index.
	ArrayIndexT	inToIndex	The destination index.
Return	None		
Remarks	Does nothing if either index is out of range or if the array is kept sorted (since moving could invalidate the sort).		

PeekItem()

Purpose	Pass back the data for the item at the specified index. This is used internally to read item data.		
Access	Virtual, Protected		
Prototype	<pre>virtual void PeekItem(ArrayIndexT inAtIndex, void *outItem) const;</pre>		
Parameters	This method takes the following parameters:		
	ArrayIndexT	inAtIndex	The array index to peek at.
	void *	outItem	The pointer to the item to return data for.
Return	None		

PokeItem()

Purpose	Store data for the item at the specified index.		
Access	Virtual, Protected		
Prototype	<pre>virtual void PokeItem(ArrayIndexT inAtIndex, const void *inItem, UInt32 inItemSize);</pre>		
Parameters	This method takes the following parameters:		
	ArrayIndexT	inAtIndex	The array index to poke at.
	const void*	outItem	The pointer to the item to store data for.
	UInt32	inItemSize	The size of the data block to store
Return	None		

Remove()

Purpose	Remove an item from an array.		
Access	Virtual, Public		
Prototype	<pre>virtual void Remove(const void *inItem, UInt32 inItemSize);</pre>		
Parameters	This method takes the following parameters:		
	const void *	outItem	The pointer to the item to remove data for.
	UInt32	inItemSize	The size of the data block to remove.
Return	None		

RemoveItemsAt()

Purpose	Remove items from an array starting at a specified position.		
Access	Virtual, Public		
Prototype	<pre>virtual void RemoveItemsAt(UInt32 inCount, ArrayIndexT inAtIndex);</pre>		
Parameters	This method takes the following parameters:		
	ArrayIndexT	inAtIndex	The index to start removing at.
	UInt32	inItemSize	The size of the data block to remove.
Return	None		
Remarks	Does nothing if inAtIndex is out of range. Checks if inCount would remove items past the end of the array, and adjusts it accordingly to remove the items from inAtIndex to the end of the		

array. That means you can pass a large number to remove the items from `inAtIndex` to the end of the array.

SetComparator()

Purpose	Specify the comparator for items in an array.		
Access	Public		
Prototype	<pre>void SetComparator(LComparator *inComparator, Boolean inTakeOwnership);</pre>		
Parameters	This method takes the following parameters:		
	LComparator*	inComparator	The comparator to use.
	Boolean	inTakeOwnership	Indicates the value to set for mOwnsComparator .
Return	None		

SetKeepSorted()

Purpose	Specify whether to keep an array sorted when items change.		
Access	Public		
Prototype	<pre>void SetKeepSorted(Boolean inKeepSorted);</pre>		
Parameters	This method takes the following parameter:		
	Boolean	inKeepSorted	Whether to keep the array sorted or not.
Return	None		

ShiftItems()

Purpose	is an internal method that moves items within the Handle used for internal storage. It moves items in the range <code>inStartIndex</code> to <code>inEndIndex</code> (inclusive).		
Access	Virtual, Protected		
Prototype	<pre>virtual void ShiftItems(ArrayIndexT inStartIndex, ArrayIndexT inEndIndex, SInt32 inIndexShift, SInt32 inDataShift);</pre>		
Parameters	This method takes the following parameters:		
	<code>ArrayIndexT</code>	<code>inStartIndex</code>	The index to start shifting at.
	<code>ArrayIndexT</code>	<code>inEndIndex</code>	The index to end shifting at.
	<code>SInt32</code>	<code>inIndexShift</code>	The amount to shift.
	<code>SInt32</code>	<code>inDataShift</code>	This is unused.
Return	None		

Sort()

Purpose	Sort items in the array.
Access	Virtual, Public
Prototype	<pre>virtual void Sort();</pre>
Parameters	None
Return	None

StoreNewItems()

Purpose	This is an internal method that stores values within the internal storage Handle. Items all have the same value, and space must already have been allocated for them.	
Access	Virtual, Protected	
Prototype	<pre>virtual void StoreNewItems(UInt32 inCount, ArrayIndexT inAtIndex, const void *inItem, UInt32 inItemSize);</pre>	
Parameters	This method takes the following parameters:	
	UInt32	inCount The number of items.
	ArrayIndexT	inAtIndex The index to start storing at.
	const void*	inItem The item to store.
	UInt32	inItemSize The size of the item to store.
Return	None	

SwapItems()

Purpose	Swap the values of the Items at the specified indices. This method does nothing if either index is out of range or if array is kept sorted (since swapping could invalidate the sort).	
Access	Virtual, Public	
Prototype	<pre>virtual void SwapItems(ArrayIndexT inIndexA, ArrayIndexT inIndexB);</pre>	
Parameters	This method takes the following parameters:	

LArray

	ArrayIndexT	inIndexA	The first index.
	ArrayIndexT	inIndexB	The second index.
Return	None		

Unlock()

Purpose	Unlock the Handle that stores the data for the items in the array. This class maintains a lock count, so each call to Lock() should be balanced by a corresponding call to <code>Unlock()</code> .		
Access	Public		
Prototype	<code>void Unlock() const;</code>		
Parameters	None		
Return	None		

ValidIndex()

Purpose	Indicate whether an index is valid (between 1 and the number of items) for the array.					
Access	Public					
Prototype	<pre>Boolean ValidIndex(ArrayIndexT &ioIndex) const;</pre>					
Parameters	This method takes the following parameter: <table><tr><td>ArrayIndexT&</td><td>ioIndex</td><td>The reference to the index to check. If ioIndex is the special flag index_Last, the index's value is changed to the actual index of the last item.</td></tr></table>			ArrayIndexT&	ioIndex	The reference to the index to check. If ioIndex is the special flag index_Last, the index's value is changed to the actual index of the last item.
ArrayIndexT&	ioIndex	The reference to the index to check. If ioIndex is the special flag index_Last, the index's value is changed to the actual index of the last item.				
Return	Boolean indicating whether the index is a valid one, false if it is invalid.					

operator=()

Purpose	Disposes array's existing data and copies data of the specified array. See comments for CopyArray() for detailed information about how the copy is done.		
Access	Public		
Prototype	<code>LArray& operator=(const LArray &inArray);</code>		
Parameters	This method takes the following parameter: <code>const LArray& inArray The reference to the array.</code>		
Return	A reference to the array.		

mComparator

Purpose	This data member stores the comparator for the array.
Access	Protected
Prototype	<code>LComparator *mComparator;</code>

mDataAllocated

Purpose	Stores the data allocated.
Access	Protected
Prototype	<code>USInt32 mDataAllocated;</code>

LArray

mDataStored

Purpose	Stores the amount of space to allocate for storage.
Access	Protected
Prototype	<code>USInt32 mDataStored;</code>

mIsSorted

Purpose	Indicates whether the array is supposed to be sorted.
Access	Protected
Prototype	<code>Boolean mIsSorted;</code>

mItemCount

Purpose	The number of items in the array.
Access	Protected
Prototype	<code>USInt32 mItemCount;</code>

mItemsH

Purpose	The handle to the array items.
Access	Protected
Prototype	<code>Handle mItemsH;</code>

mItemSize

Purpose	The size of the items.
Access	Protected
Prototype	<code>USInt32 mItemSize;</code>

mIteratorListHead

Purpose	The head for the iterator for the array.
Access	Protected
Prototype	<code>mutable LArrayIterator *mIteratorListHead;</code>

mKeepSorted

Purpose	Indicates whether the array is to be kept sorted.
Access	Protected
Prototype	<code>Boolean mKeepSorted;</code>

mLockCount

Purpose	A reference counter to count the number of times the array is locked (decremented by Unlock()).
Access	Protected
Prototype	<code>mutable USInt32 mLockCount;</code>

LArray

mOwnsComparator

Purpose This indicates whether the array owns the comparator.

Access Protected

Prototype `Boolean mOwnsComparator;`

LArrayIterator

Description	LArrayIterator is a PowerPlant class that is useful for iterating through arrays of items.	
Methods	The methods in this class are:	
	LArrayIterator()	~LArrayIterator()
	ArrayDied()	CalcNextIndex()
	CalcPreviousIndex()	Current()
	GetCurrentIndex()	GetNextIterator()
	ItemsInserted()	ItemsRemoved()
	Next()	Previous()
	PtrToCurrent()	PtrToNext()
	PtrToPrevious()	ResetTo()
	SetNextIterator()	
Data Members	The data members in this class are:	
	mNextIterator	mArray
	mCurrIndex	mNextIndex
Operation	<p>LArrayIterator provides the functionality needed to walk through an array from an arbitrary starting point, going forward or backward through the array elements. Each LArrayIterator object is associated with a single array. An array may have an arbitrary number of iterators, but each iterator operates on only one array.</p> <p>To learn more about using iterators, refer to <i>The PowerPlant Book</i> with the rest of the CodeWarrior documentation.</p> <p>Index values are signed, 32-bit integers.</p>	
Source files	(Array Classes)	
	<code>LArrayIterator.h</code>	
	<code>LArrayIterator.cp</code>	

LArrayIterator

See also [LArray](#)

LArrayIterator()

Purpose	The constructor creates an object with the passed-in parameters. It constructs an iterator for an array starting at a particular position.							
Access	Public							
Prototype	<pre>LArrayIterator(const LArray &inArray, ArrayIndexT inPosition = from_Start);</pre>							
Parameters	The parameters for this constructor are:							
	<table><tr><td>const LArray&</td><td>inArray</td><td>The reference to the array to iterate on.</td></tr><tr><td>ArrayIndexT</td><td>inPosition</td><td>The position in the array to set the iterator to. The default is from_Start.</td></tr></table>	const LArray&	inArray	The reference to the array to iterate on.	ArrayIndexT	inPosition	The position in the array to set the iterator to. The default is from_Start.	
const LArray&	inArray	The reference to the array to iterate on.						
ArrayIndexT	inPosition	The position in the array to set the iterator to. The default is from_Start.						

~LArrayIterator()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<pre>~LArrayIterator();</pre>

ArrayDied()

Purpose	This method sets the values of the mCurrIndex and mNextIndex data members to the constant value <code>index_ArrayDied</code> , indicating that the array was deleted.
Access	Protected

Prototype	<code>void ArrayDied();</code>
Parameters	None
Return	None

CalcNextIndex()

Purpose	This method calculates the mNextIndex value based on the value of mCurrIndex . If the index is at the end, the value <code>index_AfterEnd</code> is assigned.
Access	Protected
Prototype	<code>void CalcNextIndex();</code>
Parameters	None
Return	None

CalcPreviousIndex()

Purpose	This method adjusts the internal indexes to access the previous item in an array.
Access	Protected
Prototype	<code>void CalcPreviousIndex();</code>
Parameters	None
Return	None

Current()

Purpose	This method retrieves the current item in the array and (optionally) the item's size.
---------	---

LArrayIterator

Access	Public	
Prototype	<pre>Boolean Current(void *outItem, UInt32 &ioItemSize); Boolean Current(void *outItem);</pre>	
Parameters	The parameters for these methods are:	
	void*	outItem
	The pointer to the array item that is passed out upon exiting this method.	
	UInt32 &	ioItemSize
	A reference to the size of the item retrieved from the array.	
Return	Returns true if the current item exists. Returns false if the current item does not exist, which happens when: <ul style="list-style-type: none">• Current item was deleted• Current item is past end of Array• Current item is before beginning of Array• Array was deleted	
Remarks	You should use the <code>ioItemSize</code> version of this method with arrays having variable-sized elements.	

GetCurrentIndex()

Purpose	This method returns the value of the mCurrIndex data member.
Access	Inline, Public
Prototype	<pre>ArrayIndexT GetCurrentIndex() const;</pre>
Parameters	None
Return	The value of mCurrIndex .

GetNextIterator()

Purpose	This method returns the value of the mNextIterator data member.
Access	Inline, Public
Prototype	<code>LArrayIterator* GetNextIterator();</code>
Parameters	None
Return	The value of mNextIterator .

ItemsInserted()

Purpose	This method keeps track of items that have been inserted into the array at the specified index.		
Access	Protected		
Prototype	<code>void ItemsInserted(UInt32 inCount, ArrayIndexT inAtIndex);</code>		
Parameters	The parameters for this method are:		
	UInt32	inCount	The number of items inserted.
	ArrayIndexT	inAtIndex	The index that is currently pointed to.
	<hr/>		
Return	None		

ItemsRemoved()

Purpose	This method keeps track of items starting at the specified index that have been removed from the array.
Access	Protected

LArrayIterator

Prototype `void ItemsRemoved(UInt32 inCount,
 ArrayIndexT inAtIndex);`

Parameters The parameters for this method are:

UInt32	inCount	The number of items removed.
ArrayIndexT	inAtIndex	The index that is currently pointed to.

Return None

Next()

Purpose This method moves to the next item in the array and passes back a copy of that item and (optionally) the item's size.

Access Public

Prototype `Boolean Next(void *outItem, UInt32 &ioItemSize);`
`Boolean Next(void *outItem);`

Parameters The parameters for these methods are:

void*	outItem	The pointer to the array item that is passed out upon exiting this method.
UInt32 &	ioItemSize	A reference to the size of the item retrieved from the array.

Return Returns true if the next item exists

Returns false if next item does not exist, which happens when:

- Current item is at or past end of the Array
- Array was deleted

Previous()

Purpose	This method moves to the previous item in the array and passes back a copy of that item.							
Access	Public							
Prototype	<pre>Boolean Previous(void *outItem, UInt32 &ioItemSize); Boolean Previous(void *outItem);</pre>							
Parameters	The parameters for these methods are:							
	<table><tr><td>void*</td><td>outItem</td><td>The pointer to the array item that is passed out upon exiting this method.</td></tr><tr><td>UInt32 &</td><td>ioItemSize</td><td>A reference to the size of the item retrieved from the array.</td></tr></table>	void*	outItem	The pointer to the array item that is passed out upon exiting this method.	UInt32 &	ioItemSize	A reference to the size of the item retrieved from the array.	
void*	outItem	The pointer to the array item that is passed out upon exiting this method.						
UInt32 &	ioItemSize	A reference to the size of the item retrieved from the array.						
Return	Returns true if the previous item exists. Returns false if the previous item does not exist, which happens when: <ul style="list-style-type: none">• Current item is at or before the start of the Array• Array was deleted							

PtrToCurrent()

Purpose	Return a pointer to the current item in the array and (optionally) pass back the item's size.	
Access	Public	
Prototype	<pre>void* PtrToCurrent(UInt32 &outItemSize); void* PtrToCurrent();</pre>	
Parameters	The parameter for these methods is:	

LArrayIterator

	<div><div>USInt32 outItemSize</div><div>&</div></div> <div>A reference to the size of the item retrieved from the array.</div>
Return	Returns a pointer to the current item. Returns nil and item size of zero if there is no current item.

PtrToNext()

Purpose	Return a pointer to the next item in the array and (optionally) pass back the item's size.		
Access	Public		
Prototype	<pre>void* PtrToNext(USInt32 &outItemSize); void* PtrToNext();</pre>		
Parameters	The parameter for these methods is: <table><tr><td>USInt32 outItemSize</td><td>A reference to the size of the item retrieved from the array.</td></tr></table>	USInt32 outItemSize	A reference to the size of the item retrieved from the array.
USInt32 outItemSize	A reference to the size of the item retrieved from the array.		
Return	Returns a pointer to the next item. Returns nil and item size of zero if there is no next item.		

PtrToPrevious()

Purpose	Return a pointer to the previous item in the array and (optionally) pass back the item's size.
Access	Public
Prototype	<pre>void* PtrToPrevious(USInt32 &outItemSize); void* PtrToPrevious();</pre>
Parameters	The parameter for these methods is:

	<table><tr><td>USInt32 &</td><td>outItemSize</td><td>A reference to the size of the item retrieved from the array.</td></tr></table>	USInt32 &	outItemSize	A reference to the size of the item retrieved from the array.
USInt32 &	outItemSize	A reference to the size of the item retrieved from the array.		
Return	Returns a pointer to the previous item. Returns nil and item size of zero if there is no previous item.			

ResetTo()

Purpose	Reset the current item to the specified index value. The standard enumeration constants <code>from_Start</code> and <code>from_End</code> are recognized.		
Access	Public		
Prototype	<code>void ResetTo(ArrayIndexT inPosition);</code>		
Parameters	The parameter for this method is:		
	<code>ArrayIndexT</code>	<code>inPosition</code>	The position to set to, using the enumerations of <code>from_End</code> , <code>index_AfterEnd</code> , etc.
Return	None		

SetNextIterator()

Purpose	This method set the value of the mNextIterator data member to the value passed-in.
Access	Protected
Prototype	<code>void SetNextIterator(LArrayIterator *inIterator);</code>
Parameters	The parameter for this method is:

LArrayIterator

	<div>LArrayIterator*</div> <div>inIterator</div> <div>The value to assign to mNextIterator.</div>
Return	None

mNextIterator

Purpose This data member is used to maintain a singly-linked list of Iterators for a particular Array. [LArray](#) traverses this linked list of Iterators at various times.

An Array needs a list of all its current Iterators in order to update the Iterators when the Array changes (adding/removing items or even deleting the Array). This keeps the Iterators in synch with the Array.

Access Protected

Prototype `LArrayIterator *mNextIterator;`

mArray

Purpose This data member stores a reference to the array of elements that are iterated over.

Access Protected

Prototype `const LArray &mArray;`

mCurrIndex

Purpose This data member stores the index of the array element currently pointed to.

Access Protected

Prototype	<code>ArrayIndexT mCurrIndex;</code>
-----------	--------------------------------------

mNextIndex

Purpose	This data member stores the index of the array element next in the array.
---------	---

Access	Protected
--------	-----------

Prototype	<code>ArrayIndexT mNextIndex;</code>
-----------	--------------------------------------

LAttachable

Overview	LAttachable is a PowerPlant class that is used for storing Attachment information in lists.	
Methods	The methods in this class are:	
	LAttachable()	~LAttachable()
	AddAttachment()	ExecuteAttachments()
	GetDefaultAttachable()	RemoveAllAttachments()
	RemoveAttachment()	SetDefaultAttachable()
Data Members	The data members in this class are:	
	sDefaultAttachable	mAttachments
Operation	Typically, you don't need to override any methods in this class. As it exists, the class provides all the implementation that you are likely to need.	
Source files	(Feature Classes)	
	LAttachable.h	
	LAttachable.cp	
See Also	LAttachment	

LAttachable()

Purpose	The copy constructor initializes the mAttachments data member, but does not make a copy of any other member data. The default constructor initializes the mAttachments data member, and also calls SetDefaultAttachable() with a pointer to this object.
Access	Public
Prototypes	<pre>LAttachable(); LAttachable::LAttachable(const LAttachable& inOriginal);</pre>

LAttachable

Parameters	<code>const LAttachable&</code>	<code>inOriginal</code>	This is a reference to the LAttachable object you wish to copy
------------	-------------------------------------	-------------------------	--

~LAttachable()

Purpose	The destructor destroys the LAttachable object, using a call to RemoveAllAttachments() .
Access	Public, Virtual
Prototype	<code>virtual ~LAttachable();</code>

AddAttachment()

Purpose	This method adds an Attachment to the list.	
Access	Public, Virtual	
Prototype	<pre>virtual void AddAttachment(LAttachment *inAttachment, LAttachment *inBefore, Boolean inOwnsAttachment)</pre>	
Parameters	This method has the following parameters:	
LAttachment*	inAttachment	A pointer to the Attachment to add to the list.
LAttachment*	inBefore	A pointer to the attachment to precede in the list. Set this to nil to add to the end of the list.
Boolean	inOwnsAttachment	If true, the Attachable assumes ownership of the Attachment and will delete the Attachment when the Attachable itself is deleted.
Return	None	

ExecuteAttachments()

Purpose	This method tells all associated Attachments to execute themselves for the specified message.	
Access	Virtual, Public	
Prototype	<code>Boolean ExecuteAttachments(MessageT inMessage, void *ioParam);</code>	
Parameters	This method has the following parameters:	
	<code>MessageT</code>	<code>inMessage</code> The message that is passed from the framework.
	<code>void*</code>	<code>ioParam</code> A pointer to the data block that accompanies the message.
Return	The Boolean return value specifies whether the default Host action should be executed. The value is <code>false</code> if any Attachment's Execute() function returns <code>false</code> , otherwise it's <code>true</code> .	

GetDefaultAttachable()

Purpose	This method returns a pointer to the sDefaultAttachable data member.	
Access	Static, Public, Inline	
Prototype	<code>static LAttachable* GetDefaultAttachable()</code>	
Parameters	None	
Return	A pointer to sDefaultAttachable .	

RemoveAllAttachments()

Purpose	This method removes all Attachments from an Attachable. All Attachments owned by this Attachable are deleted.
---------	---

LAttachable

Access	Virtual, Public
Prototype	<code>void RemoveAllAttachments();</code>
Parameters	None
Return	None

RemoveAttachment()

Purpose	This method removes an Attachment from an Attachable. If this Attachable is the owner of the Attachment, the Attachment's owner is set to <code>nil</code> , and the caller should assume control of the Attachment.		
Access	Virtual, Public		
Prototype	<code>void RemoveAttachment(LAttachment *inAttachment);</code>		
Parameters	This method has the following parameter:		
	<code>LAttachment*</code>	<code>inAttachment</code>	The pointer to the Attachment to remove.
Return	None		

SetDefaultAttachable()

Purpose	This method sets the pointer for the sDefaultAttachable data member.		
Access	Static, Public, Inline		
Prototype	<code>static void SetDefaultAttachable(LAttachment* inAttachment);</code>		
Parameters	This method has the following parameter:		
	<code>LAttachment*</code>	<code>inAttachment</code>	The pointer to the Attachment.

Return	None
--------	------

sDefaultAttachable

Purpose	This data member contains the pointer to the default Attachable object.
Access	Protected
Prototype	<code>static LAttachable* sDefaultAttachable;</code>

mAttachments

Purpose	This data member is a list of all the Attachments.
Access	Protected
Prototype	<code>TArray<LAttachment*> *mAttachments;</code>

LAttachment

Description	LAttachment is a PowerPlant class that is used for forming relationships between objects. An Attachment is usually an object that modifies the runtime behavior of another object.	
Methods	The methods in this class are:	
	LAttachment()	~LAttachment()
	Execute()	ExecuteSelf()
	GetExecuteHost()	GetMessage()
	GetOwnerHost()	SetExecuteHost()
	SetMessage()	SetOwnerHost()
Data Members	The data members in this class are:	
	mOwnerHost	mMessage
	mExecuteHost	
Operation	The concept of an Attachment is a powerful underlying concept of PowerPlant's inner workings. For a detailed discussion on this topic, refer to <i>The PowerPlant Book</i> .	
Source files	(Feature Classes)	
	<code>LAttachment.h</code>	
	<code>LAttachment.cp</code>	
See also	LAttachable	
	LYieldAttachment	

LAttachment

LAttachment()

Purpose	The constructors create the object and initialize the data members to the passed-in values.	
Access	Public	
Prototype	<pre>LAttachment(MessageT inMessage, Boolean inExecuteHost); LAttachment(LStream *inStream);</pre>	
Parameters	The parameters for the constructors are:	
MessageT	inMessage	The value to set mMessage to.
Boolean	inExecuteHost	The value to set mExecuteHost to.
LStream*	inStream	The data in Stream must be: <ul style="list-style-type: none">• MessageT—Message that Attachment responds to.• Boolean—Should we execute the Host? (mExecuteHost)• Boolean—Does the Host own this object? (mOwnerHost)

~LAttachment()

Purpose	The destructor destroys the object, removing itself from any Attachments.	
Access	Virtual, Public	
Prototype	<pre>virtual ~LAttachment();</pre>	
Parameters	None	

Execute()

Purpose	If the message passed to this method is the message for which the Attachment is designed, this method calls ExecuteSelf() .	
Access	Virtual, Public	
Prototype	<pre>virtual Boolean Execute(MessageT inMessage, void *ioParam);</pre>	
Parameters	This method has the following parameters:	
	MessageT	inMessage
	The message to which the attachment responds.	
	void*	ioParam
	A pointer to the host attachable object.	
Return	A Boolean with the value of mExecuteHost . The PowerPlant framework uses this value to decide whether the host object should also perform the task in question.	
Remarks	You do not typically need to override this function.	

ExecuteSelf()

Purpose	This method performs the attachment task. You typically want to override this method in your class that inherits from LAttachment.	
Access	Virtual, Public	
Prototype	<pre>void ExecuteSelf(MessageT inMessage, void* ioParam);</pre>	
Parameters	This method has the following parameters:	

LAttachment

	MessageT	inMessage	The message to which the attachment responds.
	void*	ioParam	A pointer to the host attachable object.
Return	None		
Remarks	To get an idea of how to implement this method in your class override, look at classes like LPaintAttachment to see how it is implemented.		

GetExecuteHost()

Purpose	This method is an accessor for the value of mExecuteHost .
Access	Inline, Public
Prototype	Boolean GetExecuteHost();
Parameters	None
Return	A Boolean indicating the value of mExecuteHost .

GetMessage()

Purpose	This method is an accessor for the value of mMessage .
Access	Public, Inline
Prototype	MessageT GetMessage();
Parameters	None
Return	A MessageT indicating the value of mMessage .

GetOwnerHost()

Purpose	This method is an accessor for the value of mOwnerHost .
Access	Public, Inline
Prototype	<code>LAttachable* GetOwnerHost();</code>
Parameters	None
Return	An LAttachable pointer indicating the value of mOwnerHost .

SetExecuteHost()

Purpose	Sets the value of mExecuteHost , that specifies whether the host action should execute after executing all Attachments.
Access	Virtual, Public
Prototype	<code>void SetExecuteHost(Boolean inExecuteHost);</code>
Parameters	This method takes a Boolean to assign to mExecuteHost .
Return	None

SetMessage()

Purpose	Sets the value of mMessage .
Access	Virtual, Public
Prototype	<code>void SetMessage(MessageT inMessage);</code>
Parameters	A MessageT to assign to mMessage .
Return	None

LAttachment

SetOwnerHost()

Purpose	Sets the value of mOwnerHost .
Access	Virtual, Public
Prototype	<code>void SetOwnerHost(LAttachable *inHost);</code>
Parameters	A pointer to an LAttachable to assign to mOwnerHost .
Return	None

mOwnerHost

Purpose	This data member points to the host attachable object.
Access	Protected
Prototype	<code>LAttachable *mOwnerHost;</code>

mMessage

Purpose	This data member contains the message value to which the attachable object responds.
Access	Protected
Prototype	<code>MessageT mMessage;</code>

mExecuteHost

Purpose	This data member indicates whether or not to execute the host object action.
Access	Protected

Prototype Boolean mExecuteHost;

LBeepAttachment

Overview	LBeepAttachment is a PowerPlant class that is used for beeping when a message is received.
Methods	The methods in this class are: LBeepAttachment() ExecuteSelf()
Data Members	There are no data members in this class.
Operation	This object is a simple Attachment designed to respond to any message you indicate, typically a click message. When you create the object with LBeepAttachment() , you specify the message to which you want the Attachment to respond. When attached to any host, this Attachment beeps when the appropriate message is received.
Source files	(Utility Classes) UAttachments.h UAttachments.cp
See also	LAttachment LPane

LBeepAttachment()

Purpose	The constructor for this object is a placeholder.
Access	Public
Prototype	<pre>LBeepAttachment(MessageTinMessage, Boolean inExecuteHost) ; LBeepAttachment(LStream *inStream);</pre>
Parameters	The parameters for these constructors are:

LBeepAttachment

MessageT	inMessage	The message that we respond to.
LStream*	inStream	A pointer to a stream object that contains the information to create the LBeepAttachment object.

ExecuteSelf()

Purpose This method beeps using `SysBeep()`. This is an override of [ExecuteSelf\(\)](#) in [LAttachment](#).

LBorderAttachment

Overview	LBorderAttachment is a PowerPlant class that is used for drawing a border within the Frame of a Pane. It is often used in conjunction with LPaintAttachment.
Methods	The methods in this class are: LBorderAttachment() ExecuteSelf()
Data Members	The data members in this class are: mPenState mForeColor mBackColor
Operation	This Attachment is designed to respond to the <code>msg_DraworPaint</code> message. When you create this Attachment, you specify a PenState, foreground and background colors, and whether the host should also draw.
Source files	(Utility Classes) <code>UAttachments.h</code> <code>UAttachments.cp</code>
See also	LAttachment LPane

LBorderAttachment()

Purpose	The constructor creates the object containing the passed-in parameters.
Access	Public
Prototype	<pre>LBorderAttachment(PenState *inPenState, RGBColor *inForeColor, RGBColor *inBackColor, Boolean inExecuteHost);</pre>

LBorderAttachment

```
LBorderAttachment( LStream *inStream );
```

Parameters	The parameters for these constructors are:		
	PenState*	inPenState	A pointer to a PenState.
	RGBColor*	inForeColor	A pointer to an RGBColor.
	RGBColor*	inBackColor	A pointer to an RGBColor.
	Boolean	inExecuteHost	A Boolean indicating the value to be set for mExecuteHost .
	LStream*	inStream	A pointer to a stream object that contains the information to create the LBorderAttachment object.

ExecuteSelf()

Purpose	This method draws a rectangular border. This is an override of ExecuteSelf() in LAttachment .
---------	---

mPenState

Purpose	This member holds the pen information.
Access	Protected
Prototype	PenState mPenState;

mForeColor

Purpose	This member holds the foreground color information.
Access	Protected
Prototype	RGBColor mForeColor;

mBackColor

Purpose This member holds the background color information.

Access Protected

Prototype `RGBColor mBackColor;`

LBroadcaster

Overview	LBroadcaster is a PowerPlant class that is used for creating Broadcast and Listen relationships between objects. A broadcaster object communicates to a Listener object by sending messages. This is udeful if you want to have an object that can send messages to many other objects.	
Methods	The methods in this class are:	
	LBroadcaster()	~LBroadcaster()
	AddListener()	BroadcastMessage()
	IsBroadcasting()	RemoveListener()
	StartBroadcasting()	StopBroadcasting()
Data Members	The data members in this class are:	
	mListeners	mIsBroadcasting
Operation	<p>A Broadcaster sends messages to its Listeners. LBroadcaster is an abstract, mix-in class. Broadcasters have a list of Listeners to which they send messages via the BroadcastMessage() method. You attach a Listener to a Broadcaster using the AddListener() function.</p> <p>Classes derived from LBroadcaster should call the BroadcastMessage() method whenever they want to announce some happening in the code, such as a change in state. For example, the destructor for LBroadcaster sends a <code>msg_BroadcasterDied</code> message to its Listeners.</p>	
Source files	(Feature Classes)	
	<code>LBroadcaster.h</code>	
	<code>LBroadcaster.cp</code>	
See also	LListener	

LBroadcaster

LBroadcaster()

Purpose	The copy constructor for LBroadcaster makes a copy of the state of mIsBroadcasting data member. Listener links are not copied. The default constructor has no Listeners initially.
Access	Public
Prototype	<code>LBroadcaster();</code> <code>LBroadcaster(const LBroadcaster& inOriginal);</code>
Parameters	None
Return	No return value for a constructor

~LBroadcaster()

Purpose	The destructor destroys the LBroadcaster object, and notifies all Listeners that this Broadcaster object is going away.
Access	Public, Virtual
Prototype	<code>virtual ~LBroadcaster();</code>
Parameters	None
Return	None

AddListener()

Purpose	This method adds a Listener to a Broadcaster object. You must associate Broadcasters with Listeners, not the converse: <code>theBroadcaster->AddListener(theListener); //</code> <code>correct</code> <code>theListener->AddBroadcaster(theBroadcaster); //</code> <code>incorrect!</code>
---------	--

This method takes care of notifying the Listener to update its list of Broadcasters.

Access	Public			
Prototype	<code>void AddListener(LListener *inListener);</code>			
Parameters	<p>This method takes the following parameter:</p> <table><tr><td><code>LListener</code> *</td><td><code>inListener</code></td><td>a pointer to the LListener object that will be associated with the LBroadcaster</td></tr></table>	<code>LListener</code> *	<code>inListener</code>	a pointer to the LListener object that will be associated with the LBroadcaster
<code>LListener</code> *	<code>inListener</code>	a pointer to the LListener object that will be associated with the LBroadcaster		
Return	None			

BroadcastMessage()

Purpose	<p>This method broadcasts a message to all associated Listeners. Listeners are associated with a Broadcaster using AddListener(). This method does not send a message to its Listeners if broadcasting is turned off. Broadcasting is turned on using StartBroadcasting(), and can be turned off using StopBroadcasting().</p>							
Access	Public							
Prototype	<code>void BroadcastMessage(MessageT inMessage, void *ioParam = nil);</code>							
Parameters	<p>This method takes the following parameters:</p> <table> <tr> <td><code>MessageT</code></td><td><code>inMessage</code></td><td>This is the message you want to send to all Listeners.</td></tr> <tr> <td><code>void</code></td><td><code>*ioParam</code></td><td>This is a pointer to a parameter structure that the Listeners may use. The default value is nil, meaning no parameter.</td></tr> </table>		<code>MessageT</code>	<code>inMessage</code>	This is the message you want to send to all Listeners.	<code>void</code>	<code>*ioParam</code>	This is a pointer to a parameter structure that the Listeners may use. The default value is nil, meaning no parameter.
<code>MessageT</code>	<code>inMessage</code>	This is the message you want to send to all Listeners.						
<code>void</code>	<code>*ioParam</code>	This is a pointer to a parameter structure that the Listeners may use. The default value is nil, meaning no parameter.						
Return	None							
Remarks	<ul style="list-style-type: none"> Does not send message if broadcasting is turned off via StopBroadcasting(). 							

- The meaning of the `ioParam` parameter depends on the message.

IsBroadcasting()

Purpose	This method tells you whether broadcasting is turned on or not.
Access	Public, Inline
Prototype	<code>Boolean IsBroadcasting() ;</code>
Parameters	None
Return	A Boolean indicating “true” if broadcasting is turned on, or “false” otherwise.

RemoveListener()

Purpose	This method tells a Listener to remove itself from the broadcast list. In response to this, a Listener should update its list of Broadcasters.	
Access	Public	
Prototype	<code>void RemoveListener(LListener *inListener);</code>	
Parameters	This method takes the following parameter:	
	LListener *	<code>inListener</code> A pointer to the Listener object that we want to remove from the broadcast list.
Return	None	

StartBroadcasting()

Purpose	This method enables broadcasting, by setting the mIsBroadcasting data member to true.
---------	---

Access	Public, Inline
Prototype	<code>void StartBroadcasting();</code>
Parameters	None
Return	None

StopBroadcasting()

Purpose	This method disables broadcasting, by setting the mIsBroadcasting data member to false. A Broadcaster that has broadcast turned off does not broadcast messages.
Access	Public, Inline
Prototype	<code>void StopBroadcasting();</code>
Parameters	None
Return	None

mListeners

Purpose	This data member is a list of pointers to LListener objects. In general, you should use the AddListener() and RemoveListener() methods to access this list.
Access	Protected
Prototype	TArray <LListener*> mListeners;

mIsBroadcasting

Purpose	This data member indicates whether broadcasting is enabled or not. This member should be accessed using the IsBroadcasting() method.
Access	Protected

LBroadcaster

Prototype Boolean mIsBroadcasting;

LButton

Overview	<p>LButton is a PowerPlant class that implements a button. The graphic for the button needs to be stored as a resource of one of these types:</p> <ul style="list-style-type: none">• ICN#• ICON• PICT								
Methods	<p>The methods in this class are:</p> <table><tbody><tr><td>LButton()</td><td>DrawGraphic()</td></tr><tr><td>DrawSelf()</td><td>HotSpotAction()</td></tr><tr><td>HotSpotResult()</td><td>PointIsInFrame()</td></tr><tr><td>SetGraphics()</td><td>SetGraphicsType()</td></tr></tbody></table>	LButton()	DrawGraphic()	DrawSelf()	HotSpotAction()	HotSpotResult()	PointIsInFrame()	SetGraphics()	SetGraphicsType()
LButton()	DrawGraphic()								
DrawSelf()	HotSpotAction()								
HotSpotResult()	PointIsInFrame()								
SetGraphics()	SetGraphicsType()								
Data Members	<p>The data members in this class are:</p> <table><tbody><tr><td>mGraphicsType</td><td>mNormalID</td></tr><tr><td>mPushedID</td><td></td></tr></tbody></table>	mGraphicsType	mNormalID	mPushedID					
mGraphicsType	mNormalID								
mPushedID									
Operation	<p>When you create an object of this type, you provide two resource ID numbers. One is for the graphical element in its non-depressed state, and the other is for the depressed (user-pushed) state.</p> <p>If you use the <code>ICN#</code> resource for the button, the Mac OS automatically picks the icon family member that best matches the display settings of the monitor on which it is rendered.</p> <p>Although buttons are usually small, you can use a large <code>PICT</code> as a button. Note that <code>ICON</code> and <code>ICN#</code> resources describe images that are of Mac OS-standard, specific dimensions.</p> <p>When the user releases the mouse button with the cursor inside the button perimeter, the button sends a message to its Listeners.</p>								
Source files	<p>(Pane Classes)</p> <p><code>LButton.h</code></p> <p><code>LButton.cp</code></p>								

LButton

Ancestors [LBroadcaster](#)

[LControl](#)

[LPane](#)

LButton()

Purpose The constructors create new objects from the passed-in parameters.

Access Public

Prototype

```
LButton();  
LButton( const LButton &inOriginal );  
LButton( const SPaneInfo &inPaneInfo,  
MessageT inClickedMessage,  
OSType inGraphicsType,  
ResIDT inNormalID,  
ResIDT inPushedID );  
LButton( LStream *inStream );
```

Parameters These constructors have the following parameters:

const LButton&	inOriginal	A reference to the LButton object you want to copy.
const SPaneInfo&	inPaneInfo	A reference to the SPaneInfo object that is the super view.
MessageT	inClickedMessage	The message sent when the button is clicked.
OSType	inGraphicsType	Graphics Type ('ICN#', 'ICON', or 'PICT')
ResIDT	inNormalID	Resource ID for normal graphic
ResIDT	inPushedID	Resource ID for pushed graphic
LStream*	inStream	A pointer to a stream object that contains the information to create the LButton object.

DrawGraphic()

Purpose	Draw the graphic for a button. The pane must already be focused; refer to LPane for more information.		
Access	Virtual, Protected		
Prototype	<code>virtual void DrawGraphic(ResIDT inGraphicID);</code>		
Parameters	The parameter for this method is:		
	ResIDT	inGraphicID	The resource ID of the graphic to draw in the button.
Return	None		

DrawSelf()

Purpose	This method is an override of the base class method DrawSelf() in LPane . It draws the normal graphic in the button.
---------	--

HotSpotAction()

Purpose	This method is an override of the base class method HotSpotAction() in LControl . It causes the buttons to toggle between two graphics, depending on whether the mouse is inside or outside the button.
---------	---

HotSpotResult()

Purpose	This method is an override of the base class method HotSpotResult() in LControl . It broadcasts a message to Listeners when the button is clicked.
---------	--

PointIsInFrame()

Purpose This method is an override of the base class method [PointIsInFrame\(\)](#) in [LPane](#). It gives you information about whether a point lies within a given frame.

SetGraphics()

Purpose Sets the graphic resources to use for drawing the button.

Access Virtual, Public

Prototype `virtual void SetGraphics(ResIDT inNormalID,
ResIDT inPushedID);`

Parameters This method has the following parameters:

ResIDT	inNormalID	Resource ID for normal graphic
ResIDT	inPushedID	Resource ID for pushed graphic

Return None

SetGraphicsType()

Purpose This method sets the value of the [mGraphicsType](#) data member.

Access Virtual, Public

Prototype `virtual void SetGraphicsType(
OSType inGraphicsType);`

Parameters This method has the following parameters:

OSType	inGraphicsType	The value to set for mGraphicsType .
--------	----------------	--

Return	None
--------	------

mGraphicsType

Purpose	This method holds the resource type descriptor for the graphic elements of the button. It should be one of the following: <ul style="list-style-type: none">• ICN#• ICON'• 'PICT'
Access	Protected
Prototype	<code>OSType mGraphicsType;</code>

mNormalID

Purpose	This data member holds the resource ID for the normal button graphic.
Access	Protected
Prototype	<code>ResIDT mNormalID;</code>

mPushedID

Purpose	This data member holds the resource ID for the pushed button graphic.
Access	Protected
Prototype	<code>ResIDT mPushedID;</code>

LCaption

Overview	LCaption is a PowerPlant class that is used for displaying static text.	
Methods	The methods in this class are:	
	LCaption()	~LCaption()
	DrawSelf()	GetDescriptor()
	GetTextTraitsID()	GetValue()
	SetDescriptor()	SetTextTraitsID()
	SetValue()	
Data Members	The data members in this class are:	
	mText	mTxtrID
Operation	<p>This class uses a text traits resource to specify characteristics such as font, size, style, color, and justification. LCaption uses the UTextDrawing class to draw text.</p> <p>You can configure the text and the text traits resource in the Constructor resource editor, or you can may these characteristics at runtime.</p> <p>It is possible that you will encounter a drawing problem if you modify the contents of the caption at runtime. DrawSelf() uses the DrawWithJustification() method of UTextDrawing, that does not erase the previous contents of the caption. The best way to erase the contents is to attach an LEraseAttachment object to the caption.</p>	
Source files	(Pane Classes)	
	LCaption.h	
	LCaption.cp	
See also	LAttachment	
	LEraseAttachment	
	UTextDrawing	

LCaption

LCaption()

Purpose The constructors create the object from the passed-in parameters.

Access Public

Prototype

```
LCaption();  
LCaption( const LCaption &inOriginal );  
LCaption( const SPaneInfo &inPaneInfo,  
ConstStringPtr inString,  
ResIDT inTextTraitsID );  
LCaption( LStream *inStream );
```

Parameters The parameters for these constructors are:

SPaneInfo&	inPaneInfo	A reference to the Pane that hosts this caption.
LCaption&	inOriginal	A reference to a caption to be copied.
ConstStringPtr	inString	A pointer to the caption text.
ResIDT	inTextTraitsID	The text traits for the caption.
LStream*	inStream	A pointer to a stream object that contains the information to create the LCaption object.

~LCaption()

Purpose The destructor destroys the object.

Access Virtual, Public

Prototype `virtual ~LCaption();`

Parameters None

DrawSelf()

Purpose	Draw the caption using the colors and traits specified.
Access	Virtual, Protected
Prototype	<code>virtual void DrawSelf();</code>
Parameters	None
Return	None

GetDescriptor()

Purpose	Return contents of the caption as a string.			
Access	Virtual, Public			
Prototype	<pre>virtual StringPtr GetDescriptor(Str255 outDescriptor) const;</pre>			
Parameters	<p>The parameter for this method is:</p> <table><tr><td>Str255</td><td>outDescriptor</td><td>A pointer to a buffer, to contain the string on exit.</td></tr></table>	Str255	outDescriptor	A pointer to a buffer, to contain the string on exit.
Str255	outDescriptor	A pointer to a buffer, to contain the string on exit.		
Return	A pointer to a string, using the StringPtr type.			

GetTextTraitsID()

Purpose	This method returns the value of the mTxtrID data member.
Access	Inline, Public
Prototype	<code>ResIDT GetTextTraitsID() const;</code>
Parameters	None
Return	A resource ID, using the ResIDT type.

GetValue()

Purpose	Return the integer value represented by the text of the caption, held in the mText data member.		
Access	Virtual, Public		
Prototype	<code>virtual SInt32 GetValue() const;</code>		
Parameters	None		
Return	SInt32 containing the value of mText .		

SetDescriptor()

Purpose	Set contents of the caption from a string.		
Access	Virtual, Public		
Prototype	<code>virtual void SetDescriptor(ConstStringPtr inDescriptor);</code>		
Parameters	The parameter for this method is:		
	ConstStringPtr	inDescriptor	A pointer to a buffer, containing the string to set for the caption.
Return	None		

SetTextTraitsID()

Purpose	This member sets the value of the mTxtrID data member to a resource ID for a resource containing the text traits information.		
Access	Public		
Prototype	<code>void SetTextTraitsID(ResIDT inTxtrID);</code>		

Parameters	The parameter for this method is:		
	ResIDT	inTxtrID	A ResIDT containing the resource ID value to put in the mTxtrID data member.
Return	None		

SetValue()

Purpose	Set a caption to the text representation of an integer value.		
Access	Virtual, Public		
Prototype	<code>virtual void SetValue(SInt32 inValue);</code>		
Parameters	The parameter for this method is:		
	SInt32	inValue	The integer value to set the caption to.
Return	None		

mText

Purpose	This data member contains the caption string.
Access	Protected
Prototype	<code>LStr255 mText;</code>

mTxtrID

Purpose	This data member contains a resource ID for a resource that contains the text traits information for the caption.
Access	Protected
Prototype	<code>ResIDT mTxtrID;</code>

LCicnButton

Overview	LCicnButton is a PowerPlant class that is very similar to LButton . It is different from LButton in that it uses the Mac OS 'cicn' color icon resource format as the graphical element in the button.	
Methods	The methods in this class are: LCicnButton() DrawSelf() HotSpotResult() ~LCicnButton() HotSpotAction() SetCicns()	
Data Members	The data members in this class are: mNormalID mNormalCicnH mPushedID mPushedCicnH	
Operation	As with the LButton class, you provide resource ID numbers for the button in both its normal and pushed states.	
Source files	(Pane Classes) LCicnButton.h LCicnButton.cp	
See also	LButton LControl	

LCicnButton()

Purpose	The constructors create new objects from the passed-in parameters.
Access	Public
Prototype	<pre>LCicnButton(); LCicnButton(const LCicnButton &inOriginal); LCicnButton(const SPaneInfo &inPaneInfo, MessageT inClickedMessage,</pre>

LCicnButton

```
ResIDT inNormalID,  
ResIDT inPushedID );  
LCicnButton( LStream *inStream );
```

Parameters These constructors have the following parameters:

const LCicnButton &	inOriginal	A reference to the object you want to copy.
const SPaneInfo&	inPaneInfo	A reference to the SPaneInfo object that is the
MessageT	inClickedMes sage	The message sent when the button is depressed.
ResIDT	inNormalID	Resource ID for normal graphic
ResIDT	inPushedID	Resource ID for pushed graphic
LStream*	inStream	A pointer to a stream object that contains the information to create the LButton object.

~LCicnButton()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LCicnButton();</code>
Parameters	None

DrawSelf()

Purpose	This method behaves just like the one in LCicnButton() named DrawSelf() , except it draws a 'cicn' button instead.
---------	--

HotSpotAction()

Purpose This method behaves just like the one in [LCicnButton\(\)](#) named [HotSpotAction\(\)](#), except it works for a 'cicn' button instead.

HotSpotResult()

Purpose This method behaves just like the one in [LCicnButton\(\)](#) named [HotSpotResult\(\)](#), except it works for a 'cicn' button instead.

SetCicns()

Purpose Specify new ID's for the normal and pushed 'cicn' resources.

Access Virtual, Public

Prototype `virtual void SetCicns(ResIDT inNormalID,
ResIDT inPushedID);`

Parameters The parameters for this method are:

ResIDT	inNormalID	The resource ID for the normal graphic.
ResIDT	inPushedID	The resource ID for the pushed graphic.

Return None

mNormalID

Purpose This data member is just like the one in [LButton](#) named [mNormalID](#), except it is for 'cicn' buttons.

LCicnButton

mPushedID

Purpose	This data member is just like the one in LButton named mPushedID , except it is for 'cicn' buttons.
---------	---

mNormalCicnH

Purpose	This data member holds a handle to the resource for the normal 'cicn'.
Access	Protected
Prototype	CIconHandle mNormalCicnH;

mPushedCicnH

Purpose	This data member holds a handle to the resource for the pushed 'cicn'.
Access	Protected
Prototype	CIconHandle mPushedCicnH;

LCleanupTask

Overview	LCleanupTask is a PowerPlant class that is an abstract base class for any operation which needs to be performed at application shutdown time. It patches <code>ExitToShell()</code> to ensure that tasks are performed even if the user force-quits application.	
Methods	The methods in this class are: LCleanupTask() CleanUpAtExit() ETSPatch() ~LCleanupTask() DoCleanup()	
Data Members	The data members in this class are: sCleanupTaskList sNewETSRoutine sOldETSRoutine	
Operation	This class may be used as a mix-in to ensure that an object cleans up after itself when the application quits.	
Source files	(Networking Classes) <code>LCleanupTask.h</code> <code>LCleanupTask.cp</code>	
See also	LOpenTptUDPEndpoint	

LCleanupTask()

Purpose	The constructor creates the object. Don't use the copy constructor.
Access	Protected
Prototype	<code>LCleanupTask();</code> <code>LCleanupTask(LCleanupTask&); /* copy ctor */</code>
Parameters	None

LCleanupTask

~LCleanupTask()

Purpose	The destructor destroys the LCleanupTask object.
Access	Virtual, Protected
Prototype	<code>~LCleanupTask() ;</code>

CleanUpAtExit()

Purpose	Should be called only once whenever the application quits, either by the normal Quit command exit or by a user abort. This method runs all of the cleanup tasks and removes them from the list.
Access	Static, Public
Prototype	<code>void CleanUpAtExit() ;</code>
Parameters	None
Return	None

DoCleanup()

Purpose	Override to perform any task that must be performed before the app quits.
Access	Virtual, Protected
Prototype	<code>void DoCleanup() ;</code>
Parameters	None
Return	None

ETSPatch()

Purpose	Patch <code>ExitToShell()</code> .
Access	Static, Protected
Prototype	<code>void ETSPatch();</code>
Parameters	None
Return	None

sCleanupTaskList

Purpose	The task list for cleanup.
Access	Static, Protected
Prototype	<code>LInterruptSafeList*sCleanupTaskList;</code>

sOldETSRoutine

Purpose	Temporary storage for the <code>ExitToShell()</code> routine.
Access	Static, Protected
Prototype	<code>UniversalProcPtrsOldETSRoutine;</code>

sNewETSRoutine

Purpose	The pointer to the new <code>ExitToShell()</code> routine.
Access	Static, Protected
Prototype	<code>UniversalProcPtrsNewETSRoutine;</code>

LClipboard

Overview	LClipboard is a PowerPlant class that is used for supporting the global clipboard in your application. It contains all the methods for setting and getting data of arbitrary type and size.	
Methods	The methods in this class are:	
	LClipboard()	~LClipboard()
	ExecuteSelf()	ExportSelf()
	GetClipboard()	GetData()
	GetDataSelf()	ImportSelf()
	SetData()	SetDataSelf()
Data Members	The data members in this class are:	
	sClipboard	mImportPending
	mExportPending	
Operation	To use LClipboard, you create a single instance of it in your application, and attach the clipboard object to your application object. As an attachment, LClipboard looks for the msg_Event message. If a suspend or resume event occurs, LClipboard converts the local clipboard to the global scrap, or vice versa. You always get a pointer to the clipboard with the static method GetClipboard() . To learn more information on using this object, refer to <i>The PowerPlant Book</i> .	
Source files	(Support Classes) LClipboard.h LClipboard.cp	
Ancestors	LAttachment	

LClipboard

LClipboard()

Purpose	The constructor creates the clipboard for your application.
Access	Public
Prototype	<code>LClipboard();</code>
Parameters	None

~LClipboard()

Purpose	The destructor destroys the clipboard object.
Access	Virtual, Public
Prototype	<code>virtual ~LClipboard();</code>

ExecuteSelf()

Purpose	Clipboard watches for Suspend and Resume events. The <code>inMessage</code> parameter will be <code>msg_Event</code> . The <code>ioParam</code> will be an <code>EventRecord*</code> . This method is an override of ExecuteSelf() in LAttachment .
---------	---

ExportSelf()

Purpose	Export the data in a local scrap to the global scrap.
Access	Protected
Prototype	<code>virtual void ExportSelf();</code>
Parameters	None

Return	None
Remarks	This implementation does nothing since this class uses the global scrap when setting and getting clipboard data. Subclasses should override this method if they maintain a local scrap.

GetClipboard()

Purpose	This method returns the value of sClipboard , a pointer to the clipboard.
Access	Inline, Static, Public
Prototype	<code>static LClipboard* GetClipboard();</code>
Parameters	None
Return	A pointer to an LClipboard object.
Remarks	There is only one LClipboard object in an application, since this is a static data member.

GetData()

Purpose	<p>This method passes back the data in the Clipboard of the specified type in a Handle, and returns the size of the data.</p> <p>If <code>ioDataH</code> is <code>nil</code>, the data is not passed, but its size is returned. Otherwise, <code>ioDataH</code> is resized if necessary to hold all the data.</p> <p>This is a wrapper method which imports the global scrap if necessary, then calls a lower level function to actually retrieve the data.</p>
Access	Virtual, Public
Prototype	<code>virtual SInt32 GetData(ResType inDataType, Handle ioDataH);</code>
Parameters	This method has the following parameters:

	ResType	inDataType	This is the resource ID for the type of data.
	Handle	ioDataH	This is a handle to the data that is resized to accommodate the data.
Return	SInt32 indicating the size of the data, in bytes.		

GetDataSelf()

Purpose	Pass back the data in the Clipboard of the specified type in a Handle and return the size of the data If ioDataH is nil, the data is not passed data, but its size is returned Otherwise, ioDataH is resized if necessary to hold all the data This implementation gets the data from the global scrap. Subclasses should override to maintain a local scrap.		
Access	Virtual, Protected		
Prototype	<code>virtual SInt32 GetDataSelf(ResType inDataType, Handle ioDataH);</code>		
Parameters	This method has the following parameters:		
	ResType	inDataType	This is the resource ID for the type of data.
	Handle	ioDataH	This is a handle to the data that is resized to accommodate the data.
Return	SInt32 indicating the size of the data, in bytes.		

ImportSelf()

Purpose	This implementation does nothing since this class uses the global scrap when setting and getting clipboard data. Subclasses should override this method if they maintain a local scrap.
Access	Virtual, Protected
Prototype	<code>virtual void ImportSelf();</code>
Parameters	None
Return	None

SetData()

Purpose	This method sets the Clipboard contents to the data specified by a pointer and length, or it sets the Clipboard contents to the data in a Handle.		
Access	Virtual, Public		
Prototype	<code>virtual void SetData(ResType inDataType, Ptr inDataPtr, SInt32 inDataLength, Boolean inReset); virtual void SetData(ResType inDataType, Handle inDataH, Boolean inReset);</code>		
Parameters	This method has the following parameters:		
	ResType	inDataType	This is the resource ID for the type of data.
	Ptr	inDataPtr	This is a pointer to some data to put on the Clipboard.
	Handle	inDataH	This is a handle to the data that is resized to accommodate the data.

LClipboard

	SInt32	inDataLength	This is the size of the data pointed to by <code>inDataPtr</code> .
	Boolean	inReset	This specifies whether to clear the existing contents of the Clipboard before storing the new data.
Return	None		

SetDataSelf()

Purpose	This method sets the Clipboard contents to the data specified by a pointer and length. This implementation sets the data in the global scrap. Subclasses should override this method to maintain a local scrap.		
Access	Virtual, Protected		
Prototype	<pre>virtual void SetDataSelf(ResType inDataType, Ptr inDataPtr, SInt32 inDataLength, Boolean inReset);</pre>		
Parameters	This method has the following parameters:		
	ResType	inDataType	This is the resource ID for the type of data.
	Ptr	inDataPtr	This is a pointer to some data to put on the Clipboard.
	SInt32	inDataLength	This is the size of the data pointed to by <code>inDataPtr</code> .
	Boolean	inReset	This specifies whether to clear the existing contents of the Clipboard before storing the new data.
Return	None		

sClipboard

Purpose	This data member is a static pointer to the Clipboard. There is only one Clipboard, hence the reason for the static pointer.
Access	Protected
Prototype	<code>static LClipboard *sClipboard;</code>

mImportPending

Purpose	This data member indicates whether an import is pending. If the scrap changed while we were suspended, then we set the import pending flag to true. We don't need to import the data now, since the program may never request the data. This member would be set on receiving a Resume event.
Access	Protected
Prototype	<code>Boolean mImportPending;</code>

mExportPending

Purpose	This data member indicates whether an export is pending. This member would be set on receiving a Suspend event.
Access	Protected
Prototype	<code>Boolean mExportPending;</code>

LCmdEnablerAttachment

Overview	LCmdEnablerAttachment is a PowerPlant class that is used for coupling commands and updating chores in PowerPlant object relationships.
Methods	The methods in this class are: LCmdEnablerAttachment() ExecuteSelf()
Data Members	The data members in this class are: mCmdToEnable
Operation	<p>This Attachment responds to the <code>msg_CommandStatus</code> message. When you create the Attachment, you also specify the command that should be enabled.. When it executes, this Attachment enables the menu item associated with the command. It also prevents the host from executing, because it has already enabled the related command. It does not do any item manipulation such as setting a mark in the menu.</p> <p>You can use this Attachment as a model for handling other updating tasks. For example, you might create a check mark Attachment that puts a check mark in front of a menu item.</p>
Source files	(Utility Classes) <code>UAttachments.h</code> <code>UAttachments.cp</code>
See also	LAttachment LPane

LCmdEnablerAttachment()

Purpose	The constructor creates the object containing the passed-in parameters.
Access	Public

LCmdEnablerAttachment

Prototype	<pre>LCmdEnablerAttachment(CommandT inCmdToEnable); LCmdEnablerAttachment(LStream* inStream);</pre>	
Parameters	The parameters for these constructors are:	
	Command T	inCmdToEnable A command that we want to enable.
	LStream*	inStream A pointer to a stream object that contains the information to create the LCmdEnablerAttachment object.

ExecuteSelf()

Purpose	This method decides whether to enable the command, then sets mExecuteHost appropriately. This is an override of ExecuteSelf() in LAttachment .
---------	--

mCmdToEnable

Purpose	This member holds the value of the command for which we want to control enabling behavior.
Access	Protected
Prototype	<pre>CommandT mCmdToEnable;</pre>

LColorEraseAttachment

Overview	LColorEraseAttachment is a PowerPlant class that is used for erasing the Frame of a Pane.
Methods	The methods in this class are: LColorEraseAttachment() ExecuteSelf()
Data Members	The data members in this class are: mForeColor mBackColor
Operation	This Attachment is used with the <code>msg_DrawOrPrint</code> message.
Source files	(Utility Classes) <code>UAttachments.h</code> <code>UAttachments.cp</code>
See also	LAttachment LPane

LColorEraseAttachment()

Purpose	The constructor creates the object containing the passed-in parameters.
Access	Public
Prototype	<pre>LColorEraseAttachment(RGBColor *inBackColor, Boolean inExecuteHost); LColorEraseAttachment(PenState* inPenState, RGBColor *inForeColor, RGBColor *inBackColor, Boolean inExecuteHost); LColorEraseAttachment(LStream *inStream);</pre>
Parameters	The parameters for these constructors are:

LColorEraseAttachment

PenState*	inPenState	A pointer to a PenState.
RGBColor*	inForeColor	A pointer to an RGBColor.
RGBColor*	inBackColor	A pointer to an RGBColor.
Boolean	inExecuteHost	A Boolean indicating the value to be set for mExecuteHost .
LStream*	inStream	A pointer to a stream object that contains the information to create the LColorEraseAttachment object.

ExecuteSelf()

Purpose	This method erases a rectangular area. This is an override of ExecuteSelf() in LAttachment .
---------	--

mForeColor

Purpose	This member holds the foreground color information.
Access	Protected
Prototype	<code>RGBColor mForeColor;</code>

mBackColor

Purpose	This member holds the background color information.
Access	Protected
Prototype	<code>RGBColor mBackColor;</code>

LCommander

Overview LCommander is a PowerPlant class that is used for managing the state of menu items while they are the active target or in the active chain of command.

Methods The methods in this class are:

<u>LCommander()</u>	<u>~LCommander()</u>
<u>AddSubCommander()</u>	<u>AllowSubRemoval()</u>
<u>AllowTargetSwitch()</u>	<u>AttemptQuit()</u>
<u>AttemptQuitSelf()</u>	<u>BeTarget()</u>
<u>DontBeTarget()</u>	<u>FindCommandStatus()</u>
<u>GetDefaultCommander()</u>	<u>GetLatentSub()</u>
<u>GetSuperCommander()</u>	<u>GetTarget()</u>
<u>GetTopCommander()</u>	<u>GetUpdateCommandStatus()</u>
<u>HandleKeyPress()</u>	<u>InitCommander()</u>
<u>IsOnDuty()</u>	<u>IsSyntheticCommand()</u>
<u>IsTarget()</u>	<u>ObeyCommand()</u>
<u>PostAction()</u>	<u>PostAnAction()</u>
<u>ProcessCommand()</u>	<u>ProcessCommandStatus()</u>
<u>ProcessKeyPress()</u>	<u>PutChainOnDuty()</u>
<u>PutOnDuty()</u>	<u>RemoveSubCommander()</u>
<u>RestoreTarget()</u>	<u>SetDefaultCommander()</u>
<u>SetLatentSub()</u>	<u>SetSuperCommander()</u>
<u>SetTarget()</u>	<u>SetUpdateCommandStatus()</u>
<u>SwitchTarget()</u>	<u>TakeChainOffDuty()</u>
<u>TakeOffDuty()</u>	

Data Members The data members in this class are:

LCommander

	sTopCommander	sTarget
	sDefaultCommander	sUpdateCommandStatus
	mSuperCommander	mSubCommanders
	mOnDuty	
Operation	A commander may be on or off duty, and has functions to manage the duty state. This is an important concept, because an off-duty commander will not receive or respond to events. To learn more about Commanders and how to use them with PowerPlant, refer to <i>The PowerPlant Book</i> .	
Source files	(Commander Classes) LCommander.h LCommander.cp	
Ancestors	LAttachable	
See Also	LMenu LMenuBar	

LCommander()

Purpose	The constructor creates objects from the passed-in parameters.
Access	Public
Prototype	<pre>LCommander(); LCommander(const LCommander &inOriginal); LCommander(LCommander *inSuper);</pre>
Parameters	The parameters for these constructors are:

<code>const LCommander&</code>	<code>inOriginal</code>	The reference to the object to copy.
<code>LCommander*</code>	<code>inSuper</code>	The pointer to the super commander.

~LCommander()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LCommander();</code>

AddSubCommander()

Purpose	Adds a subcommander to the commander object.			
Access	Virtual, Protected			
Prototype	<code>virtual void AddSubCommander(LCommander *inSub);</code>			
Parameters	<div>The parameter for this method is:<table><tr><td><code>LCommander*</code></td><td><code>inSub</code></td><td>The pointer to the subcommander.</td></tr></table></div>	<code>LCommander*</code>	<code>inSub</code>	The pointer to the subcommander.
<code>LCommander*</code>	<code>inSub</code>	The pointer to the subcommander.		
Return	None			

AllowSubRemoval()

Purpose	This method indicates whether removal of subcommanders is allowed.
Access	Virtual, Public

LCommander

Prototype	<code>virtual Boolean AllowSubRemoval (LCommander* inSub) ;</code>		
Parameters	The parameter for this method is:		
	<code>LCommander*</code>	<code>inSub</code>	Unused.
Return	Boolean indicating whether the removal is allowed.		

AllowTargetSwitch()

Purpose	<p>Grant permission to switch the target to the specified Commander.</p> <p>Both the current target and <code>inNewTarget</code> must be Subordinates of this Commander.</p> <p>This method passes the request up the command chain by calling AllowTargetSwitch() for its SuperCommander.</p> <p>Subclasses should override this function if they wish to disallow a target switch under certain circumstances.</p>		
Access	Virtual, Public		
Prototype	<code>virtual Boolean AllowTargetSwitch (LCommander *inNewTarget) ;</code>		
Parameters	The parameter for this method is:		
	<code>LCommander*</code>	<code>inNewTarget</code>	The pointer to the new target.
Return	A Boolean indicating whether the target switch was allowed.		

AttemptQuit()

Purpose	This method asks all subcommanders if we can quit.
Access	Virtual, Public

Prototype	virtual Boolean AttemptQuit(long inSaveOption);	
Parameters	The parameter for this method is:	
	long inSaveOption	The save option to pass to AttemptQuitSelf() .
Return	Boolean indicating whether it is valid to quit, false if not.	

AttemptQuitSelf()

Purpose	This is a private method that indicates whether it is valid to quit.	
Access	Virtual, Protected	
Prototype	virtual Boolean AttemptQuitSelf(SInt32 inSaveOption);	
Parameters	The parameter for this method is:	
	long inSaveOption	Unused.
Return	Boolean indicating whether it is valid to quit, false if not.	

BeTarget()

Purpose	This method is called when the Commander is becoming the Target. Subclasses should override this function if they wish to behave differently when they are and are not the target. At entry, the class variable sTarget points to this command.	
Access	Virtual, Protected	
Prototype	virtual void BeTarget();	
Parameters	None	

LCommander

Return None

DontBeTarget()

Purpose This method is called when the commander will no longer be the target.

Subclasses should override this function if they wish to behave differently when they are and are not the target. At entry, the class variable [sTarget](#) points to this commander. [sTarget](#) will be changed soon afterwards to the new target.

Access Virtual, Protected

Prototype `virtual void DontBeTarget()`

Parameters None

Return None

FindCommandStatus()

Purpose This method passes back the status of a command.

Subclasses must override to enable/disable and mark commands. PowerPlant uses the enabling and marking information to set the appearance of Menu items.

Access Virtual, Public

Prototype `virtual void FindCommandStatus(
CommandT inCommand,
Boolean &outEnabled,
Boolean &outUsesMark,
UInt16 &outMark,
Str255 outName);`

Parameters The parameters for this method are:

	Command T	inCommand	The command passed to this method.
	Boolean&	outEnabled	Tells whether the command is enabled.
	Boolean&	outUsesMark	Tells whether the command uses a mark.
	UInt16&	outMark	The mark information for the command.
	Str255	outName	The name string.

Return	None
--------	------

GetDefaultCommander()

Purpose	This method returns the value of the sDefaultCommander data member.
Access	Inline, Static, Public
Prototype	<code>static LCommander* GetDefaultCommander();</code>
Parameters	None
Return	Returns the value of sDefaultCommander .

GetLatentSub()

Purpose	Find the latent subcommander of a Commander. A Commander may have one or no latent subcommander.
Access	Virtual, Public
Prototype	<code>virtual LCommander* GetLatentSub();</code>
Parameters	None

LCommander

Return Returns the latent subcommander in a pointer to an LCommander.

GetSuperCommander()

Purpose Return the value of the [mSuperCommander](#) data member.

Access Inline, Public

Prototype `LCommander* GetSuperCommander();`

Parameters None

Return Returns the value of [mSuperCommander](#).

GetTarget()

Purpose Return the value of the [sTarget](#) data member.

Access Inline, Static, Public

Prototype `static LCommander* GetTarget();`

Parameters None

Return Returns the value of [sTarget](#).

GetTopCommander()

Purpose Returns the value of [sTopCommander](#) data member.

Access Inline, Static, Public

Prototype `static LCommander* GetTopCommander();`

Parameters None

Return Returns the value of [sTopCommander](#).

GetUpdateCommandStatus()

Purpose	Returns the value of sUpdateCommandStatus data member.
Access	Inline, Static, Public
Prototype	<code>static Boolean GetUpdateCommandStatus()</code>
Parameters	None
Return	Returns the value of sUpdateCommandStatus .

HandleKeyPress()

Purpose	This method processes key presses.		
Access	Virtual, Public		
Prototype	<pre>virtual Boolean HandleKeyPress(const EventRecord &inKeyEvent);</pre>		
Parameters	This method has the following parameter:		
	<div>const</div> <div>EventRecord&</div>	<div>inKeyEvent</div>	<div>The key press event.</div>
Return	Boolean of true if the key press was handled, else false.		

InitCommander()

Purpose	Private function for initializing data members from the constructors.
Access	Private
Prototype	<code>void InitCommander(LCommander *inSuper);</code>
Parameters	This method has the following parameter:

LCommander

LCommander *	inSuper	Pointer to the SuperCommander.
--------------	---------	--------------------------------

Return	None
--------	------

IsOnDuty()

Purpose	Indicate whether this commander is on duty.
Access	Public
Prototype	Boolean IsOnDuty() const;
Parameters	None
Return	Boolean of true if the command is on duty, else false.

IsSyntheticCommand()

Purpose	Indicate whether a command is synthetic. If so, pass back the associated Menu ID and item number. If not synthetic, outMenuID and outMenuItem are undefined.
Access	Static, Public
Prototype	static Boolean IsSyntheticCommand(CommandT inCommand, ResIDT &outMenuID, SInt16 &outMenuItem);
Parameters	This method has the following parameters:

Command T	inCommand	Command passed to this method.
ResIDT&	outMenuID	The menu resource ID.
SInt16&	outMenuItem	The menu item number.

Return	Boolean of <code>true</code> if the command is synthetic, else <code>false</code> .
Remarks	<p>A synthetic command number has the Menu ID in the high 16 bits and the item number in the low 16 bits, with the result negated.</p> $\text{syntheticCmd} = - (\text{MenuID} \ll 16) - \text{ItemNumber}$ <p>A synthetic command is the negative of the value returned by the Toolbox traps <code>MenuSelect()</code> and <code>MenuKey()</code>.</p> <p>The LMenu and LMenuBar classes return synthetic command numbers for menu items whose actual command number is <code>cmd_UseMenuItem</code>. You should use synthetic command numbers when the menu choice depends on the runtime name of the menu item. For example, an item in a Font menu.</p>

IsTarget()

Purpose	Return whether this Commander is the target
Access	Public
Prototype	<code>Boolean IsTarget() const;</code>
Parameters	None
Return	Return <code>true</code> if the Commander is the target, else <code>false</code> .

ObeyCommand()

Purpose	<p>Issue a command to a Commander.</p> <p>Subclasses must override this method in order to respond to commands.</p>
Access	Virtual, Public
Prototype	<pre>virtual Boolean ObeyCommand(CommandT inCommand, void *ioParam);</pre>

LCommander

Parameters	This method has the following parameters:		
	Command T	inCommand	Command passed to this method.
	void*	ioParam	The data block to accompany the command.
Return	Returns whether the command was handled (<code>true</code>) or not (<code>false</code>).		

PostAction()

Purpose	<p>If an attachment doesn't intercept the <code>msg_PostAction</code>, this function will post the action up to the supercommander of this object. The default NULL parameter will effectively clear or "commit" the last semantic action.</p> <p>If an attachment didn't process the posting and there's no supercommander, this function will attempt to Redo, Finalize, and delete the action.</p>		
Access	Virtual, Public		
Prototype	<code>virtual void PostAction(LAction *inAction);</code>		
Parameters	This method has the following parameter:		
	LAction*	inAction	The action to post.
Return	None		

PostAnAction()

Purpose	<p>Post the action to the present target. This is a static member function that can be used by anything needing to post an action.</p> <p>If there is no present target, function will attempt to Redo, Finalize, and delete the action.</p>
---------	--

Access	Static, Public			
Prototype	<code>static void PostAnAction(LAction *inAction);</code>			
Parameters	<div>This method has the following parameter:</div> <table><tr><td>LAction*</td><td>inAction</td><td>The action to post.</td></tr></table>	LAction*	inAction	The action to post.
LAction*	inAction	The action to post.		
Return	None			

ProcessCommand()

Purpose	Issue a command to a Commander. This function lets Attachments handle the Command before calling the normal ObeyCommand() function.						
Access	Virtual, Public						
Prototype	<pre>virtual Boolean ProcessCommand(CommandT inCommand, void *ioParam);</pre>						
Parameters	<div>This method has the following parameters:</div> <table><tr><td>Command T</td><td>inCommand</td><td>Command passed to this method.</td></tr><tr><td>void*</td><td>ioParam</td><td>The data block to accompany the command.</td></tr></table>	Command T	inCommand	Command passed to this method.	void*	ioParam	The data block to accompany the command.
Command T	inCommand	Command passed to this method.					
void*	ioParam	The data block to accompany the command.					
Return	Returns whether the command was handled (true) or not (false).						

ProcessCommandStatus()

Purpose	Pass back the status of a command. This method lets Attachments set the command status before calling the normal FindCommandStatus() method.
---------	--

LCommander

Access	Virtual, Public		
Prototype	virtual void ProcessCommandStatus(CommandT inCommand, Boolean &outEnabled, Boolean &outUsesMark, UInt16 &outMark, Str255 outName);		
Parameters	The parameters for this method are:		
	Command T	inCommand	Command passed to this method.
	Boolean&	outEnabled	Tells whether the command is enabled.
	Boolean&	outUsesMark	Tells whether the command uses a mark.
	UInt16&	outMark	The mark information for the command.
	Str255	outName	The name string.
Return	None		

ProcessKeyPress()

Purpose	Processes keystrokes.		
Access	Virtual, Public		
Prototype	<pre>virtual Boolean ProcessKeyPress(const EventRecord &inKeyEvent);</pre>		
Parameters	This method has the following parameter:		
	const EventRecord&	inKeyEvent	The key press event.
Return	Returns true if handled, else false.		

PutChainOnDuty()

Purpose	Put on duty a chain of Commanders. This is a wrapper function that guarantees that a Commander can't be put on duty until all its Superiors are on duty.
Access	Protected
Prototype	<code>void PutChainOnDuty();</code>
Parameters	None
Return	None

PutOnDuty()

Purpose	Called when a Commander is going on duty. Subclasses should override this function if they wish to behave differently when on duty than when off duty.
Access	Virtual, Protected
Prototype	<code>virtual void PutOnDuty();</code>
Parameters	None
Return	None

RemoveSubCommander()

Purpose	This method removes a Subcommander.
Access	Virtual, Protected
Prototype	<code>virtual void RemoveSubCommander(LCommander *inSub);</code>
Parameters	This method has the following parameter:

LCommander

	<div><div>LCommander *</div><div>inSub</div><div>The pointer to the commander to remove.</div></div>
Return	None

RestoreTarget()

Purpose	Set target to the Commander which was the target when this Commander was last on duty.
Access	Virtual, Public
Prototype	<code>virtual void RestoreTarget();</code>
Parameters	None
Return	None

SetDefaultCommander()

Purpose	Sets the value of the sDefaultCommander data member.			
Access	Inline, Static, Public			
Prototype	<pre>static void SetDefaultCommander(LCommander *inCommander);</pre>			
Parameters	<div>This method has the following parameter:</div> <table><tr><td>LCommander*</td><td>inCommander</td><td>The pointer to the commander to set.</td></tr></table>	LCommander*	inCommander	The pointer to the commander to set.
LCommander*	inCommander	The pointer to the commander to set.		
Return	None			

SetLatentSub()

Purpose	Specify the Subcommander that will be put on duty when this Commander is put on duty.		
	This method does nothing if this Commander is already on duty. <i>inSub</i> may be <i>nil</i> , in which case this Commander will have no Latent Subcommander. This will also be the case if <i>inSub</i> is not a Subordinate of this Commander (which raises a Signal).		
Access	Virtual, Public		
Prototype	<code>virtual void SetLatentSub(LCommander *inSub);</code>		
Parameters	This method has the following parameter:		
	LCommander *	inSub	The pointer to the commander to remove.
Return	None		

SetSuperCommander()

Purpose	Command chain maintenance method.		
Access	Virtual, Public		
Prototype	<code>virtual void SetSuperCommander(LCommander *inSuper);</code>		
Parameters	This method has the following parameter:		
	LCommander *	inSuper	The pointer to the Supercommander to set.
Return	None		

SetTarget()

Purpose	Set the Target class variable. Called internally.				
Access	Static, Protected				
Prototype	<code>static void SetTarget(LCommander *inNewTarget);</code>				
Parameters	This method has the following parameter:				
	<table><tr><td>LCommander *</td><td>inNewTarget</td><td>The pointer to the Commander to set as the Target.</td></tr></table>	LCommander *	inNewTarget	The pointer to the Commander to set as the Target.	
LCommander *	inNewTarget	The pointer to the Commander to set as the Target.			
Return	None				

SetUpdateCommandStatus()

Purpose	This method sets the value of the sUpdateCommandStatus data member.				
Access	Static, Public				
Prototype	<code>static void SetUpdateCommandStatus(Boolean inDirty);</code>				
Parameters	This method has the following parameter:				
	<table><tr><td>Boolean</td><td>inDirty</td><td>The value indicating whether to mark as dirty or not.</td></tr></table>	Boolean	inDirty	The value indicating whether to mark as dirty or not.	
Boolean	inDirty	The value indicating whether to mark as dirty or not.			
Return	None				

SwitchTarget()

Purpose	Try to change Target and return whether the specified Commander did indeed become the Target.
---------	---

A Target switch can fail if some superior of the old and new Target disallows the switch. For example, this could happen when performing data entry validation.

Access	Static, Public				
Prototype	<pre>static Boolean SwitchTarget(LCommander *inNewTarget);</pre>				
Parameters	This method has the following parameter:				
	<table> <tr> <td>LCommander *</td><td>inNewTarget</td><td>The pointer to the Commander to set as the Target.</td></tr> </table>	LCommander *	inNewTarget	The pointer to the Commander to set as the Target.	
LCommander *	inNewTarget	The pointer to the Commander to set as the Target.			
Return	Boolean indicating whether the switch was allowed (true) or not (false).				

TakeChainOffDuty()

Purpose	<p>Take a chain of Commanders off duty.</p> <p>This is a wrapper function that traverses a command chain from the Commander that firsts receives this message up to the specified Commander. The <code>inUpToCmdr</code> is not taken off duty.</p> <p>It is an error if <code>inUpToCmdr</code> is not a Superior of the Commander that first receives this message. Note that this includes the case where <code>inUpToCmdr</code> is the Commander that first receives this message (since a Commander is not a Superior of itself). <code>nil</code> is a valid value for <code>inUpToCmdr</code>, since the Commander at the top of a chain of command has a <code>nil</code> <code>SuperCommander</code>.</p>				
Access	Protected				
Prototype	<pre>void TakeChainOffDuty(const LCommander *inUpToCmdr);</pre>				
Parameters	This method has the following parameter:				
	<table> <tr> <td>LCommander *</td><td>inUpToCmdr</td><td>The pointer to the Commander to take off duty.</td></tr> </table>	LCommander *	inUpToCmdr	The pointer to the Commander to take off duty.	
LCommander *	inUpToCmdr	The pointer to the Commander to take off duty.			

LCommander

Return None

TakeOffDuty()

Purpose A Commander is going off duty. Subclasses should override this method if they wish to behave differently when on duty than when off duty.

Access Protected

Prototype `virtual void TakeOffDuty();`

Parameters None

Return None

sTopCommander

Purpose The top Commander pointer.

Access Protected

Prototype `static LCommander *sTopCommander;`

sTarget

Purpose The Target pointer.

Access Protected

Prototype `static LCommander *sTarget;`

sDefaultCommander

Purpose	The default Commander pointer.
Access	Protected
Prototype	<code>static LCommander *sDefaultCommander;</code>

sUpdateCommandStatus

Purpose	Indicates whether command status should be updated.
Access	Protected
Prototype	<code>static Boolean sUpdateCommandStatus;</code>

mSuperCommander

Purpose	The pointer to the Supercommander.
Access	Protected
Prototype	<code>LCommander *mSuperCommander;</code>

mSubCommanders

Purpose	An array of pointers to Subcommanders.
Access	Protected
Prototype	<code>TArray<LCommander*> mSubCommanders;</code>

LCommander

mOnDuty

Purpose	Indicates whether the Commander is on duty or not.
Access	Protected
Prototype	ETriState mOnDuty;

LCommanderPane

Overview	LCommanderPane is a PowerPlant class is used for creating Attachments to facilitate drawing, printing, clicking, and command handling.
Methods	The methods in this class are: LCommanderPane() ~LCommanderPane()
Data Members	There are no data members in this class.
Operation	This object is a subclass of LCommander and LPane . An object of this class does nothing by itself, but becomes powerful when combined with Attachments which control all drawing, printing, clicking, and command handling.
Source files	(Utility Classes) UAttachments.h UAttachments.cp
See also	LAttachment LCommander LPane

LCommanderPane()

Purpose	The constructors create the object using the passed-in parameters.
Access	Public
Prototype	<pre>LCommanderPane(SPaneInfo &inPaneInfo, LCommander *inSuper); LCommanderPane(LStream *inStream);</pre>
Parameters	The parameters for these constructors are:

LCommanderPane

SPaneInfo&	inPaneInfo	A reference to the Pane that works in conjunction with LCommanderPane.
LCommander*	inSuper	A pointer to the SuperView that works in conjunction with LCommanderPane.
LStream*	inStream	A pointer to a stream object that contains the information to create the LCommanderPane object.

~LCommanderPane()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LCommanderPane();</code>

LComparator

Description	LComparator is a PowerPlant class that is used for comparing objects. Comparators are objects that know how to compare to other objects or structures.									
Methods	The methods in this class are: <table><tr><td>LComparator()</td><td>~LComparator()</td></tr><tr><td>Clone()</td><td>Compare()</td></tr><tr><td>CompareToKey()</td><td>GetComparator()</td></tr><tr><td>IsEqualTo()</td><td>IsEqualToKey()</td></tr></table>		LComparator()	~LComparator()	Clone()	Compare()	CompareToKey()	GetComparator()	IsEqualTo()	IsEqualToKey()
LComparator()	~LComparator()									
Clone()	Compare()									
CompareToKey()	GetComparator()									
IsEqualTo()	IsEqualToKey()									
Data Members	The data members in this class are: sComparator									
Operation	Subclasses will need to implement the Compare() method. The compare Compare() result should be one of the following: <ul style="list-style-type: none">• < 0 — object/data item 1 is less than object/data item 2• 0 — object/data item 1 is equal to object/data item 2• > 0 — object/data item 1 is greater than object/data item 2									
Source files	(Array Classes) LComparator.h LComparator.cp									

LComparator()

Purpose	The constructor is a placeholder, it doesn't do anything for this class.
Access	Public
Prototype	<code>LComparator();</code>
Parameters	None

LComparator

~LComparator()

Purpose	The destructor is a placeholder, it doesn't do anything for this class.
Access	Virtual, Public
Prototype	<code>~LComparator();</code>

Clone()

Purpose	This method simply creates this object by calling <code>new</code> .
Access	Virtual, Public
Prototype	<code>virtual LComparator* Clone();</code>
Parameters	None
Return	A pointer to a new comparator object.

Compare()

Purpose	This method takes two pointers to data blocks, and their sizes, and determines what the result of the comparison should be.
Access	Virtual, Public
Prototype	<pre>virtual SInt32 Compare(const void* inItemOne, const void* inItemTwo, USInt32 inSizeOne, USInt32 inSizeTwo) const;</pre>
Parameters	This method has the following parameters:

<code>const void*</code>	<code>inItemOne</code>	A data pointer to the first item.
<code>const void*</code>	<code>inItemTwo</code>	A data pointer to the second item.
<code>USInt32</code>	<code>inSizeOne</code>	The size of the first item.
<code>USInt32</code>	<code>inSizeTwo</code>	The size of the second item.

Return `SInt32` that indicates one of the following:

- `< 0` — object/data item 1 is less than object/data item 2
- `0` — object/data item 1 is equal to object/data item 2
- `> 0` — object/data item 1 is greater than object/data item 2

CompareToKey()

Purpose This method is not yet implemented.

Access Virtual, Public

Prototype `virtual SInt32 CompareToKey(
const void* inItem,
USInt32 inSize,
const void* inKey) const;`

Parameters This method has the following parameters:

<code>const void*</code>	<code>inItem</code>	The pointer to the first item.
<code>USInt32</code>	<code>inSize</code>	The size of the first item.
<code>const void*</code>	<code>inKey</code>	The key to compare with.

Return `SInt32`

GetComparator()

Purpose	This method is an accessor for sComparator . If sComparator does not yet exist, the object is created before returning from this method.
Access	Static, Public
Prototype	<code>static LComparator* GetComparator();</code>
Parameters	None
Return	A pointer to sComparator .

IsEqualTo()

Purpose	This method detects whether two objects are equal.	
Access	Virtual, Public	
Prototype	<pre>virtual Boolean IsEqualTo(const void* inItemOne, const void* inItemTwo, UInt32 inSizeOne, UInt32 inSizeTwo) const;</pre>	
Parameters	This method has the following parameters:	
	<code>const void*</code>	<code>inItemOne</code> A data pointer to the first item.
	<code>const void*</code>	<code>inItemTwo</code> A data pointer to the second item.
	<code>UInt32</code>	<code>inSizeOne</code> The size of the first item.
	<code>UInt32</code>	<code>inSizeTwo</code> The size of the second item.
Return	Boolean indicating true if the items are equal, false otherwise.	

IsEqualToKey()

Purpose	This method compares an item to the key to see if they are identical.	
Access	Virtual, Public	
Prototype	<pre>virtual Boolean IsEqualToKey(const void* inItem, UInt32 inSize, const void* inKey) const;</pre>	
Parameters	This method has the following parameters:	
	<pre>const void* inItem</pre>	The pointer to the first item.
	<pre>UInt32 inSize</pre>	The size of the first item.
	<pre>const void* inKey</pre>	The key to compare with.
Return	Boolean indicating true if the items are equal, false otherwise.	

sComparator

Purpose	This data member stores the pointer to the object we use as a standard for comparisons.
Access	Static, Protected
Prototype	<pre>static LComparator* sComparator;</pre>

LControl

Overview LControl is a PowerPlant class that is used for manipulation and control of radio buttons, check boxes, pop-up menus, and other visual interface controls.

A control is an object of any class which inherits from the LControl class. All the standard Mac OS controls are provided by PowerPlant, and some additional controls as well.

Methods The methods in this class are:

LControl()	~LControl()
BroadcastValueMessage()	ClickSelf()
FindHotSpot()	GetMaxValue()
GetMinValue()	GetValue()
GetValueMessage()	HotSpotAction()
HotSpotResult()	IncrementValue()
PointInHotSpot()	SetMaxValue()
SetMinValue()	SetValue()
SetValueMessage()	SimulateHotSpotClick()
TrackHotSpot()	

Data Members The data members in this class are:

mValueMessage	mValue
mMinValue	mMaxValue

Operation Controls are a large topic of discussion. In order to master the concepts and terminology of Controls within PowerPlant, a careful reading of the “Controls and Messaging” chapter in *The PowerPlant Book* is a good thing.

Source files (Pane Classes)

LControl.h

LControl.cp

LControl

- Ancestors
- [LAttachable](#)

[LBroadcaster](#)

[LPane](#)

LControl()

- Purpose
- The constructors create objects from the passed-in parameters.
- Access
- Public

Prototype

```
LControl();
LControl( const LControl &inOriginal );
LControl( const SPaneInfo &inPaneInfo,
MessageT inValueMessage,
SInt32 inValue,
SInt32 inMinValue,
SInt32 inMaxValue );
LControl( LStream *inStream );
```

- Parameters
- These constructors have the following parameters:

const LControl&	inOriginal	A reference to the LControl object you want to copy.
const SPaneInfo&	inPaneInfo	A reference to the SPaneInfo object that is the super view.
MessageT	inValueMessage	The message sent when the control is interacted with.
SInt32	inValue	The value of the control.
SInt32	inMinValue	The minimum control value.
SInt32	inMaxValue	The maximum control value.
LStream*	inStream	A pointer to a stream object that contains the information to create the LControl object.

~LControl()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LControl();</code>

BroadcastValueMessage()

Purpose	Broadcasts a message indicating a control change to the Listeners.
Access	Virtual, Protected
Prototype	<code>virtual void BroadcastValueMessage();</code>
Parameters	None
Return	None

ClickSelf()

Purpose	Tracks the mouse click and determines if a hot spot within a control was selected. If the hot spot was selected then this method responds to the mouse click by calling the HotSpotResult() method defined for the class.		
Access	Virtual, Protected		
Prototype	<code>virtual void ClickSelf(const SMouseDownEvent &inMouseDown);</code>		
Parameters	This method has the following parameters:		
	<code>const SMouseDownEvent&</code>	<code>inMouseDown</code>	The event information for the mouse down.
Return	None		

FindHotSpot()

Purpose	Determine which hot spot, if any, was clicked.			
Access	Virtual, Protected			
Prototype	<code>virtual SInt16 FindHotSpot(Point inPoint);</code>			
Parameters	<p>This method has the following parameters:</p> <table><tr><td>Point</td><td>inPoint</td><td>Local coordinates for the Control.</td></tr></table>	Point	inPoint	Local coordinates for the Control.
Point	inPoint	Local coordinates for the Control.		
Return	SInt16 containing the value of the hot spot selected.			
Remarks	If the return value is 0 then no hot spot was selected. Valid hot spots are integer values greater than 0.			

GetMaxValue()

Purpose	Get the maximum value for the control, found in the mMaxValue data member.
Access	Public
Prototype	<code>SInt32 GetMaxValue() const;</code>
Parameters	None
Return	SInt32 containing the value in mMaxValue .

GetMinValue()

Purpose	Get the minimum value for the control, found in the mMinValue data member.
Access	Public
Prototype	<code>SInt32 GetMinValue() const;</code>
Parameters	None

Return	SInt32 containing the value in mMinValue .
--------	--

GetValue()

Purpose	Return the current value of a Control.
Access	Virtual, Public
Prototype	<code>virtual SInt32 GetValue() const;</code>
Parameters	None
Return	SInt32 containing the value in mValue

GetValueMessage()

Purpose	Returns the value of the message.
Access	Public
Prototype	<code>MessageT GetValueMessage() const;</code>
Parameters	None
Return	MessageT containing the value in mValueMessage .

HotSpotAction()

Purpose	This method allows you to take action while the hot spot is down clicked by the mouse.
Access	Virtual, Protected
Prototype	<code>virtual void HotSpotAction(SInt16 inHotSpot, Boolean inCurrInside, Boolean inPrevInside);</code>
Parameters	This method has the following parameters:

LControl

	SInt16	inHotSpot	The hot spot number.
	Boolean	inCurrInside	Parameter is true if the mouse is currently inside the hot spot.
	Boolean	inPrevInside	Parameter is true if the mouse was previously in the hot spot.
Return	None		
Remarks	Depending on whether the mouse is inside or outside the hot spot you can override to change the control's visual appearance. While the mouse is down you can override to continuously perform some action.		

HotSpotResult()

Purpose	Perform an action as a result of clicking and releasing the mouse inside a hot spot.		
Access	Virtual, Protected		
Prototype	virtual void HotSpotResult(SInt16 inHotSpot);		
Parameters	This method has the following parameters: <div><div>SInt16</div><div>inHotSpot</div><div>The hotspot number.</div></div>		
Return	None		
Remarks	Subclasses need to override this function to implement behavior associated with clicking in a Control hot spot.		

IncrementValue()

Purpose	Increment the current value of the control.		
Access	Virtual, Public		
Prototype	<code>virtual void IncrementValue(SInt32 inIncrement);</code>		

Parameters	This method has the following parameters:		
	SInt32	inIncrement	The amount the current control value is incremented.
Return	None		
Remarks	The incremental change in the amount of the value can be a negative amount.		

PointInHotSpot()

Purpose	Determine if the mouse is in the hot spot while down-clicking the mouse.		
Access	Virtual, Protected		
Prototype	<code>virtual Boolean PointInHotSpot(Point inPoint, SInt16 inHotSpot);</code>		
Parameters	This method has the following parameters:		
	Point	inPoint	The local coordinates for the control.
	SInt16	inHotSpot	The hot spot number.
Return	Boolean true if the point is with the hot spot, else false.		

SetMaxValue()

Purpose	If the value is greater than mMaxValue then set to the maximum value.		
Access	Virtual, Public		
Prototype	<code>virtual void SetMaxValue(SInt32 inMaxValue);</code>		
Parameters	This method has the following parameters:		
	SInt32	inMaxValue	Maximum value to set for the control.
Return	None		

SetMinValue()

Purpose	If the value is less than mMinValue then set to the minimum value.	
Access	Virtual, Public	
Prototype	<code>virtual void SetMinValue(SInt32 inMinValue);</code>	
Parameters	This method has the following parameters:	
	SInt32 inMinValue	The minimum value to set for the control.
Return	None	

SetValue()

Purpose	If the value has changed then set the new value and inform Listeners of the value change.	
Access	Virtual, Public	
Prototype	<code>virtual void SetValue(SInt32 inValue);</code>	
Parameters	This method has the following parameters:	
	SInt32 inValue	The new value to set for the control.
Return	None	

SetValueMessage()

Purpose	Sets the value of the message.
Access	Public
Prototype	<code>void SetValueMessage(MessageT inValueMessage);</code>

Parameters	This method has the following parameters:		
	MessageT	inValueMessage	The value to set for the message.
Return	None		

SimulateHotSpotClick()

Purpose	This method allows you to simulate clicking the hot spot of a Control. Visual feedback is provided as if the user had clicked inside the hot spot of a Control. HotSpotAction() is called to simulate a down of the mouse. Then HotSpotResult() is used to simulate release within a particular hot spot.		
Access	Virtual, Public		
Prototype	virtual void SimulateHotSpotClick(SInt16 inHotSpot);		
Parameters	This method has the following parameters:		
	SInt16	inHotSpot	The hot spot number.
Return	None		
Remarks	This function can be used to implement keystroke equivalents for Control-Click operations.		

TrackHotSpot()

Purpose	After down clicking this function tracks the mouse and PointInHotSpot() determines if a point is within a particular hotspot. Movement inside and outside the hotspot is tracked. HotSpotAction() is called to take action during mouse tracking. Once the mouse is released the location of the event is recorded.
---------	--

LControl

Access	Virtual, Protected	
Prototype	<code>virtual Boolean TrackHotSpot(SInt16 inHotSpot, Point inPoint);</code>	
Parameters	This method has the following parameters:	
	SInt16	inHotSpot The hotspot number.
	Point	inPoint Local coordinates for the control.
Return	Boolean true if mouse released inside the hot spot, else false.	

mValueMessage

Purpose	This data member is set using SetValueMessage() . The message is broadcast to all Listeners via BroadcastValueMessage() .
Access	Protected
Prototype	<code>MessageT mValueMessage;</code>

mValue

Purpose	Contains the value for the control.
Access	Protected
Prototype	<code>SInt32 mValue;</code>

mMinValue

Purpose	Minimum value for the control.
Access	Protected
Prototype	<code>SInt32 mMinValue;</code>

mMaxValue

Purpose	Maximum value for the control.
Access	Protected
Prototype	<code>SInt32 mMaxValue;</code>

LDataArrived

Overview	LDataArrived is a PowerPlant class that assists in the task of sending a message when data arrives via an asynchronous channel.	
Methods	The methods in this class are: LDataArrived() GetDataBuffer() GetRemoteAddress() ~LDataArrived() GetDataSize()	
Data Members	The data members in this class are: mDataBuffer mRemoteAddress mDataSize mMustReleaseMemory	
Source files	(Networking Classes) UNetworkMessages.h UNetworkMessages.cp	

LDataArrived()

Purpose	The constructor creates the object.
Access	
Prototype	<pre>LDataArrived(MessageT inMessageType, ResultT inResultCode, void* inDataBuffer, UInt32 inDataSize, LInternetAddress* inRemoteAddress, Boolean inMustReleaseMemory, LEndpoint* inEndpoint); LDataArrived(); /* do not use */ LDataArrived(LDataArrived&); /* do not use */</pre>
Parameters	The parameters for this constructor are:

LDataArrived

MessageT	inMessageType	The message.
ResultT	inResultCode	The result code.
void*	inDataBuffer	The buffer.
UInt32	inDataSize	The size of the data.
LInternetAddress*	inRemoteAddress	The remote address.
Boolean	inMustReleaseMemory	Whether to release memory or not.
LEndpoint*	inEndpoint	The network endpoint.

~LDataArrived()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LDataArrived();</code>

GetDataBuffer()

Purpose	Retrieve the value of the mDataBuffer data member.
Access	Public
Prototype	<code>void* GetDataBuffer() const;</code>
Parameters	None
Return	Returns the value of mDataBuffer .

GetDataSize()

Purpose	Retrieve the size of the data, held in the mDataSize data member.
Access	Public
Prototype	<code>UInt32 GetDataSize() const;</code>
Parameters	None
Return	The value of mDataSize .

GetRemoteAddress()

Purpose	Retrieve the value of the mRemoteAddress data member.
Access	Public
Prototype	<code>LInternetAddress* GetRemoteAddress() const;</code>
Parameters	None
Return	The address, held in mRemoteAddress .

mDataBuffer

Purpose	The data buffer.
Access	Protected
Prototype	<code>void* mDataBuffer;</code>

mDataSize

Purpose	The size of the data.
---------	-----------------------

LDataArrived

Access	Protected
Prototype	UInt32 mDataSize;

mRemoteAddress

Purpose	The address.
Access	Protected
Prototype	LInternetAddress* mRemoteAddress;

mMustReleaseMemory

Purpose	Whether to release memory or not.
Access	Protected
Prototype	Boolean mMustReleaseMemory;

LDataStream

Description	<p>LDataStream is a stream whose bytes are maintained in a block of memory delimited by a pointer to the first byte and a byte count.</p> <p>This class supports reading and writing of bytes from or to the stream in arbitrary size blocks.</p>								
Methods	<p>The methods in this class are:</p> <table><tr><td>LDataStream()</td><td>~LDataStream()</td></tr><tr><td>GetBuffer()</td><td>SetBuffer()</td></tr><tr><td>GetBytes()</td><td>PutBytes()</td></tr><tr><td>operator =</td><td></td></tr></table>	LDataStream()	~LDataStream()	GetBuffer()	SetBuffer()	GetBytes()	PutBytes()	operator =	
LDataStream()	~LDataStream()								
GetBuffer()	SetBuffer()								
GetBytes()	PutBytes()								
operator =									
Data Members	<p>There one data members in this class:</p> <p>mBuffer</p>								
Source files	<p>(File & Stream Classes)</p> <p>LDataStream.h</p> <p>LDataStream.cp</p>								
See also	<p>LStream</p>								

LDataStream()

Purpose	<p>There are three constructor methods.</p> <p>The default constructor sets mBuffer to nil. The second constructor is this class' copy constructor.</p> <p>The third constructor creates the LDataStream object from a pointer and a byte count. inBuffer points to the first byte of the stream, which is inLength bytes long.</p>
Access	Public

LDataStream

Prototypes	<pre>LDataStream(); // Default Constructor LDataStream(const LDataStream& inOriginal); LDataStream(void* inBuffer, SInt32 inLength);</pre>
Return	No return value for a constructor.
Remarks	Just calls the inherited methods.

~LDataStream()

Purpose	The destructor closes the data and resource forks before deletion.
Access	Public, Virtual
Prototype	<pre>virtual ~LDataStream();</pre>
Parameters	None
Return	No return value for a destructor

GetBuffer()

Purpose	Return a pointer to the entire DataStream.
Access	Public, Virtual
Prototype	<pre>void* GetBuffer()</pre>
Parameters	None
Return	mBuffer.
Remarks	This method is actually found in LDataStream.h.

SetBuffer()

Purpose	Set the stream buffer (<code>mBuffer</code>) to <code>inBuffer</code> and the length of the stream to <code>inLength</code> .		
Access	Public, Virtual		
Prototype	<pre>void SetBuffer(void* inBuffer, SInt32 inLength);</pre>		
Parameters	<code>void*</code>	<code>inBuffer</code>	Pointer to the DataStream.
	<code>SInt32</code>	<code>inLength</code>	size of stream in bytes to write
Return	None		

GetBytes()

Purpose	Reads data from the DataStream to a buffer. Passes back the number of bytes actually read which may be less than the amount requested.		
Access	Public, Virtual		
Prototype	<pre>ExceptionCode GetBytes(void* outBuffer, SInt32& ioByteCount);</pre>		
Parameters	<code>void*</code>	<code>outBuffer</code>	Pointer to the data buffer
	<code>SInt32&</code>	<code>ioByteCount</code>	Number of bytes to read
Return	<code>readErr</code> if attempt to read past the end of the DataStream. Otherwise, returns <code>noErr</code> .		

PutBytes()

Purpose	Writes data from a buffer to the DataStream. Passes back the number of bytes actually written which may be less than the amount requested.								
Access	Public, Virtual								
Prototype	<pre>ExceptionCode PutBytes(const void* inBuffer, SInt32& ioByteCount);</pre>								
Parameters	<table><tr><td>void*</td><td>inBuffer</td><td>Pointer to the data buffer</td></tr><tr><td>SInt32&</td><td>ioByteCount</td><td>Number of bytes to write</td></tr></table>			void*	inBuffer	Pointer to the data buffer	SInt32&	ioByteCount	Number of bytes to write
void*	inBuffer	Pointer to the data buffer							
SInt32&	ioByteCount	Number of bytes to write							
Return	Write if attempt to read past the end of the DataStream. Otherwise, returns noErr.								

operator =

Purpose	Assignment operator. The DataStream will point to the same buffer as the original.		
Access	Public		
Prototype	LDataStream&operator = (const LDataStream&inOriginal);		
Parameters	const LDataStream&	inOriginal	Address of the original DataStream
Return	Address of the DataStream object.		
Remarks	operator = does not create a duplicate of the DataStream, but assigns a DataStream the same memory address as the original. If you delete one, you loose both. You have been warned.		

mBuffer

Purpose	Pointer to the data buffer.
Access	Protected
Prototype	<code>void* mBuffer</code>

LDefaultOutline

Overview	LDefaultOutline is a PowerPlant class that is used for drawing an outline around the default button in a dialog.
Methods	The methods in this class are: LDefaultOutline() DrawSelf()
Data Members	There are no data members in this class.
Operation	You typically won't create an object of this class yourself. PowerPlant makes one for you when you create a dialog box and specify a default button.
Source files	(Pane Classes) LLStdControl.h LLStdControl.cp
Ancestors	LAttachable LPane

LDefaultOutline()

Purpose	This method onstructs an outline for the specified host Pane.
Access	Public
Prototype	LDefaultOutline(LPane* inHostPane);
Parameters	The parameter for this constructor is:

LPane*	inHostPane	This is a pointer to the host pane object.
--------	------------	--

LDefaultOutline

DrawSelf()

Purpose This method draws the outline. This is an override of [DrawSelf\(\)](#) in [LPane](#).

LDialogBox

Overview	LDialogBox is a PowerPlant class that is used for creating dialog boxes in your program.	
Methods	The methods in this class are:	
	LDialogBox()	~LDialogBox()
	FinishCreateSelf()	HandleKeyPress()
	ListenToMessage()	SetCancelButton()
	SetDefaultButton()	
Data Members	The data members in this class are:	
	mDefaultButtonID	mCancelButtonID
	mDefaultOutline	
Operation	<p>LDialogBox is a simple extension of the LWindow class, with the addition of button tracking for the default and cancel buttons. The data members that identify the default and cancel buttons are mDefaultButtonID and mCancelButtonID. You can set the buttons in the Constructor resource editor, or you can set them at runtime. You specify the button by its Pane ID number.</p> <p>To learn more about using this class, refer to <i>The PowerPlant Book</i>.</p>	
Source files	(Pane Classes)	
	LDialogBox.h	
	LDialogBox.cp	
Ancestors	LAttachable	
	LCommander	
	LListener	
	LModelObject	
	LPane	

[LView](#)

[LWindow](#)

See Also

[LControl](#)

[LDefaultOutline](#)

LDialogBox()

Purpose The constructors create objects from the passed-in parameters.

Access Public

Prototype `LDialogBox();`
`LDialogBox(SWindowInfo &inWindowInfo);`
`LDialogBox(ResIDT inWINDid,`
`Uint32 inAttributes,`
`LCommander *inSuper);`
`LDialogBox(LStream *inStream);`

Parameters The parameters for this constructor are:

<code>SWindowInfo&</code>	<code>inWindowInfo</code>	The reference to the window information for the SuperView.
<code>ResIDT</code>	<code>inWINDid</code>	The accompanying 'WIND' resource ID for this dialog box.
<code>Uint32</code>	<code>inAttributes</code>	A value that specifies the attributes for the window.
<code>LCommander*</code>	<code>inSuper</code>	A pointer to a SuperCommander.
<code>LStream*</code>	<code>inStream</code>	A pointer to a stream object that contains the information to create the LDialogBox object.

~LDialogBox()

Purpose	The destructor destroys the object.
Access	Public
Prototype	<code>virtual ~LDialogBox();</code>

FinishCreateSelf()

Purpose	Finish creation by linking the dialog box as a Listener to its default and cancel buttons. This is an override of FinishCreateSelf() in LPane .
---------	---

HandleKeyPress()

Purpose	This method handles keyboard equivalents for hitting the default and cancel buttons of the dialog box. The default button is Enter or Return. The cancel button is Escape or Command-Period. This is an override of HandleKeyPress() in LCommander .
---------	--

ListenToMessage()

Purpose	This method respond to messages from Broadcasters. The dialog box responds to the <code>cmd_Close</code> message by deleting itself. This method is an override of ListenToMessage() in LListener .
Remarks	Negative message numbers are relayed to the <code>ProcessCommand()</code> . This allows subclasses of the dialog box to handle messages as commands. Or, if you don't want to subclass this dialog box, the <code>SuperCommander</code> of the dialog box will receive the messages as commands.

SetCancelButton()

Purpose	Using this method you specify the Pane ID of the cancel button of the dialog box. The cancel button must be derived from LControl . This object is then added as a Listener.			
Access	Virtual, Public			
Prototype	<code>virtual void SetCancelButton(PaneIDT inButtonID) ;</code>			
Parameters	<div>The parameter for this method is:<table><tr><td>PaneIDT</td><td>inButtonID</td><td>The pane ID of the cancel button.</td></tr></table></div>	PaneIDT	inButtonID	The pane ID of the cancel button.
PaneIDT	inButtonID	The pane ID of the cancel button.		
Return	None			
Remarks	Certain keyboard equivalents (Escape and Command-Period) simulate a click inside the cancel button.			

SetDefaultButton()

Purpose	Using this method you specify the Pane ID of the default button of the dialog box. The default button must be derived from LControl .			
Access	Virtual, Public			
Prototype	<code>virtual void SetDefaultButton(PaneIDT inButtonID) ;</code>			
Parameters	<div>The parameter for this method is:<table><tr><td>PaneIDT</td><td>inButtonID</td><td>The pane ID of the default button.</td></tr></table></div>	PaneIDT	inButtonID	The pane ID of the default button.
PaneIDT	inButtonID	The pane ID of the default button.		
Return	None			
Remarks	The default button has a default outline around it and certain keyboard equivalents (Return and Enter) simulate a click inside it.			

mDefaultButtonID

Purpose	This data member stores the pane ID of the default button.
Access	Protected
Prototype	PaneIDT mDefaultButtonID;

mCancelButtonID

Purpose	This data member stores the pane ID of the cancel button.
Access	Protected
Prototype	PaneIDT mCancelButtonID;

mDefaultOutline

Purpose	This data member stores the pointer to a LDefaultOutline .
Access	Protected
Prototype	LDefaultOutline *mDefaultOutline;

LDocApplication

Overview	<p>LDocApplication is a PowerPlant class that is used for applications that open and manipulate one or more documents. LDocApplication also handles the ae_OpenDoc and ae_PrintDoc Apple Events.</p>																			
Methods	<p>The methods in this class are:</p> <table><tr><td>LDocApplication()</td><td>~LDocApplication()</td></tr><tr><td>ChooseDocument()</td><td>CountSubModels()</td></tr><tr><td>DoAEOpenOrPrintDoc()</td><td>FindCommandStatus()</td></tr><tr><td>GetPositionOfSubModel()</td><td>GetSubModelByName()</td></tr><tr><td>GetSubModelByPosition()</td><td>HandleAppleEvent()</td></tr><tr><td>HandleCreateElementEvent()</td><td>MakeNewDocument()</td></tr><tr><td>ObeyCommand()</td><td>OpenDocument()</td></tr><tr><td>PrintDocument()</td><td>SendAECreateDocument()</td></tr><tr><td>SendAEOpenDoc()</td><td>SetupPage()</td></tr></table>		LDocApplication()	~LDocApplication()	ChooseDocument()	CountSubModels()	DoAEOpenOrPrintDoc()	FindCommandStatus()	GetPositionOfSubModel()	GetSubModelByName()	GetSubModelByPosition()	HandleAppleEvent()	HandleCreateElementEvent()	MakeNewDocument()	ObeyCommand()	OpenDocument()	PrintDocument()	SendAECreateDocument()	SendAEOpenDoc()	SetupPage()
LDocApplication()	~LDocApplication()																			
ChooseDocument()	CountSubModels()																			
DoAEOpenOrPrintDoc()	FindCommandStatus()																			
GetPositionOfSubModel()	GetSubModelByName()																			
GetSubModelByPosition()	HandleAppleEvent()																			
HandleCreateElementEvent()	MakeNewDocument()																			
ObeyCommand()	OpenDocument()																			
PrintDocument()	SendAECreateDocument()																			
SendAEOpenDoc()	SetupPage()																			
Data Members	<p>There are no data members for this class.</p>																			
Operation	<p>LDocApplication is almost an abstract class (some functions are virtual, designed to be overridden). You should always create a subclass rather than use this one.</p> <p>To learn more about handling documents, refer to <i>The PowerPlant Book</i>. Also be sure to examine the TextDocument demo on the CodeWarrior Reference CD.</p>																			
Source files	<p>(Commander Classes)</p> <p>LDocApplication.h</p> <p>LDocApplication.cp</p>																			
See also	<p>LApplication</p> <p>LDocument</p>																			

LDocApplication

LDocApplication()

Purpose	Default constructor. Override in derived class.
Access	Public
Prototype	<code>LDocApplication() ;</code>
Parameters	None
Return	No return value for a constructor

~LDocApplication()

Purpose	Virtual function, override in derived calss if appropriate.
Access	Public
Prototype	<code>~LDocApplication() ;</code>
Parameters	None
Return	No return value for a destructor

ChooseDocument()

Purpose	Virtual function, must be overridden by your derived class
Access	Public
Prototype	<code>virtual void ChooseDocument() ;</code>
Parameters	None
Return	None
Remarks	See the Text Document demo on the CodeWarrior Reference CD for a good example.

CountSubModels()

Purpose	Return the number of AppleEvent submodels of a particular type.		
Access	Public		
Prototype	<pre>virtual SInt32 CountSubModels(DescType inModelID) const;</pre>		
Parameters	DescType	inModelID	AppleEvent model ID
Return	Number of AE submodels.		
Remarks	If inModelID is 'cDocument', calls LApplication::CountSubModels() to return the number of 'cWindow' elements (in this case, a window = a document). Otherwise, calls the overridden member function <code>LApplication::CountSubModels()</code> with inModelID.		

DoAEOpenOrPrintDoc()

Purpose	Respond to an AppleEvent (usually from the Finder) to open or print a number of documents. Loops through the list to extract the descriptor and builds an FSSpec for each document, then open or print it as appropriate. Uses exceptions to catch OS errors.		
Access	Public		
Prototype	<pre>virtual void DoAEOpenOrPrintDoc(const AppleEvent& inAppleEvent, AppleEvent& /* outAEReply */, SInt32 inAENumber);</pre>		
Parameters	const AppleEvent&	inAppleEvent	AppleEvent

LDocApplication

	AppleEvent& SInt32	outAEReply inAENumber	ignored number of items to open or print
Return	None		
Remarks	aeOutReply is not used in this function and is commented out so that the code will compile without warning messages.		

FindCommandStatus()

Purpose	Process and return the status of a menu command. The inCommand parameter specifies the command. On return, outEnabled indicates whether the command is enabled, outUsesMark tells whether it is marked, outMark returns the mark character (if any), and outName returns the name of the command.
Access	Public
Prototype	<pre>virtual void FindCommandStatus(CommandT inCommand, Boolean& outEnabled, Boolean& outUsesMark, UInt16& outMark, Str255 outName);</pre>
Return	None
Remarks	Always enables cmd_New, cmd_Open, and cmd_PageSetup. For all other commands, calls the overridden LApplication::FindCommandStatus().

GetPositionOfSubModel()

Purpose	Virtual function returns the position (1 = first) of a SubModel within an Application.
Access	Public

Prototype	<code>Virtual SInt32 GetPositionOfSubModel(DescType inModelID, const LModelObject* inSubModel) const;</code>
Return	Always returns 1 for 'cDocument' model IDs, otherwise calls <code>LApplication::GetPositionOfSubModel()</code> .

GetSubModelByName()

Purpose	Pass back an AppleEvent token representing an AppleEvent submodel identified by its name.
Access	Public
Prototype	<code>virtual void GetSubModelByName(DescType inModelID, Str255 inName, AEDesc& outToken) const;</code>
Return	None
Remarks	If the <code>inModelID</code> is 'cDocument', calls <code>LDocument::FindNamedDocument()</code> to find the name, put it in a token. If it can't be found, throw an exception. All other ids get sent to <code>LApplication::GetSubModelByName()</code> .

GetSubModelByPosition()

Purpose	Pass back an AppleEvent token representing an AppleEvent submodel identified by its numerical position.
Access	Public
Prototype	<code>virtual void GetSubModelByPosition(DescType inModelID, SInt32 inPosition, AEDesc& outToken) const;</code>
Return	None

LDocApplication

Remarks	If <code>inModelID</code> is 'cDocument', throws an error exception for 'errAENoSuchObject', otherwise calls the overridden <code>LApplication::GetSubModelByPostion()</code> .
---------	---

HandleAppleEvent()

Purpose	Called when the application gets an Apple Event, usually a command to Open or Print a document in the Finder. Handles <code>ae_OpenDoc</code> and <code>ae_PrintDoc</code> . All other events get passed back to <code>LApplication</code> .
Access	Public
Prototype	<pre>virtual void HandleAppleEvent(const AppleEvent& inAppleEvent, AppleEvent& outAEReply, AEDesc& outResult, long inAENumber);</pre>
Return	None

HandleCreateElementEvent()

Purpose	Respond to an AppleEvent to create a new 'cDocument' or 'cWindow' element, often created in <code>LDocApplication::SendAECreatDocument()</code> . For other element classes, calls <code>LModelObject::HandleCreateElementEvent()</code> (via LApplication).
Access	Public
Prototype	<pre>virtual LModelObject* HandleCreateElementEvent(DescType inElemClass, DescType inInsertPosition, LModelObject* inTargetObject, const AppleEvent& inAppleEvent, AppleEvent& outAEReply);</pre>

MakeNewDocument()

Purpose Create a new Document and pass back an AppleEvent Model object representing that Document.

Access Public

Prototype `virtual LModelObject* MakeNewDocument();`

ObeyCommand()

Purpose Respond to commands

Access Public

Prototype `virtual Boolean ObeyCommand(
CommandTinCommand,
void*ioParam);`

OpenDocument()

Purpose Open a Document specified by an FSSpec. Subclass must override.

Access Public

Prototype `virtual void OpenDocument(
FSSpec* /* inMacFSSpec */);`

PrintDocument()

Purpose Print a Document specified by an FSSpec. Subclass must override.

Access Public

Prototype `virtual void PrintDocument()`

LDocApplication

```
FSSpec* /* inMacFSSpec */;
```

SendAECreatDocument()

Purpose	Self-send an AppleEvent to create a new Document (so that AppleScript can record the action).
Access	Public
Prototype	<code>virtual void SendAECreatDocument();</code>

SendAEOpenDoc()

Purpose	Self-send an AppleEvent to open a document (so that AppleScript can record the action).
Access	Public
Prototype	<code>virtual void SendAEOpenDoc(FSSpec&inFileSpec);</code>

SetupPage()

Purpose	Handle the Page Setup command.
Access	Public
Prototype	<code>virtual void SetupPage();</code>

LDocument

Overview	LDocument is a PowerPlant class that is used for opening, saving, closing, reverting, and printing a document. LDocument is an abstract class which provides a framework for supporting standard document operations via AppleEvents.	
Methods	The methods in this class are:	
	LDocument()	~LDocument()
	AskSaveAs()	AttemptClose()
	AttemptQuitSelf()	Close()
	DoAEClose()	DoAESave()
	DoPrint()	DoRevert()
	DoSave()	FindByFileSpec()
	FindCommandStatus()	FindNamedDocument()
	GetAEProperty()	GetDescriptor()
	GetDocumentList()	HandleAppleEvent()
	IsModified()	MakeCurrent()
	MakeSelfSpecifier()	ObeyCommand()
	SendAEClose()	SendAESaveAs()
	SetModified()	UsesFileSpec()
Data Members	The data members in this class are:	
	sDocumentList	mPrintRecordH
	mIsModified	mIsSpecified
Operation	You probably won't need to override all of these methods in a normal application, as many of them are complete.	
	Methods related to saving, reverting, and printing a document are empty, and need to be overridden.	
Source files	(Commander Classes)	

LDocument

LDocument.h

LDocument.cp

Ancestors [LAttachable](#)
 [LCommander](#)

LDocument()

Purpose	The constructors create objects from the passed-in parameters.		
Access	Public		
Prototype	<pre>LDocument () ; LDocument(LCommander *inSuper) ;</pre>		
Parameters	The parameter for these constructors is:		
	LCommander*	inSuper	The pointer to the Supercommander.

~LDocument()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<pre>virtual ~LDocument () ;</pre>

AskSaveAs()

Purpose	Ask the user to save a Document and give it a name.
Access	Virtual, Public

Prototype	<pre>virtual Boolean AskSaveAs (FSSpec &outFSSpec, Boolean inRecordIt);</pre>		
Parameters	The parameters for this method are:		
	FSSpec&	outFSSpec	The reference to the file we are working with.
	Boolean	inRecordIt	Unused.
Return	Returns false if the user cancels the operation, else it returns true.		

AttemptClose()

Purpose	Attempt to close a document. The Document might not close if it is modified and the user cancels the operation when asked whether to save the changes.		
Access	Virtual, Public		
Prototype	<pre>virtual void AttemptClose(Boolean inRecordIt);</pre>		
Parameters	The parameter for this method is:		
	Boolean	inRecordIt	Unused.
Return	None		

AttemptQuitSelf()

Purpose	Try to close a Document when quitting the program.		
Access	Virtual, Public		
Prototype	<pre>virtual Boolean AttemptQuitSelf(SInt32 inSaveOption);</pre>		

LDocument

Parameters	The parameter for this method is:			
	<table><tr><td>SInt32</td><td>inSaveOption</td><td>The save options for the document.</td></tr></table>	SInt32	inSaveOption	The save options for the document.
SInt32	inSaveOption	The save options for the document.		
Return	Return <code>false</code> if the user cancels when asked whether to save changes to a modified Document.			

Close()

Purpose	Close a document.
Access	Public
Prototype	<code>virtual void Close();</code>
Parameters	None
Return	None

DoAEClose()

Purpose	Close a document in response to a close AppleEvent.		
Access	Virtual, Public		
Prototype	<pre>virtual void DoAEClose(const AppleEvent &inCloseAE);</pre>		
Parameters	The parameter for this method is:		
	const AppleEvent&	inCloseAE	The reference to the AppleEvent received.
Return	None		

DoAESave()

Purpose	Save a Document in a particular file. This is equivalent to a "Save As" operation. Subclasses should override this function to save the file, and should set mIsSpecified to true.							
Access	Virtual, Public							
Prototype	<pre>virtual void DoAESave(FSSpec& inFileSpec, OSType inFileType);</pre>							
Parameters	The parameters for this method are: <table><tr><td>FSSpec&</td><td>inFileSpec</td><td>The reference to the file we are working with.</td></tr><tr><td>OSType</td><td>inFileType</td><td>The resource type for the file.</td></tr></table>		FSSpec&	inFileSpec	The reference to the file we are working with.	OSType	inFileType	The resource type for the file.
FSSpec&	inFileSpec	The reference to the file we are working with.						
OSType	inFileType	The resource type for the file.						
Return	None							

DoPrint()

Purpose	Print a document.
Access	Virtual, Public
Prototype	<pre>virtual void DoPrint();</pre>
Parameters	None
Return	None

DoRevert()

Purpose	Revert a document to its last saved version.
---------	--

LDocument

Access	Virtual, Public
Prototype	<code>virtual void DoRevert();</code>
Parameters	None
Return	None

DoSave()

Purpose	Save a Document, which must already be specified. Subclasses should override this method to save a document to an existing file.
Access	Virtual, Public
Prototype	<code>virtual void DoSave();</code>
Parameters	None
Return	None

FindByFileSpec()

Purpose	Return the document that uses the specified FSSpec .		
Access	Static, Public		
Prototype	<code>static LDocument* FindByFileSpec(const FSSpec &inFileSpec);</code>		
Parameters	The parameters for this method are:		
	<code>const FSSpec&</code>	<code>inFileSpec</code>	The reference to the file we are working with.
Return	A pointer to the LDocument that corresponds to the file we're interesting in.		

FindCommandStatus()

Purpose Return whether a command is enabled and/or marked in a Menu. This method is an override of [FindCommandStatus\(\)](#) in [LCommander](#).

FindNamedDocument()

Purpose Return the document with the specified name.

Access Static, Public

Prototype `static LDocument* FindNamedDocument (ConstStringPtr inName);`

Parameters The parameter for this method is:

ConstStringPtr	inName	The string pointer for the name to be found.
----------------	--------	--

Return A pointer to the LDocument we are interested in, or nil if not found.

GetAEProperty()

Purpose This method returns an AppleEvent property.

Access Virtual, Public

Prototype `virtual void GetAEProperty(DescType inProperty, const AEDesc &inRequestedType, AEDesc &outPropertyDesc) const;`

Parameters The parameters for this method are:

	DescType	inProperty	The descriptor type for the property.
	const AEDesc&	inRequestedType	The AppleEvent descriptor type.
	AEDesc&	outPropertyDesc	The property descriptor.

Return	None
--------	------

GetDescriptor()

Purpose	This method is a pure virtual method that must be implemented in your subclass. It returns the AppleEvent descriptor.		
Access	Pure Virtual, Public		
Prototype	<pre>virtual StringPtr GetDescriptor(Str255 outDescriptor) const = 0;</pre>		
Parameters	The parameter for this method is:		
	Str255	outDescriptor	The string.

Return	A StringPtr to the descriptor string.
--------	---------------------------------------

GetDocumentList()

Purpose	Returns the value of the sDocumentList data member.
Access	Static, Public
Prototype	<pre>static TArray<LDocument*>& GetDocumentList();</pre>
Parameters	None
Return	Returns the value of the sDocumentList .

HandleAppleEvent()

Purpose	This method handles an AppleEvent.
Access	Virtual, Public
Prototype	<pre>virtual void HandleAppleEvent(const AppleEvent &inAppleEvent, AppleEvent &outAEReply, AEDesc &outResult, long inAENumber);</pre>
Parameters	The parameters for this method are:

const AppleEvent&	inAppleEvent	The reference to the AppleEvent.
AppleEvent&	outAEReply	The AppleEvent reply.
AEDesc&	outResult	The AppleEvent result.
long	inAENumber	The AppleEvent number.

Return	None
--------	------

IsModified()

Purpose	Return whether a document has been modified since it was last saved.
Access	Virtual, Public
Prototype	<pre>virtual Boolean IsModified();</pre>
Parameters	None
Return	None

MakeCurrent()

Purpose	Make this Document the current one. You should override this method to do what makes sense for a particular kind of document. In most cases, this means selecting the main window for a document.
Access	Virtual, Public
Prototype	<code>virtual void MakeCurrent();</code>
Parameters	None
Return	None

MakeSelfSpecifier()

Purpose	AppleEvent Object Model Support	
Access	Virtual, Public	
Prototype	<code>virtual void MakeSelfSpecifier(AEDesc &inSuperSpecifier, AEDesc &outSelfSpecifier) const;</code>	
Parameters	The parameters for this method are:	
	AEDesc& inSuperSpecifier	The AppleEvent super specifier.
	AEDesc& outSelfSpecifier	The AppleEvent self specifier.
Return	None	

ObeyCommand()

Purpose	Respond to commands.
---------	----------------------

Access	Virtual, Public	
Prototype	<pre>virtual Boolean ObeyCommand(CommandT inCommand, void *ioParam);</pre>	
Parameters	The parameters for this method are:	
	Command T	inCommand Command passed to this method.
	void*	ioParam The data block accompanying the command.
Return	Return true if the command was handled, else return false.	

SendAEClose()

Purpose	Sends the close AppleEvent.	
Access	Virtual, Public	
Prototype	<pre>virtual void SendAEClose(SInt32 inSaveOption, FSSpec &inFileSpec, Boolean inExecute);</pre>	
Parameters	The parameters for this method are:	
	SInt32	inSaveOption The save options.
	FSSpec&	inFileSpec The reference to the file we are working with.
	Boolean	inExecute The value passed to SendAppleEvent().
Return	None	

SendAESaveAs()

Purpose	Sends the Save As AppleEvent.		
Access	Virtual, Public		
Prototype	<pre>virtual void SendAESaveAs (FSSpec &inFileSpec, OSType inFileType, Boolean inExecute);</pre>		
Parameters	The parameters for this method are:		
	OSType	inFileType	The resource type for the file.
	FSSpec&	inFileSpec	The reference to the file we are working with.
	Boolean	inExecute	The value passed to SendAppleEvent().
Return	None		

SetModified()

Purpose	Specify whether a document has been modified since it was last saved.		
Access	Virtual, Public		
Prototype	<pre>virtual void SetModified(Boolean inModified);</pre>		
Parameters	The parameter for this method is:		
	Boolean	inModified	The value to set in mIsModified .
Return	None		

UsesFileSpec()

Purpose	Return whether the Document has a file that uses the specified FSSpec.	
Access	Virtual, Public	
Prototype	<code>virtual Boolean UsesFileSpec(const FSSpec& inFileSpec) const;</code>	
Parameters	The parameter for this method is:	
	<code>const FSSpec&</code>	<code>inFileSpec</code> The reference to the file we are working with.
Return	Return <code>true</code> if the document has a file with the FSSpec, else return <code>false</code> .	

sDocumentList

Purpose	The static document list of pointers to LDocument objects.
Access	Static, Protected
Prototype	<code>static TArray<LDocument*> sDocumentList;</code>

mPrintRecordH

Purpose	The handle to the print record.
Access	Protected
Prototype	<code>THPrint mPrintRecordH;</code>

LDocument

mIsModified

Purpose	Indicates whether a document is modified (dirty) or not.
Access	Protected
Prototype	Boolean mIsModified;

mIsSpecified

Purpose	Indicates whether a document has the specified information to save without interacting with the user first.
Access	Protected
Prototype	Boolean mIsSpecified;

LDragAndDrop

Overview	LDragAndDrop is a PowerPlant mix-in class that is used for adding drag and drop features to a Pane.
Methods	<p>The methods in this class are:</p> <div>LDragAndDrop()FocusDropArea()</div> <div>HiliteDropArea()PointInDropArea()</div>
Data Members	<p>The data members in this class are:</p> <div>mPane</div>
Operation	<p>LDragAndDrop inherits from LDropArea and is a concrete implementation of that class. A pane that supports drag and drop should inherit from LDragAndDrop, not LDropArea.</p> <p>For more information on drag and drop using PowerPlant, refer to <i>PowerPlant Advanced Topics</i>.</p>
Source files	<p>(Feature Classes)</p> <div>LDragAndDrop.h</div> <div>LDragAndDrop.cp</div>
Ancestors	LDropArea

LDragAndDrop()

Purpose	This creates a drag and drop object from the passed-in parameters.
Access	Public
Prototype	<pre>LDragAndDrop(WindowPtr inMacWindow, LPane *inPane);</pre>
Parameters	The parameters for this constructor are:

LDragAndDrop

WindowPtr	inMacWindow	Toolbox window containing the drag and drop. It may be nil if the drag and drop is not in a window (e.g., if printing the drag and drop).
LPane*	inPane	Pane associated with the drag and drop. The drop area of the drag and drop is the Frame of the Pane.

FocusDropArea()

Purpose	Set up a local coordinate system and clipping region for a drop area.
Access	Virtual, Public
Prototype	<code>virtual void FocusDropArea();</code>
Parameters	None
Return	None

HiliteDropArea()

Purpose	Hilite a drop area to indicate that it can accept the current drag. For a drag and drop, the drop area is the Frame of its associated Pane inset by one pixel to account for the border which usually surrounds a drop-capable Pane.
Access	Virtual, Protected
Prototype	<code>virtual void HiliteDropArea(DragReference inDragRef);</code>
Parameters	The parameter for this method is:

	DragReference	inDragRef	A reference to a drop area to hilite.
Return	None		

PointInDropArea()

Purpose	Check whether a Point, in global coordinates, is inside a drop area.		
Access	Virtual, Public		
Prototype	<code>virtual Boolean PointInDropArea(Point inPoint);</code>		
Parameters	The parameter for this method is:		
	Point	inPoint	A point to check.
Return	Boolean true if the point is within the drop area, else false.		

mPane

Purpose	This data member stores the Pane that we're operating with.
Access	Protected
Prototype	<code>LPane *mPane;</code>

LDragTask

Description	LDragTask is a PowerPlant class that is used for encapsulating a single drag action initiated using the Mac OS Drag Manager.	
Methods	The methods in this class are:	
	LDragTask()	~LDragTask()
	AddFlavors()	AddRectDragItem()
	DoDrag()	DropLocationIsFinderTrash()
	GetDragReference()	GetDragRegion()
	InitDragTask()	MakeDragRegion()
Data Members	The data members in this class are:	
	mDragRef	mDragRegion
	mEventRecord	
Operation	Normally, you will use this class in conjunction with the LDragAndDrop class. To learn more information about using this class, refer to <i>PowerPlant Advanced Topics</i> .	
Source files	(Feature Classes)	
	<code>LDragTask.h</code>	
	<code>LDragTask.cp</code>	
See also	LDragAndDrop	

LDragTask()

Purpose	The constructors create objects from the passed-in parameters.
Access	Public
Prototypes	<pre>LDragTask(const EventRecord &inEventRecord); // This constructor is for dragging a single, rectangular item</pre>

LDragTask

```
LDragTask( const EventRecord &inEventRecord,  
const Rect &inItemRect,  
ItemReference inItemRef,  
FlavorType inFlavor,  
void *inDataPtr,  
Size inDataSize,  
FlavorFlags inFlags );  
  
// This constructor is for dragging an item outlined by a region  
LDragTask( const EventRecord &inEventRecord,  
RgnHandle inItemRgn,  
ItemReference inItemRef,  
FlavorType inFlavor,  
void *inDataPtr,  
Size inDataSize,  
FlavorFlags inFlags );
```

Parameters	These constructors have the following parameters:	
const EventRecord&	inEventRecord	A reference to the Mac OS Toolbox mouse down EventRecord.
const Rect&	inItemRect	The Rect coordinates of the drag.
ItemReference	inItemRef	A reference to the item that is in the drag.
FlavorType	inFlavor	This is the “flavor,” or data type for the drag.
void*	inDataPtr	This is a pointer to the data block for the drag.
Size	inDataSize	This is the size of the data block.
FlavorFlags	inFlags	This is a value for flavor flags, defined in the Drag.h Mac OS system header.
RgnHandle	inItemRgn	This must be in global coordinates, and will not be disposed of by this class.
Remarks	The inItemRect is in the local coordinates of the current port, so you may need to call SetPort (or FocusDraw) beforehand.	

~LDragTask()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LDragTask();</code>

AddFlavors()

Purpose	Add flavored items to the DragTask. If you use the short form of the LDragTask() constructor (EventRecord only), you must override this function to add items to the drag task.		
Access	Virtual, Protected		
Prototype	<code>void AddFlavors(DragReference inDragRef);</code>		
Parameters	The parameter for this method is:		
	DragReference	inDragRef	A reference to an item to add to the drag.
Return	None		

AddRectDragItem()

Purpose	This is a utility method for adding a rectangular item to the drag.		
Access	Virtual, Protected		
Prototype	<code>void AddRectDragItem(ItemReference inItemRef, const Rect &inItemRect);</code>		
Parameters	This method has the following parameters:		

LDragTask

	ItemReference	inItemRef	A reference to the item to add to the drag.
	const Rect&	inItemRect	The Rect must be in Global coordinates.
Return	None		

DoDrag()

Purpose	This method performs a drag task. This function calls other member functions to add items to the drag and build the drag region, then calls the Drag Manager to track the drag. You will not normally override this function.
Access	Virtual, Public
Prototype	<code>OSErr DoDrag();</code>
Parameters	None
Return	This method returns an <code>OSErr</code> error code if an error occurs executing the drag. If no error occurs, the return value will be <code>noErr</code> .

DropLocationIsFinderTrash()

Purpose	Return whether the drop location of the Drag is the Finder's Trash can. Normally, you'll call this routine after the Drag completes to determine whether to delete the dragged items.
Access	Public
Prototype	<code>Boolean DropLocationIsFinderTrash();</code>
Parameters	None
Return	A <code>Boolean</code> indicating whether the drop location of the drag is the Finder's trash can.

Remarks	Normally, you'll call this routine after the drag completes to determine whether to delete the dragged items.
---------	---

GetDragReference()

Purpose	This method returns the value of mDragRef .
Access	Public, Inline
Prototype	<code>DragReference GetDragReference();</code>
Parameters	None
Return	A <code>DragReference</code> containing the value of mDragRef .

GetDragRegion()

Purpose	This method returns the value of mDragRegion .
Access	Public, Inline
Prototype	<code>RgnHandle GetDragRegion();</code>
Parameters	None
Return	A <code>RgnHandle</code> containing the value of mDragRegion .

InitDragTask()

Purpose	This method is a private initializer for the class.
Access	Private
Prototype	<code>void InitDragTask();</code>
Parameters	None
Return	None

LDragTask

MakeDragRegion()

Purpose	Build the region outlining the items to be dragged.		
Access	Virtual, Protected		
Prototype	<pre>void MakeDragRegion(DragReference inDragRef, RgnHandle inDragRegion);</pre>		
Parameters	This method has the following parameters:		
	DragReference	inDragRef	A reference to the item in the drag.
	RgnHandle	inDragRegion	The drag region for the item.
Return	None		
Remarks	If you use the short form of the LDragTask() constructor (EventRecord only), you must override this function to specify the drag region for each item in the drag task.		

mDragRef

Purpose	This data member is the drag reference.
Access	Protected
Prototype	<pre>DragReference mDragRef;</pre>

mDragRegion

Purpose	This data member is the drag region.
Access	Protected
Prototype	<pre>RgnHandle mDragRegion;</pre>

mEventRecord

Purpose	This data member is the event record, containing information about the event that just occurred.
Access	Protected
Prototype	<code>const EventRecord &mEventRecord;</code>

LDropArea

Description LDropArea is a PowerPlant abstract base class that is used for providing the interface to implement tracking and receiving functionality.

Methods The methods in this class are:

<u>LDropArea()</u>	<u>~LDropArea()</u>
<u>AddDropArea()</u>	<u>DoDragDrawing()</u>
<u>DoDragInput()</u>	<u>DoDragReceive()</u>
<u>DoDragSendData()</u>	<u>DragAndDropIsPresent()</u>
<u>DragIsAcceptable()</u>	<u>EnterDropArea()</u>
<u>FindDropArea()</u>	<u>FocusDropArea()</u>
<u>HandleDragDrawing()</u>	<u>HandleDragInput()</u>
<u>HandleDragReceive()</u>	<u>HandleDragSendData()</u>
<u>HandleDragTracking()</u>	<u>HiliteDropArea()</u>
<u>InTrackingWindow()</u>	<u>InsideDropArea()</u>
<u>InstallHandlers()</u>	<u>ItemIsAcceptable()</u>
<u>LeaveDropArea()</u>	<u>PointInDropArea()</u>
<u>ReceiveDragItem()</u>	<u>RemoveDropArea()</u>
<u>UnhiliteDropArea()</u>	

Data Members The data members in this class are:

<u>mDragWindow</u>	<u>mCanAcceptCurrentDrag</u>
<u>mIsHilited</u>	<u>sDragTrackingProc</u>
<u>sDragReceiveProc</u>	<u>sDropAreaList</u>
<u>sCurrentDropArea</u>	<u>sDragHasLeftSender</u>

LDropArea

Operation	LDropArea uses the LArray class. Most of the data members of LDropArea are used internally by PowerPlant, and you won't need to use them.
Source files	(Feature Classes) LDragTask.h LDragTask.cp
See also	LArray LDragAndDrop LDragTask

LDropArea()

Purpose	The constructor creates an object from the passed-in parameters. It initializes drag and drop if needed.	
Access	Public	
Prototype	LDropArea (WindowPtr inWindow);	
Parameters	This constructor has the following parameter:	
	WindowPtr	inWindow The window in which to establish a drop area.

~LDropArea()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	virtual ~LDropArea();

AddDropArea()

Purpose	Add a drop area to the list maintained by this class.		
Access	Protected, Static		
Prototype	<pre>void AddDropArea(LDropArea *inDropArea, WindowPtr inMacWindow);</pre>		
Parameters	This method has the following parameter:		
	LDropArea*	inDropArea	A pointer to the drop area object to add to the list.
	WindowPtr	inMacWindow	The window pointer for the drop area to be added.
Return	None		

DoDragDrawing()

Purpose	Draw the items for a drag in progress. This method gets called if you installed the optional DragDrawingProc for this drop area.		
Access	Virtual, Protected		
Prototype	<pre>virtual void DoDragDrawing(DragRegionMessage inMessage, RgnHandle inShowRgn, Point inShowOrigin, RgnHandle inHideRgn, Point inHideOrigin, DragReference inDragRef);</pre>		
Parameters	This method has the following parameter:		

LDropArea

	DragRegionMessage	inMessage	The message passed to this object from the framework.
	RgnHandle	inShowRgn	The show region.
	Point	inShowOrigin	The origin point.
	RgnHandle	inHideRgn	The hide region.
	Point	inHideOrigin	The hide origin point.
	DragReference	inDragRef	The reference to the area.
Return	None		

DoDragInput()

Purpose	Modify the state of the mouse and modifier keys during a drag. This method gets called if you installed the optional DragInputProc for this drop area.		
Access	Virtual, Protected		
Prototype	<pre>virtual void DoDragInput(Point ioMouse, SInt16 ioModifiers, DragReference inDragRef);</pre>		
Parameters	This method has the following parameter:		
	Point	ioMouse	The point to check against.
	SInt16	ioModifiers	The modifier keys, if any.
	DragReference	inDragRef	The reference to the area.
Return	None		

DoDragReceive()

Purpose	Receive the items from a completed drag and drop. This method gets called after items are dropped into a drop area. The drag contains items that are acceptable, as defined by the <code>AcceptableTypes</code> and <code>AcceptableTypesEx</code> member functions. This method repeatedly calls ReceiveDragItem() for each item in the drag. You should override this method if you want to process the dragged items all at once.			
Access	Virtual, Protected			
Prototype	<pre>virtual void DoDragReceive(DragReference inDragRef);</pre>			
Parameters	<p>This method has the following parameter:</p> <table><tr><td>DragReference</td><td>inDragRef</td><td>The reference to the area.</td></tr></table>	DragReference	inDragRef	The reference to the area.
DragReference	inDragRef	The reference to the area.		
Return	None			

DoDragSendData()

Purpose	Send the data associated with a particular drag item. This method gets called if you installed the optional <code>DragSendDataProc</code> for this drop area, in which case you should override this function to provide the requested data by calling <code>SetDragItemFlavorData()</code> .
Access	Virtual, Protected
Prototype	<pre>virtual void DoDragSendData(FlavorType inFlavor, ItemReference inItemRef, DragReference inDragRef);</pre>
Parameters	This method has the following parameters:

LDropArea

	FlavorType	inFlavor	This is the “flavor,” or data type for the drag.
	ItemReference	inItemRef	A reference to the item that is in the drag.
	DragReference	inDragRef	The reference to the area.
Return	None		

DragAndDropIsPresent()

Purpose	Indicates whether drag and drop is implemented on the running Mac OS system.
Access	Static, Public
Prototype	<code>static Boolean DragAndDropIsPresent();</code>
Parameters	None
Return	Boolean of true if drag and drop is installed, else false.

DragIsAcceptable()

Purpose	Indicate whether a drop area can accept the specified drag. A drag is acceptable if all items in the drag are acceptable.		
Access	Virtual, Protected		
Prototype	<code>virtual Boolean DragIsAcceptable(DragReference inDragRef);</code>		
Parameters	This method has the following parameter:		
	DragReference	inDragRef	The reference to the area.
Return	Boolean of <code>true</code> if the drop area can accept the drag, else <code>false</code> .		

EnterDropArea()

Purpose	This method is called when a drag is entering a drop area. This call will be followed by a corresponding LeaveDropArea() call (when the drag moves out of the drop area or after the drag is received by this drop area). If the drop area can accept the drag and the drag is coming from outside the drop area, hilite the drop area.		
Access	Virtual, Protected		
Prototype	<pre>virtual void EnterDropArea(DragReference inDragRef, Boolean inDragHasLeftSender);</pre>		
Parameters	This method has the following parameters:		
	DragReference	inDragRef	The reference to the area.
	Boolean	inDragHasLeftSender	
Return	None		

FindDropArea()

Purpose	Return the drop area in the specified window containing the specified point which can receive the drag.		
Access	Static, Protected		
Prototype	<pre>static LDropArea* FindDropArea(WindowPtr inMacWindow, Point inGlobalPt, DragReference inDragRef);</pre>		
Parameters	This method has the following parameters:		

LDropArea

	WindowPtr	inMacWindow	The window to search in.
	Point	inGlobalPt	The point (in global coordinates).
	DragReference	inDragRef	The reference to the area.
Return	Returns <code>nil</code> if no drop area meets the requirements, else returns a pointer to the drop area.		

FocusDropArea()

Purpose	Set up local coordinate system and clipping region for a drop area. The Drag Manager sets the port to the Window containing the drop rectangle. However, it doesn't know anything about the local coordinates and clipping region.
Access	Virtual, Public
Prototype	<code>void FocusDropArea();</code>
Parameters	None
Return	None

HandleDragDrawing()

Purpose	Drag Manager callback for drawing the items during a drag.
Access	Static, Protected
Prototype	<code>static pascal OSErr HandleDragDrawing(DragRegionMessage inMessage, RgnHandle inShowRgn, Point inShowOrigin, RgnHandle inHideRgn, Point inHideOrigin,</code>

```
void *inRefCon,  
DragReference inDragRef);
```

Parameters This method has the following parameters:

DragRegionMess age	inMessage	The message passed to this method from the framework.
RgnHandle	inShowRgn	The show region.
Point	inShowOrigin	The point in the show region.
RgnHandle	inHideRgn	The hide region.
Point	inHideOrigin	The point in the hide region.
void*	inRefCon	A pointer to the Mac OS refcon.
DragReference	inDragRef	The reference to the area.

Return Returns an OSErr value, which will be noErr if everything
succeeds.

HandleDragInput()

Purpose This is a Drag Manager callback for manipulating the mouse and
modifier keys during a drag. This is basically an exception-handling
wrapper for calling [DoDragInput\(\)](#).

Access Static, Protected

Prototype static pascal OSErr HandleDragInput (
Point *ioMouse,
SInt16 *ioModifiers,
void *inRefCon,
DragReference inDragRef);

Parameters This method has the following parameters:

LDropArea

	Point*	ioMouse	The pointer to a point structure.
	SInt16*	ioModifiers	The show region.
	void*	inRefCon	The point in the show region.
	DragReference	inDragRef	The reference to the area.
Return	Returns an OSErr value, which will be noErr if everything succeeds.		

HandleDragReceive()

Purpose	This is a Drag Manager callback for receiving a successful drop.		
Access	Static, Protected		
Prototype	static pascal OSErr HandleDragReceive(WindowPtr inMacWindow, void* inRefCon, DragReference inDragRef);		
Parameters	This method has the following parameters:		
	WindowPtr	inMacWindow	The pointer to the window receiving the drag.
	void*	inRefCon	The Mac OS refcon pointer.
	DragReference	inDragRef	The reference to the area.
Return	Returns an OSErr value, which will be noErr if everything succeeds.		

HandleDragSendData()

Purpose	This is a Drag Manager callback for sending the data for an item that is part of an accepted drag and drop.		
Access	Static, Protected		
Prototype	<pre>static pascal OSErr HandleDragSendData(FlavorType inFlavor, void *inRefCon, ItemReference inItemRef, DragReference inDragRef);</pre>		
Parameters	This method has the following parameters:		
	FlavorType	inFlavor	The flavor type of the drag and drop (enumerated in Drag.h).
	void*	inRefCon	The Mac OS refcon pointer.
	ItemReference	inItemRef	The reference to the item that was drag and dropped.
	DragReference	inDragRef	The reference to the area.
Return	Returns an OSErr value, which will be noErr if everything succeeds.		

HandleDragTracking()

Purpose	This is the Drag Manager callback for tracking a drag in progress.		
Access	Static, Protected		
Prototype	<pre>static pascal OSErr HandleDragTracking(DragTrackingMessage inMessage, WindowPtr inMacWindow, void* inRefCon, DragReference inDragRef);</pre>		

LDropArea

Parameters	This method has the following parameters:		
	DragTrackingMessage	inMessage	The message that is passed to this method.
	WindowPtr	inMacWindow	The window pointer.
	void*	inRefCon	The Mac OS refcon pointer.
	DragReference	inDragRef	The reference to the area.
Return	Returns an OSErr value, which will be noErr if everything succeeds.		

HiliteDropArea()

Purpose	Hilite a drop area to indicate that it can accept the current drag. Any subclasses should override this function to create a region representing the drop area, and call ShowDragHilite().		
Access	Virtual, Protected		
Prototype	virtual void HiliteDropArea(DragReference inDragRef);		
Parameters	This method has the following parameter:		
	DragReference	inDragRef	The reference to the area.
Return	None		

InTrackingWindow()

Purpose	Track a Drag while it is inside a Window.
Access	Static, Protected

Prototype `static void InTrackingWindow(
WindowPtr inMacWindow,
DragReference inDragRef);`

Parameters This method has the following parameters:

WindowPtr	inMacWindow	The window pointer.
DragReference	inDragRef	The reference to the area.

Return None

InsideDropArea()

Purpose Track a Drag while it is inside a drop area. This method is called repeatedly while an acceptable drag is inside a drop area. Any subclasses may override to provide additional visual feedback during a drag, such as indicating an insertion point

Access Virtual, Protected

Prototype `virtual void InsideDropArea(
DragReference inDragRef);`

Parameters This method has the following parameter:

DragReference	inDragRef	The reference to the area.
---------------	-----------	----------------------------

Return None

InstallHandlers()

Purpose Install Tracking and Receive Handlers for the Drag Manager. We use a single Tracking handler and a single Receive handler for all drag and drop operations. With the Drag Manager, you can register handlers globally or for specific windows. With PowerPlant, we want to have separate handlers for each Pane that supports drag

LDropArea

and drop. Therefore, this class maintains a list of all drag and drop Panes and dispatches calls to the proper Panes.

Access	Static, Protected
Prototype	<code>static void InstallHandlers();</code>
Parameters	None
Return	None

ItemIsAcceptable()

Purpose	This method checks for each acceptable flavor, exiting immediately upon finding an acceptable flavor.		
Access	Virtual, Protected		
Prototype	<code>virtual Boolean ItemIsAcceptable(DragReference inDragRef, ItemReference inItemRef);</code>		
Parameters	This method has the following parameters:		
	DragReference	inDragRef	The reference to the area.
	ItemReference	inItemRef	The reference to the drag item.
Return	Boolean returning true if the item is acceptable, false if it is not.		

LeaveDropArea()

Purpose	This method is called when a drag is leaving a drop area. This call will have been preceded by a corresponding EnterDropArea() call.		
Access	Virtual, Protected		
Prototype	<code>virtual void LeaveDropArea(DragReference inDragRef);</code>		

Parameters	This method has the following parameter:		
	DragReference	inDragRef	The reference to the area.
Return	None		
Remarks	Remove hiliting of the drop area if necessary.		

PointInDropArea()

Purpose	This method takes a point and indicates whether the point lies within the drop area or not.			
Access	Pure Virtual, Public			
Prototype	<code>virtual Boolean PointInDropArea(Point inGlobalPt) = 0;</code>			
Parameters	<div>This method has the following parameter:</div> <table><tr><td>Point</td><td>inGlobalPt</td><td>The point to check.</td></tr></table>	Point	inGlobalPt	The point to check.
Point	inGlobalPt	The point to check.		
Return	Boolean of true if the point is in the drop area, else false.			
Remarks	This method is a pure virtual method, and your subclass must implement it.			

ReceiveDragItem()

Purpose	Process an item which has been dragged into a drop area. This method gets called once for each item contained in a completed drag. The item will have returned true from ItemIsAcceptable() . The drop area is focused upon entry and <code>inItemBounds</code> is specified in the local coordinates of the drop area. Override this function if the drop area can accept dropped items. You may want to call <code>GetFlavorData()</code> and <code>GetFlavorDataSize()</code> if there is information associated with the dragged Item.
---------	--

LDropArea

Access	Virtual, Protected		
Prototype	<pre>virtual, void ReceiveDragItem(DragReference inDragRef, DragAttributes inDragAttrs, ItemReference inItemRef, Rect& inItemBounds);</pre>		
Parameters	This method has the following parameters:		
	DragReference	inDragRef	The reference to the drag.
	DragAttributes	inDragAttrs	The drag attributes.
	ItemReference	inItemRef	The reference to the item that was drag and dropped.
	Rect&	inItemBounds	The Rect in local coordinates.
Return	None		

RemoveDropArea()

Purpose	Remove a drop area from the list maintained by this class.		
Access	Static, Protected		
Prototype	<pre>static void RemoveDropArea(LDropArea *inDropArea, WindowPtr inMacWindow);</pre>		
Parameters	This method has the following parameters:		
	LDropArea*	inDropArea	The window pointer.
	WindowPtr	inMacWindow	The reference to the area.
Return	None		

UnhiliteDropArea()

Purpose	Unhilite a drop area when a drag leaves the area. Subclasses should override this function if they override HiliteDropArea() to do something other than call <code>ShowDragHilite()</code> .		
Access	Virtual, Protected		
Prototype	<code>virtual void UnhiliteDropArea(DragReference inDragRef);</code>		
Parameters	This method has the following parameter:		
	DragReference	inDragRef	The reference to the area.
Return	None		

mDragWindow

Purpose	This data member is the <code>WindowPtr</code> for the window containing the drop area.
Access	Protected
Prototype	<code>WindowPtr mDragWindow;</code>

mCanAcceptCurrentDrag

Purpose	This data member indicates whether the drop area can receive the drag.
Access	Protected
Prototype	<code>Boolean mCanAcceptCurrentDrag;</code>

LDropArea

mIsHilited

Purpose	This data member indicates whether the drop area is currently highlighted.
Access	Protected
Prototype	<code>Boolean mIsHilited;</code>

sDragTrackingProc

Purpose	This data member stores the procedure pointer for the drag tracking procedure.
Access	Static
Prototype	<code>static DragTrackingHandlerUPP sDragTrackingProc;</code>

sDragReceiveProc

Purpose	This data member stores the procedure pointer for the drag receive procedure.
Access	Static
Prototype	<code>static DragReceiveHandlerUPP sDragReceiveProc;</code>

sDropAreaList

Purpose	This data member stores the drop area list.
Access	Static
Prototype	<code>static TArray<SDropAreaEntry> *sDropAreaList;</code>

sCurrentDropArea

Purpose	This data member stores the current drop area.
Access	Static
Prototype	<code>static LDropArea *sCurrentDropArea;</code>

sDragHasLeftSender

Purpose	This data member indicates whether the drag has left the sender window.
Access	Static
Prototype	<code>static Boolean sDragHasLeftSender;</code>

LEditField

Overview LEditField is a PowerPlant class that encapsulates the behavior of an editable text field, as commonly appears in a dialog box.

Methods The methods in this class are:

<u>LEditField()</u>	<u>~LEditField()</u>
<u>AdjustCursorSelf()</u>	<u>AdjustTextWidth()</u>
<u>AlignTextEditRects()</u>	<u>BeTarget()</u>
<u>ClickSelf()</u>	<u>DisableSelf()</u>
<u>DontBeTarget()</u>	<u>DrawBox()</u>
<u>DrawSelf()</u>	<u>EnableSelf()</u>
<u>FindCommandStatus()</u>	<u>FocusDraw()</u>
<u>GetDescriptor()</u>	<u>GetMacTEH()</u>
<u>GetValue()</u>	<u>HandleKeyPress()</u>
<u>HideSelf()</u>	<u>InitEditField()</u>
<u>MoveBy()</u>	<u>ObeyCommand()</u>
<u>ResizeFrameBy()</u>	<u>RestorePlace()</u>
<u>SavePlace()</u>	<u>SelectAll()</u>
<u>SetDescriptor()</u>	<u>SetKeyFilter()</u>
<u>SetMaxChars()</u>	<u>SetTextTraitsID()</u>
<u>SetValue()</u>	<u>SpendTime()</u>
<u>TooManyCharacters()</u>	<u>UseWordWrap()</u>
<u>UserChangedText()</u>	

Data Members The data members in this class are:

<u>mTextEditH</u>	<u>mKeyFilter</u>
<u>mTypingAction</u>	<u>mMaxChars</u>

LEditField

	mTextTraitsID	mHasBox
	mHasWordWrap	
Operation	LEditField uses TextEdit to implement an editable text field. PowerPlant also handles undo and redo for most text-related actions.	
Source files	(Pane Classes)	
	LEditField.h	
	LEditField.cp	
Ancestors	LAttachable	
	LCommander	
	LPane	
	LPeriodical	

LEditField()

Purpose	The constructors create objects from the passed-in parameters.
Access	Public
Prototype	<pre>LEditField(); LEditField(const LEditField &inOriginal); LEditField(const SPaneInfo &inPaneInfo, Str255 inString, ResIDT inTextTraitsID, SInt16 inMaxChars, Boolean inHasBox, Boolean inHasWordWrap, KeyFilterFunc inKeyFilter, LCommander *inSuper); LEditField(const SPaneInfo &inPaneInfo, Str255 inString, ResIDT inTextTraitsID, SInt16 inMaxChars,</pre>

```

    UInt8 inAttributes,
    KeyFilterFunc inKeyFilter,
    LCommander *inSuper );
LEditField( LStream *inStream );

```

Parameters		The parameters for these constructors are:
const LEditField&	inOriginal	The reference to the object to copy.
const SPaneInfo&	inPaneInfo	Reference to the Superpane info.
Str255	inString	The string for the field.
ResIDT	inTextTraitsID	The text traits resource ID.
SInt16	inMaxChars	The max number of characters.
Boolean	inHasBox	Indicates whether to see the editAttr_Box attribute or not.
Boolean	inHasWordWrap	Indicates whether the text field wraps. This sets the editAttr_WordWrap attribute.
KeyFilterFunc	inKeyFilter	A pointer to a key press filter procedure.
LCommander*	inSuper	The Supercommander.
UInt8	inAttributes	The attributes

~LEditField()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	virtual ~LEditField();

AdjustCursorSelf()

Purpose EditField uses the standard I-Beam cursor. This is an override of [AdjustCursorSelf\(\)](#) in [LPane](#).

AdjustTextWidth()

Purpose Adjust the width of the destination rectangle of the Toolbox TextEdit record.

This function does nothing if [mHasWordWrap](#) is true. If [mHasWordWrap](#) is false, this function sets the width of the TextEdit destination rectangle to either the width of the text or a very large number, depending on the value of `inShrinkToText`.

This adjustment is needed to make autoscrolling work properly when [mHasWordWrap](#) is off. While entering text, the destination rectangle should be very wide so that the text doesn't word wrap. However, while clicking, it should be just as wide as the text so that the EditField does not autoscroll past the edge of the text.

Access Virtual, Protected

Prototype `virtual void AdjustTextWidth(
 Boolean inShrinkToText);`

Parameters The parameter for this method is:

Boolean	<code>inShrinkToText</code>	Indicates whether to adjust the rectangle or not.
---------	-----------------------------	---

Return None

AlignTextEditRects()

Purpose	Align the view and destination rectangles of the Toolbox TextEdit record with the Frame of an EditField.
Access	Virtual, Protected
Prototype	<code>virtual void AlignTextEditRects();</code>
Parameters	None
Return	None

BeTarget()

Purpose	EditField is becoming the Target. This is an override of BeTarget() in LCommander .
---------	---

ClickSelf()

Purpose	Respond to Click inside an EditField. This is an override of ClickSelf() in LPane .
---------	---

DisableSelf()

Purpose	Disable an EditField. This is an override of DisableSelf() in LPane .
---------	---

DontBeTarget()

Purpose	EditField is becoming the Target. This is an override of DontBeTarget() in LCommander .
---------	---

DrawBox()

Purpose	Draw a box around an EditField.
Access	Virtual, Protected
Prototype	<code>virtual void DrawBox();</code>
Parameters	None
Return	None
Remarks	Box around an EditField is outset from the Text by 2 pixels. The box itself is 1 pixel thick, drawn in the foreground color of the Pane (not necessarily the same as the text color). If the EditField is disabled, the box draws with a gray pattern. The 1 pixel rectangle between the box and the text draws in the background color of the text.

DrawSelf()

Purpose	Draw an EditField. This is an override of DrawSelf() in LPane .
---------	---

EnableSelf()

Purpose	Enable an EditField. This is an override of EnableSelf() in LPane .
---------	---

FindCommandStatus()

Purpose	Pass back the status of a Command. This is an override of FindCommandStatus() in LCommander .
---------	---

FocusDraw()

Purpose Prepare for drawing in the EditField. This is an override of [FocusDraw\(\)](#) in [LPane](#).

GetDescriptor()

Purpose Return the first 255 characters of the EditField as a Pascal string. The caller must allocate a Str255 variable for storing the string. This is an override of [GetDescriptor\(\)](#) in [LPane](#).

GetMacTEH()

Purpose This method returns the handle to the Mac OS TextEdit structure.

Access Public

Prototype `TEHandle GetMacTEH();`

Parameters None

Return A handle to the TextEdit structure.

GetValue()

Purpose Return the integer value represented by the contents of an EditField. An empty or non-numerical EditField evaluates to zero. This is an override of [GetValue\(\)](#) in [LPane](#).

HandleKeyPress()

Purpose Handle key stroke directed at an EditField. Return true if the EditField handles the keystroke. This is an override of [HandleKeyPress\(\)](#) in [LCommander](#).

HideSelf()

Purpose Hide an EditField. An invisible EditField can't be on duty. This is an override of [HideSelf\(\)](#) in [LPane](#).

InitEditField()

Purpose Initialize member variables of a EditField to default values.

Access Private

Prototype `void InitEditField(UInt8 inAttributes);`

Parameters The parameter for this method is:

UInt8	inAttributes	The attributes, from the enum values in LEditField.h
-------	--------------	--

Return None

MoveBy()

Purpose Move the location of the Frame by the specified amounts. This is an override of [MoveBy\(\)](#) in [LPane](#).

ObeyCommand()

Purpose Handle standard editing commands. This is an override of [ObeyCommand\(\)](#) in [LCommander](#).

ResizeFrameBy()

Purpose Change the Frame size by the specified amounts `inWidthDelta` and `inHeightDelta` specify, in pixels, how much larger to make the Frame. Positive deltas increase the size, negative deltas reduce the size. This is an override of [ResizeFrameBy\(\)](#) in [LPane](#).

RestorePlace()

Purpose Save TextEdit rectangles. This is an override of [RestorePlace\(\)](#) in [LPane](#).

SavePlace()

Purpose Save TextEdit rectangles. This is an override of [SavePlace\(\)](#) in [LPane](#).

SelectAll()

Purpose Select entire contents of an EditField.

Access Public

Prototype `virtual void SelectAll();`

Parameters None

LEditField

Return None

SetDescriptor()

Purpose Set the contents of an EditField from a Pascal string. This is an override of [SetDescriptor\(\)](#) in [LPane](#).

SetKeyFilter()

Purpose Specify the function for filtering keystrokes.

Access Virtual, Public

Prototype `virtual void SetKeyFilter(KeyFilterFunc inKeyFilter);`

Parameters The parameter for this method is:

KeyFilterFunc	inKeyFilter	The function pointer for the key filter proc.
---------------	-------------	---

Return None

SetMaxChars()

Purpose Specify the maximum number of characters that an EditField can contain.

Access Virtual, Public

Prototype `virtual void SetMaxChars(SInt16 inMaxChars);`

Parameters The parameter for this method is:

SInt16	inMaxChars	The maximum number of characters.
--------	------------	-----------------------------------

Return None

SetTextTraitsID()

Purpose Specify the resource ID of the TextTraits for an EditField. This function updates the line height to fit the text characteristics.

Access Virtual, Public

Prototype `virtual void SetTextTraitsID(ResIDT
inTextTraitsID);`

Parameters The parameter for this method is:

ResID T	inTextTraitsID	The resource ID for the text traits resource.
------------	----------------	---

Return None

SetValue()

Purpose Set the contents of an EditField to the string representation of aspecified integer number. This is an override of [SetValue\(\)](#) in [LPane](#).

SpendTime()

Purpose At idle time, flash the insertion cursor. This is an override of [SpendTime\(\)](#) in [LPeriodical](#).

TooManyCharacters()

Purpose	Indicate whether adding the specified number of characters will exceed the maximum allowed. This method assumes that the characters being added will replace the current selection.			
Access	Virtual, Protected			
Prototype	<code>virtual Boolean TooManyCharacters(SInt32 inCharsToAdd);</code>			
Parameters	<div>The parameter for this method is:<table><tr><td><code>SInt32</code></td><td><code>inCharsToAdd</code></td><td>The number of characters you want to test for the maximal limit.</td></tr></table></div>	<code>SInt32</code>	<code>inCharsToAdd</code>	The number of characters you want to test for the maximal limit.
<code>SInt32</code>	<code>inCharsToAdd</code>	The number of characters you want to test for the maximal limit.		
Return	Returns <code>true</code> if the maximum will be exceeded, else <code>false</code> .			

UseWordWrap()

Purpose	Specify whether the EditField word wraps to its frame.			
Access	Virtual, Public			
Prototype	<code>virtual void UseWordWrap(Boolean inSetting);</code>			
Parameters	<div>The parameter for this method is:<table><tr><td><code>Boolean</code></td><td><code>inSetting</code></td><td>Indicate whether to wrap or not.</td></tr></table></div>	<code>Boolean</code>	<code>inSetting</code>	Indicate whether to wrap or not.
<code>Boolean</code>	<code>inSetting</code>	Indicate whether to wrap or not.		
Return	None			

UserChangedText()

Purpose	Text of EditField has changed as a result of the user actions. You should override this to validate the field and/or dynamically update the field as the user types. This function is not called by
---------	---

[SetDescriptor\(\)](#), which is typically used to programatically change the text.

Access	Virtual, Public
Prototype	<code>virtual void UserChangedText();</code>
Parameters	None
Return	None

mTextEditH

Purpose	The handle to the TextEdit record.
Access	Protected
Prototype	<code>TEHandle mTextEditH;</code>

mKeyFilter

Purpose	The key filter proc storage space.
Access	Protected
Prototype	<code>KeyFilterFunc mKeyFilter;</code>

mTypingAction

Purpose	Pointer to an LTETypingAction object.
Access	Protected
Prototype	<code>LTETypingAction *mTypingAction;</code>

LTextField

mMaxChars

Purpose	The maximum number of characters that the field can accomodate.
Access	Protected
Prototype	<code>SInt16 mMaxChars;</code>

mTextTraitsID

Purpose	The resource ID for the text traits.
Access	Protected
Prototype	<code>ResIDT mTextTraitsID;</code>

mHasBox

Purpose	Indicates whether the <code>editAttr_Box</code> enum attribute is set.
Access	Protected
Prototype	<code>Boolean mHasBox;</code>

mHasWordWrap

Purpose	Indicates whether the <code>editAttr_WordWrap</code> enum attribute is set.
Access	Protected
Prototype	<code>Boolean mHasWordWrap;</code>

LEndpoint

Overview	LEndpoint is a PowerPlant class that forms the basis for networking. Most of the methods are pure virtual, which means that you need to supply implementations if you inherit from this class.	
Methods	The methods in this class are:	
	LEndpoint()	~LEndpoint()
	AbortThreadOperation()	AckSends()
	Bind()	DontAckSends()
	DontQueueSends()	GetLocalAddress()
	GetState()	IsAckingSends()
	IsQueuingSends()	QueueSends()
	Unbind()	
Data Members	The data members in this class are:	
	mQueueSends	
Operation	<p>This class encapsulates the idea of a "network endpoint," or one side of a two-way communication link. The endpoint is the object that is used to send or receive data over the network.</p> <p>LEndpoint is an abstract base class (thus its constructor is declared protected). Use one of the predefined subclasses, such as LMacTCPTCP endpoint or LOpenTptTCP endpoint, or use the UNetworkFactory::CreateTCP endpoint() function to create the appropriate endpoint for the system software that's installed on the user's machine.</p>	
Source files	(Networking Classes)	
	LEndpoint.h	
	LEndpoint.cp	
See also	LBroadcaster	
	UNetworkFactory	

LEndpoint

LEndpoint()

Purpose	The constructor creates the object.
Access	Public
Prototype	<code>LEndpoint () ;</code>
Parameters	None

~LEndpoint()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LEndpoint () ;</code>
Parameters	None

AbortThreadOperation()

Purpose	This method must be implemented by your class since it is pure virtual. Abort a thread operation.	
Access	Pure virtual, Public	
Prototype	<code>virtual voidAbortThreadOperation(LThread * inThread) = 0;</code>	
Parameters	The parameter for this methods is:	
	<code>LThread*</code>	<code>inThread</code> The pointer to the thread.
Return	None	

AckSends()

Purpose	Enable the acknowledgement of sent data for the endpoint.
Access	Pure virtual, Public
Prototype	<code>virtual voidAckSends() = 0;</code>
Parameters	None
Return	None

Bind()

Purpose	Reserve a TCP port for the connection.
Access	Pure virtual, Public
Prototype	<code>virtual voidBind(LInternetAddress& inLocalAddress, UInt32 inListenQueueSize = 0, Boolean inReusePort = true) = 0;</code>

Parameters The parameters for this method are:

LInternetAddress&	inLocalAddress	The local address.
UInt32	inListenQueueSize	The listen queue size in bytes, the default is 0.
Boolean	inReusePort	Whether to reuse the port or not, the default is 0.

Return None

LEndpoint

DontAckSends()

Purpose	Disable the acknowledgement of sent data for the endpoint.
Access	Pure virtual, Public
Prototype	<code>virtual void DontAckSends() = 0;</code>
Parameters	None
Return	None

DontQueueSends()

Purpose	This method sets the value of the mQueueSends data member to false.
Access	Inline, Public, Virtual
Prototype	<code>virtual void DontQueueSends();</code>
Parameters	None
Return	None

GetLocalAddress()

Purpose	Return the network address of your local computer.
Access	Pure virtual, Public
Prototype	<code>virtual LInternetAddress*GetLocalAddress() = 0;</code>
Parameters	None
Return	A pointer to the address.

GetState()

Purpose	Return the current state of the endpoint.
Access	Pure virtual, Public
Prototype	<code>virtual EEndpointStateGetState() = 0;</code>
Parameters	None
Return	None

IsAckingSends()

Purpose	Determine whether the endpoint is acknowledging sends.
Access	Pure virtual, Public
Prototype	<code>virtual BooleanIsAckingSends() = 0;</code>
Parameters	None
Return	Returns true if the endpoint is ACK-ing sends.

IsQueuingSends()

Purpose	Determines if the endpoint is queuing sends.
Access	Virtual, Public
Prototype	<code>virtual BooleanIsQueuingSends();</code>
Parameters	None
Return	Returns true if the endpoint is queuing sends.

LEndpoint

QueueSends()

Purpose	Directs the endpoint to queue sends.
Access	Virtual, Public
Prototype	<code>virtual voidQueueSends();</code>
Parameters	None
Return	None

Unbind()

Purpose	Release a previously-bound TCP port.
Access	Pure virtual, Public
Prototype	<code>virtual voidUnbind() = 0;</code>
Parameters	None
Return	None

mQueueSends

Purpose	State variable for whether to queue send traffic for the endpoint.
Access	Protected
Prototype	<code>Boolean mQueueSends;</code>

LEraseAttachment

Overview	LEraseAttachment is a PowerPlant class that is used for erasing the Frame of a Pane.
Methods	The methods in this class are: LEraseAttachment() ExecuteSelf()
Data Members	There are no data members for this class.
Operation	This Attachment is designed to respond to the <code>msg_DrawOrPrint</code> message. When you create this Attachment, you specify whether the host should also draw.
Source files	(Utility Classes) <code>UAttachments.h</code> <code>UAttachments.cp</code>
Ancestors	LAttachment LPane
See Also	LPaintAttachment

LEraseAttachment()

Purpose	The constructor creates the object.
Access	Public
Prototype	<code>LEraseAttachment(Boolean inExecuteHost);</code> <code>LEraseAttachment(LStream *inStream);</code>
Parameters	The parameters for these constructors are:

LEraseAttachment

Boolean	inExecuteHost	A Boolean indicating the value to be set for mExecuteHost .
LStream*	inStream	A pointer to a stream object that contains the information to create the LEraseAttachment object.

ExecuteSelf()

Purpose	Erases the Frame of a Pane. This is an override of ExecuteSelf() in LAttachment .
---------	---

LEventDispatcher

Overview	LEventDispatcher is a PowerPlant class that is used for handling all event processing after the application retrieves the event in the main event loop. This class is also responsible for adjusting the cursor, distributing time to idle-time processes, and initiating menu updates before displaying menus.	
Methods	The methods in this class are:	
	<u>LEventDispatcher()</u>	<u>~LEventDispatcher()</u>
	<u>AdjustCursor()</u>	<u>ClickMenuBar()</u>
	<u>DispatchEvent()</u>	<u>EventActivate()</u>
	<u>EventAutoKey()</u>	<u>EventDisk()</u>
	<u>EventHighLevel()</u>	<u>EventKeyDown()</u>
	<u>EventKeyUp()</u>	<u>EventMouseDown()</u>
	<u>EventMouseUp()</u>	<u>EventOS()</u>
	<u>EventResume()</u>	<u>EventSuspend()</u>
	<u>EventUpdate()</u>	<u>ExecuteAttachments()</u>
	<u>GetCurrentEventDispatcher()</u>	<u>UpdateMenus()</u>
	<u>UseIdleTime()</u>	
Data Members	The data members in this class are:	
	<u>sCurrentDispatcher</u>	<u>mSaveDispatcher</u>
	<u>mMouseRgnH</u>	
Operation	Most of the time, you won't need to override any functions in this class. PowerPlant supplies default behaviors that implement much of the functionality you could hope for.	
	Unless you use the Mac Toolbox directly for tasks such as running a dialog box with <code>ModalDialog()</code> , all event processing goes through the main event loop and LEventDispatcher, even for modal dialogs.	
Source files	(Features Classes)	

LEventDispatcher

LEventDispatcher.h

LEventDispatcher.cp

See also [LAttachable](#)

LEventDispatcher()

Purpose	The constructor creates the object, initializing the menus so that they are unhilit, and saving a copy of the current Event Dispatcher.
Access	Public
Prototype	<code>LEventDispatcher();</code>
Parameters	None

~LEventDispatcher()

Purpose	The destructor destroys the object, including disposal of the mouse region, and restoring of the previous Event Dispatcher.
Access	Virtual, Public
Prototype	<code>virtual ~LEventDispatcher();</code>
Parameters	None

AdjustCursor()

Purpose	Adjust the shape of the cursor (mouse pointer). If the cursor is inside an active, enabled, PowerPlant window, then the window is expected to handle cursor adjustment, using
Access	Virtual, Protected

Prototype	<code>virtual void AdjustCursor(const EventRecord &inMacEvent);</code>
-----------	---

Parameters	The parameter for this method is:
------------	-----------------------------------

<code>const EventRecord &</code>	<code>inMacEvent</code>	This passed-in reference tells the location of the mouse cursor.
--	-------------------------	--

Return	None
--------	------

ClickMenuBar()

Purpose	Respond to a mouse click in the menu bar.
---------	---

Access	Virtual, Protected
--------	--------------------

Prototype	<code>virtual void ClickMenuBar(const EventRecord &inMacEvent);</code>
-----------	---

Parameters	The parameter for this method is:
------------	-----------------------------------

<code>const EventRecord &</code>	<code>inMacEvent</code>	This passed-in reference contains event information.
--	-------------------------	--

Return	None
--------	------

DispatchEvent()

Purpose	This method processes a Mac OS Toolbox event.
---------	---

Access	Virtual, Public
--------	-----------------

Prototype	<code>virtual void DispatchEvent(const EventRecord &inMacEvent);</code>
-----------	--

Parameters	The parameter for this method is:
------------	-----------------------------------

LEventDispatcher

	<code>const EventRecord &</code>	<code>inMacEvent</code>	This passed-in reference contains event information.
Return	None		

EventActivate()

Purpose	Respond to an Activate (or Deactivate) event by enabling the appropriate window and updating the command status.		
Access	Virtual, Protected		
Prototype	<code>virtual void EventActivate(const EventRecord &inMacEvent);</code>		
Parameters	The parameter for this method is:		
	<code>const EventRecord &</code>	<code>inMacEvent</code>	This passed-in reference contains event information.
Return	None		

EventAutoKey()

Purpose	Respond to an auto-key event.		
Access	Virtual, Protected		
Prototype	<code>virtual void EventAutoKey(const EventRecord &inMacEvent);</code>		
Parameters	The parameter for this method is:		

	const EventRecord &	inMacEvent	This passed-in reference contains event information.
Return	None		

EventDisk()

Purpose	Respond to a Disk-inserted event.		
Access	Virtual, Protected		
Prototype	virtual void EventDisk(const EventRecord &inMacEvent);		
Parameters	The parameter for this method is:		
	const EventRecord &	inMacEvent	This passed-in reference contains event information.
Return	None		

EventHighLevel()

Purpose	Respond to a High Level (Apple) Event.		
Access	Virtual, Protected		
Prototype	virtual void EventHighLevel(const EventRecord &inMacEvent);		
Parameters	The parameter for this method is:		
	const EventRecord &	inMacEvent	This passed-in reference contains event information.

Return None

EventKeyDown()

Purpose Respond to a Key Down Event.

Access Virtual, Protected

Prototype `virtual void EventKeyDown(
 const EventRecord &inMacEvent);`

Parameters The parameter for this method is:

<code>const EventRecord &</code>	<code>inMacEvent</code>	This passed-in reference contains event information.
--	-------------------------	---

Return None

EventKeyUp()

Purpose Respond to a Key Up event. Note that by default, the system masks out Key Up events.

Access Virtual, Protected

Prototype `virtual void EventKeyUp(
 const EventRecord& inMacEvent);`

Parameters The parameter for this method is:

<code>const EventRecord &</code>	<code>inMacEvent</code>	This passed-in reference contains event information.
--	-------------------------	---

Return None

EventMouseDown()

Purpose	Respond to a Mouse Down event.	
Access	Virtual, Protected	
Prototype	<pre>virtual void EventMouseDown(const EventRecord &inMacEvent);</pre>	
Parameters	The parameter for this method is:	
	<pre>const inMacEvent EventRecord &</pre>	This passed-in reference contains event information.
Return	None	

EventMouseUp()

Purpose	Respond to a Mouse Up event.	
Access	Virtual, Protected	
Prototype	<pre>virtual void EventMouseUp(const EventRecord &inMacEvent);</pre>	
Parameters	The parameter for this method is:	
	<pre>const inMacEvent EventRecord &</pre>	This passed-in reference contains event information.
Return	None	

EventOS()

Purpose	Respond to an OS event (MouseMoved, Suspend, Resume).
---------	---

LEventDispatcher

Access	Virtual, Protected	
Prototype	<pre>virtual void EventOS(const EventRecord &inMacEvent);</pre>	
Parameters	The parameter for this method is:	
	<pre>const inMacEvent EventRecord &</pre>	This passed-in reference contains event information.
Return	None	

EventResume()

Purpose	Respond to a Resume event.	
Access	Virtual, Protected	
Prototype	<pre>virtual void EventResume(const EventRecord& inMacEvent);</pre>	
Parameters	The parameter for this method is:	
	<pre>const inMacEvent EventRecord &</pre>	This passed-in reference contains event information.
Return	None	

EventSuspend()

Purpose	Respond to a Suspend event.	
Access	Virtual, Protected	
Prototype	<pre>virtual void EventSuspend(const EventRecord& inMacEvent);</pre>	

Parameters	The parameter for this method is:		
	<code>const EventRecord &</code>	<code>inMacEvent</code>	This passed-in reference contains event information.
Return	None		

EventUpdate()

Purpose	Respond to an Update event.		
Access	Virtual, Protected		
Prototype	<code>virtual void EventUpdate(const EventRecord &inMacEvent);</code>		
Parameters	The parameter for this method is:		
	<code>const EventRecord &</code>	<code>inMacEvent</code>	This passed-in reference contains event information.
Return	None		

ExecuteAttachments()

Purpose	Tell all associated Attachments to execute themselves for the specified message.		
Access	Virtual, Public		
Prototype	<code>virtual Boolean ExecuteAttachments(MessageT inMessage, void *ioParam);</code>		
Parameters	The parameters for this method are:		

LEventDispatcher

	MessageT	inMessage	This passed-in value indicates the message to process.
	void*	ioParam	This passed-in pointer is for a data block that accompanies the inMessage.
Return	The return value specifies whether the default Host action should be executed. The value is false if any Attachment's Execute() function returns false, otherwise it's true.		

GetCurrentEventDispatcher()

Purpose	Get a pointer to the current LEventDispatcher object so that it can be later restored.
Access	Static, Public
Prototype	<code>static LEventDispatcher *GetCurrentEventDispatcher();</code>
Parameters	None
Return	A LEventDispatcher pointer that points to the current LEventDispatcher.
Remarks	This is a static method, and as such the same method will be called for every instance of the LEventDispatcher objects.

UpdateMenus()

Purpose	This method handles all the menu update operations that are required for event processing.
Access	Virtual, Public
Prototype	<code>virtual void UpdateMenus();</code>
Parameters	None

Return	None
Remarks	This method handles many cases, and is best understood by reading the source code.

UseIdleTime()

Purpose	Respond to a NULL or MouseMoved Event.		
Access	Virtual, Public		
Prototype	<pre>virtual void UseIdleTime(const EventRecord &inMacEvent);</pre>		
Parameters	The parameter for this method is:		
	const	inMacEvent	This passed-in reference contains event information.
	EventRecord		
	&		
Return	None		
Remarks	Typically, all this method needs to do is devote time to processing idle tasks.		

sCurrentDispatcher

Purpose	This data member provides a place to store the current EventDispatcher object pointer.
Access	Protected
Prototype	<code>static LEventDispatcher *sCurrentDispatcher;</code>

LEventDispatcher

mSaveDispatcher

Purpose	This data member provides a place to store the previous EventDispatcher object pointer. From this, the former EventDispatcher can be restored later.
Access	Protected
Prototype	<code>LEventDispatcher *mSaveDispatcher;</code>

mMouseRgnH

Purpose	This data member provides a place to store the mouse region handle.
Access	Protected
Prototype	<code>RgnHandle mMouseRgnH;</code>

LEventSemaphore

Overview	LEventSemaphore is a PowerPlant class that overrides LSemaphore .
Methods	The methods in this class are: <div>LEventSemaphore()~LEventSemaphore()</div> <div>Reset()Signal()</div>
Data Members	The data members in this class are: <div>mPostCount</div>
Operation	The Signal() method is overridden so that it can be made to simultaneously release all threads waiting on this semaphore. The Reset() method is guaranteed to raise the semaphore so that no thread can access the flagged data until the next call to Signal() .
Source files	(Thread Classes) LEventSemaphore.h LEventSemaphore.cp
See also	LSemaphore

LEventSemaphore()

Purpose	The constructor creates the object.
Access	Public
Prototype	<code>LEventSemaphore () ;</code> <code>LEventSemaphore (Boolean posted) ;</code>
Parameters	A Boolean indicating whether the semaphore is posted or not. If posted is true, the semaphore is marked as being in the posted state, and any calls to Wait() will return immediately.

LEventSemaphore

~LEventSemaphore()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>~LEventSemaphore();</code>

Reset()

Purpose	Guaranteed to raise the semaphore so that no thread can access the flagged data until the next call to Signal() . Throws an exception if the semaphore is already reset.
Access	Virtual, Public
Prototype	<code>virtual UInt32 Reset();</code>
Parameters	None
Return	Returns the number of times that Signal() was called since the last call to <code>Reset()</code> .

Signal()

Purpose	Make the semaphore available to all threads. This includes all blocked threads, as well as those that will call Wait() before the next call to Reset() . This is an override of Signal() in LSemaphore .
Access	Virtual
Prototype	<code>virtual void Signal();</code>
Parameters	None
Return	None

mPostCount

Purpose	The “post” counter
Access	protected
Prototype	UInt32 mPostCount;

LFile

Description LFile is a PowerPlant class that is used as a wrapper class for a Macintosh file with a data and/or a resource fork.

A `FSSpec` (File System Specification) record identifies a Mac file. When open, the data fork has a file `refNum`. Likewise, when open, the resource fork has a file `refNum`. The `LFile` class stores an `FSSpec` and the `refNums` for the data and resource forks.

This class does not provide many functions for manipulating files. You should get the `FSSpec` or `refNum` of the fork you want to manipulate and make direct calls to the Mac File Manager. However, use the member functions for opening and closing data and resource forks.

The only file accessing functions provided are ones for reading and writing the entire data fork.

Methods The methods in this class are:

LFile()	~LFile()
CloseDataFork()	CloseResourceFork()
CreateNewDataFile()	CreateNewFile()
EqualFileSpec()	GetSpecifier()
MakeAlias()	OpenDataFork()
OpenResourceFork()	ReadDataFork()
SetSpecifier()	UsesSpecifier()
WriteDataFork()	

Data Members The data members in this class are:

mMacFileSpec	mDataForkRefNum
mResourceForkRefNum	

Source files (File & Stream Classes)

`LFile.h`

LFile

LFile.cp

See also

LFile()

Purpose There are three constructor methods.

The default constructor sets the items in [mMacFileSpec](#) to 0, [mDataForkRefNum](#) and [mResourceForkRefNum](#) to refNum_Undefined.

The second constructor is a parameter constructor which sets [mMacFileSpec](#) to inFileSpec, [mDataForkRefNum](#) and [mResourceForkRefNum](#) to refNum_Undefined.

The third constructor is also a parameter constructor that creates a File from an AliasHandle. outWasChanged indicates if the AliasHandle was changed during resolution. inFromFile is a File Specifier for the starting point for a relative search. If nil, an absolute search should be performed. Also sets [mDataForkRefNum](#) and [mResourceForkRefNum](#) to refNum_Undefined. Calls an exception if the OS returns an error.

Access Public

Prototypes

```
LFile(); // Default Constructor
LFile (const FSSpec &inFileSpec);
LFile ( AliasHandle inAlias,
        Boolean& outWasChanged,
        FSSpec* inFromFile);
```

Return No return value for a constructor.

~LFile()

Purpose The destructor closes the data and resource forks before deletion.

Access Public

Prototype	<code>virtual ~LFile();</code>
Parameters	None
Return	No return value for a destructor

CloseDataFork()

Purpose	Close the data fork of a File (if any). It's a good idea to close files just after reading or writing, to avoid damage if something crashes.
Access	Public, Virtual
Prototype	<code>void CloseDataFork();</code>
Parameters	None
Return	None
Remarks	Throws an exception if it fails.

CloseResourceFork()

Purpose	Close the resource fork of a File (if any). It's a good idea to close files just after reading or writing, to avoid damage if something crashes.
Access	Public, Virtual
Prototype	<code>void CloseResourceFork();</code>
Parameters	None
Return	None
Remarks	Throws an exception if it fails.

CreateNewDataFile()

Purpose	Create a new disk File, with an empty data fork and no resource map. You must call <code>OpenDataFork</code> (with write permission) before you can store data in the File. The resource fork is uninitialized (no resource map), so you can't call <code>OpenResourceFork</code> for the File. You can initialize the resource fork by calling <code>CreateNewFile</code> .											
Access	Virtual, Public											
Prototype	<pre>void CreateNewDataFile(OSType inCreator, OSType inFileType, ScriptCode inScriptCode);</pre>											
Parameters	<table><tr><td>OSType</td><td>inCreator</td><td>Creator type of file</td></tr><tr><td>OSType</td><td>inFileType</td><td>File type ('TEXT', etc.)</td></tr><tr><td>ScriptCode</td><td>inScriptCode</td><td>Script code (e.g.: smSystemScript)</td></tr></table>			OSType	inCreator	Creator type of file	OSType	inFileType	File type ('TEXT', etc.)	ScriptCode	inScriptCode	Script code (e.g.: smSystemScript)
OSType	inCreator	Creator type of file										
OSType	inFileType	File type ('TEXT', etc.)										
ScriptCode	inScriptCode	Script code (e.g.: smSystemScript)										
Return	None											
Remarks	The resource fork is uninitialized (no resource map), so you can't call OpenResourceFork() for the file. You can initialize the resource fork by calling CreateNewFile() .											

CreateNewFile()

Purpose	Create a new disk File, with an empty data fork and a resource map. You must call <code>OpenDataFork</code> or <code>OpenResourceFork</code> (with write permission) before you can store information in the File. If the file already exists, but doesn't have a resource map, this function will create a resource map.
Access	Virtual, Public
Prototype	<pre>void CreateNewFile(OSType inCreator,</pre>

```
OSType inFileType,
ScriptCode inScriptCode);
```

Parameters

OSType	inCreator	Creator type of file
OSType	inFileType	File type ('TEXT', etc.)
ScriptCode	inScriptCode	Script code (e.g.: smSystemScript)

Return None

Remarks If the file already exists, but doesn't have a resource map, this function will create a resource map.

EqualFileSpec()

Purpose Compare two FSSpec structs for equality. Compares each field in the FSSpec struct.

Access Static, Public

```
Boolean EqualFileSpec(
    const FSSpec& inFileSpecA,
    const FSSpec& inFileSpecB);
```

Parameters

const FSSpec	inFileSpecA	First FSSpec struct
const FSSpec	inFileSpecB	Second FSSpec struct

Return True if both FSSpec structs are equal.

GetSpecifier()

Purpose Return the current Mac File System Specification record ([mMacFileSpec](#)) for a File.

Access Public

LFile

Prototype	<code>void GetSpecifier(FSSpec& outFileSpec) const;</code>		
Parameters	FSSpec	outFileSpec	File System Specification record
Return	None		

MakeAlias()

Purpose	Return a newly created AliasHandle for a File.		
Access	Virtual, Public		
Prototype	<code>virtual AliasHandle MakeAlias(FSSpec* inFromFile);</code>		
Parameters	FSSpec	inFromFile	File Specifier for the starting point for a relative search. Pass nil if you don't need relative path information.
Return	None		

OpenDataFork()

Purpose	Open the data fork of a File with the specified permissions and return the reference number for the opened fork A data fork must be Open before you can read or write data		
Access			
Prototype	<code>virtual SInt16 OpenDataFork(SInt16 inPrivileges);</code>		
Parameters	SInt16	inPrivileges	Read/Write permissions.

Return	Reference number for the open data fork.
Remarks	A data fork must be open before you can read or write data.

OpenResourceFork()

Purpose	Open the resource fork of a File with the specified permissions and return the reference number for the opened fork. A resource fork must be Open before you can read or write resources.
Access	Virtual, Public
Prototype	<pre>virtual SInt16 OpenResourceFork(SInt16 inPrivileges)</pre>
Parameters	<pre>SInt16 inPrivileges Read/Write permissions.</pre>
Return	Referencenumber for the open resource fork.
Remarks	A resource fork must be open before you can read or write resources.

ReadDataFork()

Purpose	Read the entire contents of a File's data fork into a newly created Handle. The caller is responsible for disposing of the Handle.
Access	Virtual, Public
Prototype	<pre>virtual Handle ReadDataFork();</pre>
Parameters	None
Return	Handle to the data fork.
Remarks	Throws an exception if it fails.

SetSpecifier()

Purpose	Set a new Toolbox File System Specification for a File. This has the side effect of closing any open forks of the file identified by the old Specifier.
Access	Virtual, Public
Prototype	<pre>virtual void SetSpecifier(const FSSpec& inFileSpec);</pre>
Parameters	<pre>const FSSpec inFileSpec new File Specification</pre>
Return	None

UsesSpecifier()

Purpose	Returns whether the File's FSSpec is the same as the input FSSpec.
Access	Public
Prototype	<pre>Boolean UsesSpecifier(const FSSpec& inFileSpec) const;</pre>
Parameters	<pre>const FSSpec inFileSpec File specification to compare</pre>
Return	True if FSSpec's are the same.
Remarks	Calls EqualFileSpec() .

WriteDataFork()

Purpose	Write to the data fork of a File from a buffer. The buffer contents completely replace any existing data.
---------	---

Access	Virtual, Public		
Prototype	<pre>SInt32 WriteDataFork(const void* inBuffer, SInt32 inByteCount);</pre>		
Parameters			
	<code>const void*</code>	<code>inBuffer</code>	Pointer to data buffer
	<code>SInt32</code>	<code>inByteCount</code>	Size of data buffer to write
Return	Number of bytes actually written.		
Remarks	Throws an exception if it fails.		

mMacFileSpec

Purpose	Stores the FSSpec of the file.
Access	Protected
Prototype	<code>FSSpec mMacFileSpec;</code>

mDataForkRefNum

Purpose	<code>refNum</code> for the File's data fork.
Access	Protected
Prototype	<code>SInt16 mDataForkRefNum;</code>

mResourceForkRefNum

Purpose	<code>refNum</code> for the File's resource fork.
Access	Protected
Prototype	<code>SInt16 mResourceForkRefNum;</code>

LFileStream

Description	<p>LFileStream combines LFile and LStream into a single class. The file uses a stream to access the data fork.</p> <p>This class does not provide many functions for manipulating files. You should get the <code>FSSpec</code> or <code>refNum</code> of the fork you want to manipulate and make direct calls to the Mac File Manager. However, use the member functions for opening and closing data forks.</p> <p>The only file accessing functions provided are ones for reading and writing the data fork, managing the file marker and file length.</p>								
Methods	<p>The methods in this class are:</p> <table><tr><td>LFileStream()</td><td>~LFileStream()</td></tr><tr><td>GetLength()</td><td>SetLength()</td></tr><tr><td>GetMarker()</td><td>SetMarker()</td></tr><tr><td>GetBytes()</td><td>PutBytes()</td></tr></table>	LFileStream()	~LFileStream()	GetLength()	SetLength()	GetMarker()	SetMarker()	GetBytes()	PutBytes()
LFileStream()	~LFileStream()								
GetLength()	SetLength()								
GetMarker()	SetMarker()								
GetBytes()	PutBytes()								
Data Members	There are no direct data members in this class								
Source files	<p>(File & Stream Classes)</p> <p><code>LFileStream.h</code></p> <p><code>LFileStream.cp</code></p>								
See also	<p>LFile</p> <p>LStream</p>								

LFileStream()

Purpose	<p>There are three constructor methods.</p> <p>All three constructor's call the corresponding LFile constructor methods.</p>
---------	--

LFileStream

Access	Public
Prototypes	<pre>LFileStream(); // Default Constructor LFileStream(const FSSpec &inFileSpec); LFileStream(AliasHandle inAlias, Boolean& outWasChanged, FSSpec* inFromFile = nil);</pre>
Return	No return value for a constructor.
Remarks	Just calls the inherited methods.

~LFileStream()

Purpose	The destructor closes the data and resource forks before deletion.
Access	Public
Prototype	<pre>virtual ~LFileStream();</pre>
Parameters	None
Return	No return value for a destructor

GetLength()

Purpose	Return the length, in bytes, of the data fork of a FileStream.
Access	Public, Virtual
Prototype	<pre>SInt32 GetLength() const;</pre>
Parameters	None
Return	Size of stream in bytes.
Remarks	Throws an exception if it fails.

SetLength()

Purpose	Close the resource fork of a File (if any). It's a good idea to close files just after reading or writing, to avoid damage if something crashes.		
Access	Public, Virtual		
Prototype	<code>void SetLength(SInt32 inLength);</code>		
Parameters	SInt32	inLength	size of stream in bytes to write
Return	None		
Remarks	Throws an exception if it fails.		

GetMarker()

Purpose	Get the Read/Write Marker position as a byte offset from the start of the data fork stored by the Mac OS.		
Access	Virtual, Public		
Prototype	<code>SInt32 GetMarker() const;</code>		
Parameters	None		
Return	Offset of Marker from start of the data fork.		
Remarks	None		

SetMarker()

Purpose	Place the Read/Write Marker at an offset from a specified position. <code>inFromWhere</code> can be <code>streamFrom_Start</code> , <code>streamFrom_End</code> , or <code>streamFrom_Marker</code> .		
Access	Virtual, Public		

LFileStream

Prototype	<pre>void SetMarker(SInt32 inOffset, EStreamFrom inFromWhere);</pre>		
Parameters	SInt32	inOffset	Offset position in bytes
	EStreamFrom	inFromWhere	Position setting
Return	None		
Remarks	Throws an exception if it fails.		

GetBytes()

Purpose	Reads data from the data fork of a DataStream to a buffer. Passes back the number of bytes actually read which may be less than the amount requested.		
Access	Public, Virtual		
Prototype	<pre>ExceptionCode GetBytes(void* outBuffer, SInt32& ioByteCount);</pre>		
Parameters	void*	outBuffer	Pointer to the data buffer
	SInt32&	ioByteCount	Number of bytes to read
Return	Error code providing status of the operation. noErr means operation succeeded.		

PutBytes()

Purpose	Writes data from a buffer to the DataStream. Passes back the number of bytes actually written which may be less than the amount requested.		
Access	Public, Virtual		

Prototype `ExceptionCode PutBytes(
 const void* inBuffer,
 SInt32& ioByteCount);`

Parameters

<code>void*</code>	<code>inBuffer</code>	Pointer to the data buffer
<code>SInt32&</code>	<code>ioByteCount</code>	Number of bytes to write

Return Error code providing status of the operation. `noErr` means operation succeeded.

LFocusBox

Overview	LFocusBox is a PowerPlant class that is used for outlining a pane to indicate that the pane is the current focus for keystrokes.								
Methods	<p>The methods in this class are:</p> <table><tr><td>LFocusBox()</td><td>AttachPane()</td></tr><tr><td>DontRefresh()</td><td>DrawSelf()</td></tr><tr><td>GetBoxRegion()</td><td>HideSelf()</td></tr><tr><td>Refresh()</td><td>ShowSelf()</td></tr></table>	LFocusBox()	AttachPane()	DontRefresh()	DrawSelf()	GetBoxRegion()	HideSelf()	Refresh()	ShowSelf()
LFocusBox()	AttachPane()								
DontRefresh()	DrawSelf()								
GetBoxRegion()	HideSelf()								
Refresh()	ShowSelf()								
Data Members	There are no data members in this class.								
Operation	This class is used internally by PowerPlant in conjunction with LListBox to highlight entries in the list.								
Source files	<p>(Pane Classes)</p> <p><code>LFocusBox.h</code></p> <p><code>LFocusBox.cp</code></p>								
Ancestors	<p>LAttachable</p> <p>LPane</p>								

LFocusBox()

Purpose	The constructors create objects from the passed-in parameters.
Access	Public
Prototype	<pre>LFocusBox () ; LFocusBox(const LFocusBox &inOriginal) ; LFocusBox(LStream *inStream) ;</pre>
Parameters	The parameters for these constructors are:

const LFocusBox&	inOriginal	The reference to the object to copy.
LStream*	inStream	The stream to read from.

AttachPane()

Purpose	Associate a host Pane with a FocusBox, sizing and positioning the FocusBox so that it fits around the Pane.		
Access	Virtual, Public		
Prototype	<pre>virtual void AttachPane(LPane *inPane, Boolean inSameBindings);</pre>		
Parameters	The parameters for this method are:		
	LPane*	inPane	The pointer to the pane.
	Boolean	inSameBin dings	If true, FocusBox uses same frame bindings as the host Pane. This makes the FocusBox move and resize along with the host Pane if their mutual SuperView moves or resizes.
Return	None		

DontRefresh()

Purpose	Validate the area occupied by a FocusBox so it won't redraw during the next Update event. This is an override of DontRefresh() in LView .
---------	---

DrawSelf()

Purpose Draw a FocusBox. This is an override of [DrawSelf\(\)](#) in [LPane](#).

GetBoxRegion()

Purpose Pass back a region defining a FocusBox. A FocusBox is a hollow rectangular region. Clip the region so that it is within the revealed rect. This method creates a new region which the caller must dispose of.

Access Protected

Prototype `virtual RgnHandle GetBoxRegion(
const Rect &inFrame,
const Rect &inRevealed) const;`

Parameters The parameters for this method are:

<code>const Rect&</code>	<code>inFrame</code>	Reference to the frame Rect.
<code>const Rect&</code>	<code>inRevealed</code>	Reference to the revealed Rect.

Return A handle to the region defining the FocusBox.

HideSelf()

Purpose A FocusBox is being hidden. This is an override of [HideSelf\(\)](#) in [LPane](#).

LFocusBox

Refresh()

Purpose Invalidate the area occupied by a FocusBox so it will redraw during the next Update event. This is an override of [Refresh\(\)](#) in [LPane](#).

ShowSelf()

Purpose A FocusBox is being shown. This is an override of [ShowSelf\(\)](#) in [LPane](#).

LGlobalsContext

Overview	<code>LGlobalsContext</code> is a PowerPlant class that is a mix-in for any class which needs to save the A5 or A4 (global variables) context for use in a Toolbox callback.
Methods	The methods in this class are: <code>LGlobalsContext()</code> <code>~LGlobalsContext()</code> <code>GetGlobals()</code>
Data Members	The data members in this class are: <code>mSavedGlobals</code>
Operation	This class is defined as a null class for PowerPC or 68K CFM builds. Designed to be used in concert with <code>StSetupGlobals</code> .
Source files	(Networking Classes) <code>UCallbackUtils.h</code> <code>UCallbackUtils.cp</code>
See also	<code>LMacTCPDNSOperation</code> <code>LMacTCPUDPEndpoint</code> <code>StMacTCPOperation</code> <code>StMacTCPUDPOperation</code> <code>StOpenTptOperation</code>

LGlobalsContext()

Purpose	The constructor creates the object, saving global variables context. Assumes that the global variables register (either A5 or A4) is correct at this time. On PowerPC or 68K CFM, it does nothing.
Access	Public

LGlobalsContext

Prototype	<code>inline LGlobalsContext();</code>
Parameters	None

~LGlobalsContext()

Purpose	The destructor destroys the object.
Access	Public
Prototype	<code>inline ~LGlobalsContext();</code>

GetGlobals()

Purpose	Returns the value of mSavedGlobals .
Access	Public
Prototype	<code>inline SInt32 GetGlobals();</code>
Parameters	None
Return	The value of mSavedGlobals .

mSavedGlobals

Purpose	The saved globals in A5 or A4.
Access	Private
Prototype	<code>SInt32 mSavedGlobals;</code>

LGrafPortView

Overview LGrafPortView is a PowerPlant class that is used as an alternative to LWindow in non-PowerPlant applications.

LGrafPortView can be useful in application frameworks and program extensions such as XCMDs.

Methods The methods in this class are:

LGrafPortView()	~LGrafPortView()
Activate()	ApplyForeAndBackColors()
ClickInContent()	CreateGrafPortView()
Deactivate()	DispatchCommand()
DoIdle()	DoKeyPress()
Draw()	DrawSelf()
EstablishPort()	GetMacPort()
InitGrafPortView()	InvalPortRect()
InvalPortRgn()	SetForeAndBackColors()
UpdatePort()	ValidPortRect()
ValidPortRgn()	

Data Members The data members in this class are:

mGrafPtr	mForeColor
mBackColor	

Operation When you create a GrafPortView, you should set the current port to the Toolbox window (or other valid GrafPort) in which you want to place the GrafPortView. Then call
LGrafPortView::CreateGrafPortView() to create and initialize the GrafPortView from a PPob resource.

Source files (Pane Classes)

LGrafPortView.h

LGrafPortView

LGrafPortView.cpp

See also

LGrafPortView()

Purpose	<p>LGrafPortView has four constructor methods. The first is the default constructor. The second constructor is a parameter constructor that copies <code>inGrafPtr</code> to <code>mGrafPtr</code> and then initializes the view.</p> <p>The third constructor creates a <code>LGrafPortView</code> object from data in the <code>SViewInfo</code> structure and sets the pane states. Signals if <code>mGrafPtr</code> is <code>nil</code>.</p> <p>The last constructor creates a <code>LGrafPortView</code> object from a stream (usually from a <code>PPob</code> resource), and sets the pane states. Signals if <code>mGrafPtr</code> is <code>nil</code>.</p>
Access	Public
Prototype	<pre>LGrafPortView(); LGrafPortView(GrafPtr inGrafPtr); LGrafPortView(const SPaneInfo& inPaneInfo, const SViewInfo &inViewInfo) : LView(inPaneInfo, inViewInfo); LGrafPortView(LStream *inStream):LView(inStream);</pre>
Parameters	None
Return	No return value for a constructor

~LGrafPortView()

Purpose	The destructor destroys the <code>LGrafPortView</code> object.
Access	Public
Prototype	<pre>virtual ~LGrafPortView();</pre>

Parameters	None
Return	No return value for a destructor

Activate()

Purpose	Checks the pane state to see if the GrafPortView is off. If so, sets it on and activates the window. If this Window can contain the Target, and it is not already OnDuty, restore the Target to what it was when the Window was last active.
Access	Public, Virtual
Prototype	<code>void Activate();</code>
Parameters	None
Return	None

ApplyForeAndBackColors()

Purpose	Uses Mac OS toolbox calls to set the foreground and background colors of the current port.
Access	Protected, Virtual
Prototype	<code>void ApplyForeAndBackColors() const;</code>
Parameters	None
Return	None
Remarks	The GrafPortView or one of its SubPanels must already be focused.

ClickInContent()

Purpose	Called when the mouse click is in the content part of a GrafPortView object. If it is enabled, sets up the extended event record, calls the object's Click method.		
Access	Public, Virtual		
Prototype	<pre>void ClickInContent(const EventRecord& inMacEvent);</pre>		
Parameters	<pre>const Even- tRecord&</pre>	<pre>inMacEvent</pre>	The Mac OS event record
Return	None		

CreateGrafPortView()

Purpose	Return a new GrafPortView object initialized from a PPob resource.		
Access	Public		
Prototype	<pre>static GrafPortView* CreateGrafPortView(ResIDT inGrafPortViewID, LCommander* inSuperCommander);</pre>		
Parameters	<pre>ResIDT</pre>	<pre>inGrafPortViewID</pre>	ID in PPob resource
	<pre>LCommander*</pre>	<pre>inSuperCommander</pre>	Pointer to the Super-Commander
Return	Pointer to a GrafPortView object.		
Remarks	GrafPortView is put inside the current GrafPort.		

Deactivate()

Purpose	Deactivates a GrafPortView object, and, if it was OnDuty, switches the Target to the SuperCommander.
Access	Public, Virtual
Prototype	<code>virtual void Deactivate();</code>
Parameters	None
Return	None

DispatchCommand()

Purpose	Handle a Command for a GrafPortView by getting the target and passing the command to it.		
Access	Public, Virtual		
Prototype	<code>void DispatchCommand(CommandT inCommand, void* ioParam);</code>		
Parameters	CommandT	inCommand	The command
	void *	ioParam	Misc. i/o parameter
Return	None		

DoIdle()

Purpose	Saves the port state and sends a message to the periodical handler to idle.
Access	Public, Virtual
Prototype	<code>void DoIdle(</code>

LGrafPortView

```
const EventRecord& inMacEvent);
```

Parameters

<code>const EventRecord& inMacEvent</code>	Address of Mac OS Event record
--	--------------------------------

Return	None
--------	------

DoKeyPress()

Purpose	Called when the GrafPortView gets a KeyDown event. If the LCommander's sTarget data member indicates that this object is the target, process the event.
---------	---

Access	Public, Virtual
--------	-----------------

Prototype	<pre>void DoKeyPress(const EventRecord& inMacEvent);</pre>
-----------	---

Parameters

<code>const EventRecord& inMacEvent</code>	Address of Mac OS Event record
--	--------------------------------

Return	None
--------	------

Draw()

Purpose	Draw a GrafPortView using LView::Draw(). inSuperDrawRgnH specifies, in port coordinates, the portion of the View's SuperView that needs to be drawn. Specify nil to draw the entire view.
---------	---

Access	Protected, Virtual
--------	--------------------

Prototype	<pre>void Draw(RgnHandle inSuperDrawRgnH);</pre>
-----------	---

Parameters

<code>RgnHandle inSuperDrawRgnH</code>	Mac OS region handle
--	----------------------

Return	None
Remarks	Overrides LPane::Draw() to deal with special non-window condition and then cycle through Subpanes. Rather than overriding this, you should override LGrafPortView::DrawSelf(), and get called from here.

DrawSelf()

Purpose	Draws the contents of the GrafPortView object. Currently erases the frame as well.
Access	Protected, Virtual
Prototype	<code>virtual void DrawSelf();</code>
Parameters	None
Return	None

EstablishPort()

Purpose	Make GrafPortView the current Port.
Access	Public, Virtual
Prototype	<code>Boolean EstablishPort();</code>
Parameters	None
Return	Always returns TRUE.
Remarks	If you call this function directly, you should call LView::OutOfFocus(nil), since changing the current port may invalidate the Focus.

LGrafPortView

GetMacPort()

Purpose	Return the Toolbox GrafPort associated with a LGrafPortView.
Access	Public, Virtual
Prototype	<code>GrafPtr GetMacPort() const;</code>
Parameters	None
Return	Mac OS GrafPtr

InitGrafPortView()

Purpose	Private initializer. Initialize the GrafPortView, assuming that mGrafPtr is already initialized, set the pane states, and initialize all mForeColor fields to black and all mBackColor fields to white.
Access	Private
Prototype	<code>void InitGrafPortView();</code>
Parameters	None
Return	None

InvalPortRect()

Purpose	Make the Mac OS Toolbox issue an update event, which will initiate a redraw of the rectangle.
Access	Public, Virtual
Prototype	<code>void InvalPortRect(const Rect *inRect);</code>
Parameters	<code>const Rect *inRect</code> Pointer to the rectangle to redraw.

Return	None
Remarks	Use this instead of the Mac OS Toolbox routine <code>InvalRect()</code> (see Pane validation for more information).

InvalPortRgn()

Purpose	Mac OS			
Access	Public			
Prototype	<pre>virtual void InvalPortRgn(RgnHandle inRgnH);</pre>			
Parameters	<table><tr><td>RgnHandle</td><td>inRgnH</td><td>Mac OS handle to the region to re-draw.</td></tr></table>	RgnHandle	inRgnH	Mac OS handle to the region to re-draw.
RgnHandle	inRgnH	Mac OS handle to the region to re-draw.		
Return	None			
Remarks	Use this instead of the Mac OS Toolbox routine <code>InvalRgn()</code> (see Pane validation for more information).			

SetForeAndBackColors()

Purpose	Specify the foreground and/or background colors of a GrafPortView.		
Access	Public, Virtual		
Prototype	<pre>void SetForeAndBackColors(const RGBColor *inForeColor, const RGBColor *inBackColor);</pre>		
Parameters			
	<code>const RGBColor</code>	<code>*inForeColor</code>	Pointer to the foreground color
	<code>const RGBColor</code>	<code>*inBackColor</code>	Pointer to the background color

LGrafPortView

Return	None
Remarks	Specify <code>nil</code> for <code>inForeColor</code> and/or <code>inBackColor</code> to leave that color trait unchanged.

UpdatePort()

Purpose	Redraw invalidated area of the GrafPortView.
Access	Public, Virtual
Prototype	<code>void UpdatePort();</code>
Parameters	None
Return	None

ValidPortRect()

Purpose	Tells the Mac OS Toolbox that the selected rectangle does not need to be redrawn.
Access	Public, Virtual
Prototype	<code>void ValidPortRect(const Rect *inRect);</code>
Parameters	<code>const Rect *inRect</code> Pointer to the rectangle.
Return	None
Remarks	Use instead of the Mac OS Toolbox routine <code>InvalRect()</code> (see Pane validation for more information).

ValidPortRgn()

Purpose	Tells the Mac OS Toolbox that the selected region does not need to be redrawn.		
Access	Public, Virtual		
Prototype	<pre>void ValidPortRgn(RgnHandle inRgnH);</pre>		
Parameters	RgnHandle	inRgnH	Mac OS Handle to the region.
Return	None		
Remarks	Use instead of the Mac OS Toolbox routine <code>InvalidRgn()</code> (see Pane validation for more information).		

mGrafPtr

Purpose	Maintains a pointer to the current GrafPort.
Access	Protected
Prototype	<code>GrafPtr mGrafPtr;</code>

mForeColor

Purpose	Maintains the current foreground color setting.
Access	Protected
Prototype	<code>RGBColor mForeColor;</code>

LGrafPortView

mBackColor

Purpose	Maintains the current background color setting.
Access	Protected
Prototype	<code>RGBColor mBackColor;</code>

LGroupBox

Overview	LGroupBox is a PowerPlant class that is usually used for visually grouping user interface elements, using a line drawn around them.						
Methods	<p>The methods in this class are:</p> <table><tr><td>LGroupBox()</td><td>~LGroupBox()</td></tr><tr><td>DrawSelf()</td><td>DrawText()</td></tr><tr><td>DrawBorder()</td><td>CalcTextBoxFrame()</td></tr></table>	LGroupBox()	~LGroupBox()	DrawSelf()	DrawText()	DrawBorder()	CalcTextBoxFrame()
LGroupBox()	~LGroupBox()						
DrawSelf()	DrawText()						
DrawBorder()	CalcTextBoxFrame()						
Data Members	<p>The data members in this class are:</p> <p>mFrameColor</p>						
Operation	<p>This class derives from LCaption. The text set with DrawText() is the title of the group. This object draws a box with the specified coordinates.</p> <p>The top of the object's frame does not coincide with the top line drawn for the group box. The other panes that are visually within the group box do not have a programming relationship to the group box in the hierarchy. They are simply visually grouped, without any program control between the group box and the other panes.</p>						
Source files	<p>(Pane Classes)</p> <p>LGroupBox.h</p> <p>LGroupBox.cp</p>						
See also	LCaption						

LGroupBox()

Purpose	The constructor creates the object from the passed-in parameters.
Access	Public
Prototype	<pre>LGroupBox () ; LGroupBox(const LGroupBox &inGroupBox) ;</pre>

LGroupBox

```
LGroupBox( const SPaneInfo &inPaneInfo,  
Str255 inString,  
ResIDT inTextTraitsID );  
LGroupBox( LStream *inStream );
```

Parameters The parameters for these constructors are:

const LGroupBox&	inGroupBox	A reference to the group box object to be copied.
const SPaneInfo&	inPaneInfo	A reference to the SuperPane to draw the group box upon.
Str255	inString	A pointer to the group box text.
ResIDT	inTextTraitsID	The text traits for the group box text.
LStream*	inStream	A pointer to a stream object that contains the information to create the LCaption object.

~LGroupBox()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual LGroupBox();</code>
Parameters	None

DrawSelf()

Purpose	This method draws the group box. It is drawn in two stages. First the frame rectangle is drawn, minus the area where the title text is drawn. Second, the title text is drawn. In order to draw without any flashing, we find the frame of the text box and clip it out of the drawing region.
Access	Virtual, Protected

Prototype	<code>virtual void DrawSelf();</code>
Parameters	None
Return	None

DrawText()

Purpose	This method draws the group box text given the chosen Rect coordinates.		
Access	Virtual, Protected		
Prototype	<code>virtual void DrawText(const Rect &inRect);</code>		
Parameters	The parameter for this method is:		
	<code>const Rect&</code>	<code>inRect</code>	A reference to the Rect to draw the group box text in.
Return	None		

DrawBorder()

Purpose	This method draws the group box frame given the chosen coordinates.		
Access	Virtual, Protected		
Prototype	<code>virtual void DrawBorder(const Rect &inRect);</code>		
Parameters	The parameter for this method is:		
	<code>const Rect&</code>	<code>inRect</code>	A reference to the Rect to draw the group box upon.
Return	None		

LGroupBox

CalcTextBoxFrame()

Purpose	This method decides where the text will be drawn, but does not draw the text.		
Access	Virtual, Protected		
Prototype	<code>virtual void CalcTextBoxFrame(Rect &outRect);</code>		
Parameters	The parameter for this method is:		
	<code>const Rect&</code>	<code>outRect</code>	A reference to the <code>Rect</code> to draw the group box text within.
Return	None		

mFrameColor

Purpose	This data member holds the color to draw the group box frame in.
Access	Protected
Prototype	<code>RGBColor mFrameColor;</code>

LGrowZone

Overview LGrowZone is a PowerPlant class to use for managing any low memory situations that occur during your program's execution. For general information about the Mac OS GrowZone concepts, refer to *Inside Macintosh: Memory*, published by Addison-Wesley.

LGrowZone's implementation provides two levels of functionality. First, it provides the classic behavior of allocating a user-specified emergency reserve buffer of memory upon creation of the GrowZone. The OS will then notify your application when to draw from this emergency reserve in low-memory situations. Second, LGrowZone can work with other PowerPlant objects to ask them to free memory when it becomes scarce.

Methods The methods in the LGrowZone class are:

<u>LGrowZone()</u>	<u>~LGrowZone()</u>
<u>AskListenersToFree()</u>	<u>DoGrowZone()</u>
<u>GetGrowZone()</u>	<u>GiveWarning()</u>
<u>GrowZoneCallBack()</u>	<u>MemoryIsLow()</u>
<u>SpendTime()</u>	<u>UseLocalReserve()</u>

Data Members The data members in the LGrowZone class are:

<u>sGrowZone</u>	<u>sGrowZoneUPP</u>
<u>mLocalReserve</u>	<u>mReserveSize</u>
<u>mGiveWarning</u>	

Operation Near the start of your application code you should create a single instance of LGrowZone, as shown here:

```
main() {
    // do init stuff here

    // create a memory reserve of 32K bytes
    new LGrowZone( 32 * 1024 );

    // the rest of your code goes here...
}
```

LGrowZone inherits from [LBroadcaster](#). Objects which are able to free memory when it is becoming depleted should be Listeners, and attach themselves to the LGrowZone object so they will be notified when memory is low. The following code shows how to add your objects as listeners for this low-memory notification. This code attaches myObj as a Listener to the LGrowZone messages.

```
LGrowZone::GetGrowZone() ->AddListener(myObj);
```

When memory is needed, you'll want myObj to be able to respond to a [ListenToMessage\(\)](#) call. The call to ListenToMessage() will occur with a msg_GrowZone message, and a pointer to the number of bytes needed. You should add code similar to the following to your object's class definition to handle the ListenToMessage() method, that you inherit from [LBroadcaster](#).

```
MyClass::ListenToMessage(MessageT inMessage, void *ioParam)
{
    if (inMessage == msg_GrowZone) {

        // Memory is low, free our cache
        SInt32 freedBytes =
            ::GetHandleSize(myCache);
            ::DisposeHandle(myCache);
        myCache = nil;

        // Pass back bytes freed
        *(SInt32*)ioParam = freedBytes;
    }
}
```

For example, suppose MyClass stores the myCache Handle with data generated from a lengthy calculation. If memory gets low, it can dispose of the cached data, meaning that it will have to recalculate the data if it needs it later. This is a common trade-off: memory versus speed.

This class also inherits from [LPeriodical](#). LGrowZone's constructor installs the object in the Periodical Repeater queue so that its [SpendTime\(\)](#) function gets called each time through the main event loop. This is required so that the LGrowZone object can monitor memory usage and can respond appropriately, perhaps by displaying a warning to the user.

Source files (Support Classes)

LGrowZone.h

LGrowZone.cp

See also [LBroadcaster](#)

[LPeriodical](#)

LGrowZone()

Purpose The constructor allocates a memory reserve of the passed-in size, and a GrowZone function for the current process is then registered with the operating system.

The constructor also installs this LGrowZone object in the Periodical Repeater queue defined by [LPeriodical](#), so that the [SpendTime\(\)](#) method will get called each time through the application's main event loop.

Access Public

Prototype `LGrowZone(Size inReserveSize);`

Parameters This method has the following parameter:

Size	inReserveSize	Indicates the size of the memory block that you wish to reserve in your heap area. The unit of this value is in bytes.
------	---------------	--

Return None

~LGrowZone()

Purpose The destructor destroys the LGrowZone object. This includes deallocating the previously-allocated memory reserve and deinstalling the GrowZone function.

Access Public

LGrowZone

Prototype	<code>virtual ~LGrowZone();</code>
Parameters	None
Return	None

AskListenersToFree()

Purpose	This method broadcasts a message to listeners that they should free up some memory if possible. In order for an object to be a listener, it must have registered with the LGrowZone object using AddListener() .
Access	Virtual, Public
Prototype	<code>virtual SInt32 AskListenersToFree(Size inBytesNeeded);</code>
Parameters	This method has the following parameter: Size inBytesNeeded The number of bytes needed.
Return	SInt32 containing the number of bytes that were freed as a result of calling this method. If no memory could be freed, then it returns 0.
Remarks	<ul style="list-style-type: none">• This method sends a <code>msg_GrowZone</code> message (via the ListenToMessage() call) to each object that has registered as a Listener, with a parameter that is a pointer to the number of bytes needed. On exit, objects that are Listeners should set this pointer equal to the number of bytes that they deallocated.• A running count of the bytes needed is updated, which could be negative if some Listener frees more than we need.

DoGrowZone()

Purpose	This method is called by the GrowZone function installed by the LGrowZone() constructor when the operating system needs more memory.
---------	--

Access	Virtual, Protected
Prototype	<code>virtual DoGrowZone(Size inBytesNeeded);</code>
Parameters	This method has the following parameter: <div> Size inBytesNeeded The number of bytes needed. </div>
Return	SInt32 containing the number of bytes that were freed as a result of calling this method. If no memory could be freed, then it returns 0.
Remarks	This method is not normally called directly in your normal application code.

GetGrowZone()

Purpose	This method is an accessor, designed to return a pointer to the LGrowZone object.
Access	Inline, Public, Static
Prototype	<code>static LGrowZone* GetGrowZone();</code>
Parameters	None
Return	A pointer to the LGrowZone object in your application.
Remarks	You might need this method if you are going to call AddListener() . This would be done to register an object for future notification of low-memory conditions.

GiveWarning()

Purpose	This method sets the mGiveWarning data member to true.
Access	Public, Inline
Prototype	<code>void GiveWarning();</code>
Parameters	None

LGrowZone

Return	None
Remarks	You might want to call this method if you wish to have the PowerPlant framework display a generic alert to the user when memory is depleted.

GrowZoneCallBack()

Purpose	This method is called when the System requires more memory.
Access	Protected, Static
Prototype	<code>static pascal SInt32 GrowZoneCallBack(Size inBytesNeeded);</code>
Parameters	This method has the following parameter: Size inBytesNeeded The number of bytes needed.
Return	SInt32 containing the number of bytes that were freed by your program.
Remarks	This is the GrowZone function registered with the System. It sets up the Motorola 68K A5 world so we can access globals, then calls a virtual function for the LGrowZone class.

MemoryIsLow()

Purpose	This method provides a way of determining whether the memory reserve allocated by the LGrowZone() constructor has been utilized.
Access	Virtual, Public
Prototype	<code>virtual Boolean MemoryIsLow();</code>
Parameters	None
Return	Return true if the LGrowZone() memory reserve has been used, false otherwise.

Remarks	Objects in your program can call this method if they wish to behave differently under low memory situations. For example, a program could disable the "New" and "Open" commands to prevent new Documents from being created when memory is low.
---------	---

SpendTime()

Purpose	This method performs periodic maintenance for the LGrowZone object, including
Access	Virtual, Public
Prototype	<code>virtual void SpendTime(const EventRecord& inMacEvent);</code>

Parameters	This method has the following parameter:
------------	--

<code>const EventRecord&</code>	<code>inMacEvent</code>	a struct containing information about the Mac OS event that has just occurred
---	-------------------------	---

An EventRecord reference is the parameter to this method, but it is not utilized by this override.

Return	None
--------	------

Remarks	<ul style="list-style-type: none"> • This method is an override of the SpendTime() base class method from LPeriodical(). • This method attempts to reallocate the LGrowZone() memory reserve if necessary, and (optionally) warn the user if memory is getting low. • This method is called each time through the main event loop. • In order for the framework to provide a warning, the GiveWarning() method must be called first.
---------	--

LGrowZone

UseLocalReserve()

Purpose	This method empties the reserve that was allocated during creation of the LGrowZone object.		
Access	Virtual, Protected		
Prototype	<code>virtual SInt32 UseLocalReserve(Size inBytesNeeded);</code>		
Parameters	This method has the following parameter: <table><tr><td><code>Size inBytesNeeded</code></td><td>The number of bytes needed.</td></tr></table>	<code>Size inBytesNeeded</code>	The number of bytes needed.
<code>Size inBytesNeeded</code>	The number of bytes needed.		
Return	SInt32 containing the number of bytes that were freed.		
Remarks	<ul style="list-style-type: none">• Calling this method is a last chance desperate attempt to free enough memory to proceed.• If the memory reserve for LGrowZone exists, it will be freed. Also, mGiveWarning is set to true. The user will receive warning alerts from the framework if memory becomes low again.		

sGrowZone

Purpose	This is a pointer to the LGrowZone object. It is accessible using the GetGrowZone() accessor method.
Access	Static, Protected
Prototype	<code>static LGrowZone* sGrowZone;</code>
Remarks	You do not normally need to directly access this data member, and should use the GetGrowZone() accessor method instead.

sGrowZoneUPP

Purpose	This is a Universal Procedure Pointer (UPP) used for pointing to the GrowZone procedure.
---------	--

Access	Static, Protected
Prototype	<code>static GrowZoneUPP sGrowZoneUPP;</code>

mLocalReserve

Purpose	This member is a Handle to the memory reserve allocated by the LGrowZone() constructor.
Access	Protected
Prototype	<code>Handle mLocalReserve;</code>

mReserveSize

Purpose	This member is a count of the size (in bytes) of the memory reserve area allocated by the LGrowZone() constructor.
Access	Protected
Prototype	<code>Size mReserveSize;</code>

mGiveWarning

Purpose	This data member is set to true if you want the PowerPlant framework to provide a default warning alert to the user under low-memory conditions.
Access	Protected
Prototype	<code>Boolean GiveWarning;</code>
Remarks	The default for this data member is false when the LGrowZone() constructor is invoked.

LGWorld

Description	LGWorld is a PowerPlant class that is used for creating GWorlds for offscreen drawing.	
Methods	The methods in this class are: LGWorld() BeginDrawing() EndDrawing() GetMacGWorld() ~LGWorld() CopyImage() GetBounds() SetBounds()	
Data Members	The data members in this class are: mMacGWorld mSavePort mBounds mSaveDevice	
Source files	(Pane Classes) UGWorld.h UGWorld.cp	
Ancestors	You can use LGWorld to help manage the creation and use of offscreen drawing areas. Refer to <i>The PowerPlant Book</i> for an example on how to use this class.	

LGWorld()

Purpose	The constructor creates an object from the passed-in parameters.
Access	Public
Prototype	<pre>LGWorld(const Rect &inBounds, SInt16 inPixelDepth = 0, GWorldFlags inFlags = 0 , CTabHandle inCTableH = nil, GDHandle inGDeviceH = nil);</pre>

LGWorld

Parameters	The parameters for this constructor are:		
	<code>const Rect&</code>	<code>inBounds</code>	The bounds Rect for the drawing area.
	<code>SInt16</code>	<code>inPixelDepth</code>	The pixel depth, the default is 0.
	<code>GWorldFlags</code>	<code>inFlags</code>	The flags, the default is 0.
	<code>CTabHandle</code>	<code>inCTableH</code>	Handle to a color table, default is <code>nil</code> .
	<code>GDHandle</code>	<code>inGDeviceH</code>	Handle to the GDevice, default is <code>nil</code> .

~LGWorld()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LGWorld();</code>

BeginDrawing()

Purpose	Sets the current port to the offscreen GWorld and locks its pixels. Every <code>BeginDrawing</code> call must be balanced by a corresponding EndDrawing() call.
Access	Public
Prototype	<code>Boolean BeginDrawing();</code>
Parameters	None
Return	Returns <code>false</code> if the offscreen pixels can't be locked. This will happen if the <code>pixPurge</code> flag was set and the pixels were purged from memory. If so, you might want to try to reallocate the pixels by calling the Toolbox routine <code>UpdateGWorld()</code> .

CopyImage()

Purpose	Copies an image from the offscreen GWorld to the specified port.		
Access	Public		
Prototype	<pre>void CopyImage(GrafPtr inDestPort, const Rect &inDestRect, SInt16 inXferMode, RgnHandle inMaskRgn);</pre>		
Parameters	The parameters for this constructor are:		
	GrafPtr	inDestPort	The destination port.
	const Rect&	inDestRect	The destination port Rect.
	SInt16	inXferMode	The transfer mode, see QuickDraw.h.
	RgnHandle	inMaskRgn	The handle to the mask region.
Return	None		
Remarks	<p>You will want to make sure that the foreground color is black and the background color is white before calling this function. CopyBits() can be unreliable if this is not the case. This isn't done automatically here so that this routine to be fast (that's why you usually use offscreen drawing).</p>		

EndDrawing()

Purpose	Unlocks the GWorld's pixels and restores the current port to what it was before BeginDrawing() was called. Every EndDrawing call must be preceded by a corresponding BeginDrawing() call.
Access	Public
Prototype	<pre>void EndDrawing();</pre>

LGWorld

Parameters	None
Return	None

GetBounds()

Purpose	Returns the value of the mBounds data member in the passed-in parameter.		
Access	Inline, Public		
Prototype	<code>void GetBounds(Rect &outBounds);</code>		
Parameters	The parameters for this method is:		
	<code>Rect&</code>	<code>outBounds</code>	The bounds Rect.
Return	None		

GetMacGWorld()

Purpose	This method retrieves a pointer to a Mac OS GWorld.
Access	Inline, Public
Prototype	<code>GWorldPtr GetMacGWorld();</code>
Parameters	None
Return	Pointer to a Mac OS GWorld.

SetBounds()

Purpose	Used to adjust the bounds of an offscreen world.
Access	Public

Prototype	<code>void SetBounds(const Rect &inBounds);</code>		
Parameters	The parameters for this method is:		
	<code>Rect&</code>	<code>outBounds</code>	The bounds Rect.
Return	None		

mMacGWorld

Purpose	Storage for the Mac OS Gworld.
Access	Protected
Prototype	<code>GWorldPtr mMacGWorld;</code>

mBounds

Purpose	Storage for the bounds Rect.
Access	Protected
Prototype	<code>Rect mBounds;</code>

mSavePort

Purpose	Storage to save the port.
Access	Protected
Prototype	<code>CGrafPtr mSavePort;</code>

mSaveDevice

Purpose Storage to save the handle to the device.

Access Protected

Prototype GDHandle mSaveDevice;

LHandleStream

Overview	LHandleStream is a PowerPlant class that is used for doing LStream operations with data blocks that are relocatable, such as handles.	
Methods	The methods in this class are: LHandleStream() DetachDataHandle() GetDataHandle() SetDataHandle() ~LHandleStream() GetBytes() PutBytes() SetLength()	
Data Members	The data members in this class are: mDataH	
Operation	General usage for LHandleStream is similar to that of LDataStream and LFileStream .	
Source files	File & Stream Classes) LHandleStream.h LHandleStream.cp	
Ancestors	LStream	

LHandleStream()

Purpose	The constructor creates objects.		
Access	Public		
Prototype	<pre>LHandleStream(); LHandleStream(Handle inHandle);</pre>		
Parameters	The parameter for the constructors is:		
	Handle	inHandle	The handle to create an object from.

~LHandleStream()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LHandleStream();</code>

DetachDataHandle()

Purpose	Dissociate the data Handle from a HandleStream. This creates a new, empty data Handle and passes back the existing Handle. Caller assumes ownership of the Handle.
Access	Public
Prototype	<code>Handle DetachDataHandle();</code>
Parameters	None
Return	A new, empty data handle.

GetBytes()

Purpose	Read bytes from a HandleStream to a buffer.
Access	Virtual, Public
Prototype	<code>virtual ExceptionCode GetBytes(void *outBuffer, SInt32 &ioByteCount);</code>
Parameters	The parameters for this method are:

void*	outBuffer	The pointer to a buffer to read into.
SInt32 &	ioByteCount	The size of the buffer on input, the number of bytes read on output.

Return	Returns an error code and passes back the number of bytes actually read, which may be less than the number requested if an error occurred. <ul style="list-style-type: none"> • readErr—Attempt to read past the end of the HandleStream
--------	---

GetDataHandle()

Purpose	Retrieves the value of mDataH .
Access	Public
Prototype	<code>Handle GetDataHandle();</code>
Parameters	None
Return	Returns the value of mDataH .

PutBytes()

Purpose	Write bytes from a buffer to a HandleStream. Grows the data Handle if necessary.
Access	Public
Prototype	<code>virtual ExceptionCode PutBytes(const void *inBuffer, SInt32 &ioByteCount);</code>
Parameters	The parameters for this method are:

<code>const void*</code>	<code>inBuffer</code>	The pointer to a buffer to write into.
<code>SInt32&</code>	<code>ioByteCount</code>	The size of the buffer on input, the number of bytes written on output.

LHandleStream

Return	Returns an error code and passes back the number of bytes actually written, which may be less than the number requested if an error occurred. <ul style="list-style-type: none">• memFullErr—Growing Handle failed when trying to write past the current end of the Stream
--------	--

SetDataHandle()

Purpose	Specify a Handle to use as the basis for a HandleStream. Class assumes ownership of the input Handle and destroys the existing data Handle. Call DetachDataHandle() beforehand if you wish to preserve the existing data Handle.			
Access	Public			
Prototype	<code>void SetDataHandle(Handle inHandle);</code>			
Parameters	The parameter for this method is: <table><tr><td>Handle</td><td>inHandle</td><td>The handle to set for the object.</td></tr></table>	Handle	inHandle	The handle to set for the object.
Handle	inHandle	The handle to set for the object.		
Return	None			

SetLength()

Purpose	Set the length, in bytes, of the HandleStream.		
Access	Public		
Prototype	virtual void SetLength(SInt32 inLength);		
Parameters	The parameter for this method is:		
	SInt32	inLength	The length of the stream.
Return	None		

mDataH

Purpose	The data handle.
Access	Protected
Prototype	Handle mDataH;

LIconPane

Overview	LIconPane is a PowerPlant class that is used for drawing a single icon from an icon family.
Methods	The methods in this class are: LIconPane() DrawSelf() ~LIconPane() SetIconID()
Data Members	The data members in this class are: mIconID
Operation	Because this class uses the Mac OS icon-handling routines, an LIconPane object draws the member of the icon family that best fits the color settings and pixel depth of the current device.
Source files	(Pane Classes) LIconPane.h LIconPane.cp
Ancestors	LAttachable LPane

LIconPane()

Purpose	The constructors create objects from the passed-in parameters.
Access	Public
Prototype	<pre>LIconPane () ; LIconPane(const SPaneInfo &inPaneInfo, ResIDT inIconID) ; LIconPane(LStream *inStream) ;</pre>
Parameters	These constructors have the following parameters:

LIconPane

const SPaneInfo&	inPaneInfo	The reference to the Superpane.
ResIDT	inIconID	The resource ID for the icon.
LStream*	inStream	The stream to construct an object from.

~LIconPane()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LIconPane();</code>

DrawSelf()

Purpose	Draws the icon. This is an override of DrawSelf() in LPane .
Access	Public
Prototype	<code>virtual void DrawSelf();</code>
Parameters	None
Return	None

SetIconID()

Purpose	Set the icon.
Access	Public
Prototype	<code>void SetIconID(ResIDT inIconID);</code>
Parameters	These constructors have the following parameters:

	ResIDT	inIconID	The resource ID for the icon.
Return	None		

mIconID

Purpose	Storage for the icon resource ID.
Access	Protected
Prototype	ResIDT mIconID;

InternetAddress

Overview LInternetAddress is a PowerPlant class that is used for wrapping the IP addresses that are used internally by the Internet to describe another computer's location.

Methods The methods in this class are:

LInternetAddress()	~LInternetAddress()
Clone()	GetDNSAddress()
GetDNSDescriptor()	GetHostPort()
GetIPAddress()	GetIPAddress()
GetIPDescriptor()	InternalLookupAddress()
InternalLookupName()	MakeOTDNSAddress()
MakeOTIPAddress()	SetDNSAddress()
SetHostPort()	SetIPAddress()

Data Members The data members in this class are:

mIPAddress	mDNSAddress
mHostPort	

Operation DNS names (like "ftp.metrowerks.com") must be mapped to IP addresses (127.0.0.1) before connections are established. This object will handle such conversions automatically.

Source files (Networking Classes)

LInternetAddress.h

LInternetAddress.cp

LInternetAddress()

Purpose The constructor creates the object.

LInternetAddress

Access	Public																
Prototype	<pre> LInternetAddress(UInt32 inHostAddress, UInt16 inHostPort); LInternetAddress(ConstStringPtr inHostAddress, UInt16 inHostPort, Boolean inLookupNow); LInternetAddress(UInt16 inHostPort); LInternetAddress(const LInternetAddress& inOriginal); </pre>																
Parameters	The parameters for this method are: <table> <tr> <td>UInt32</td><td>inHostAddress</td><td>The host address.</td></tr> <tr> <td>UInt16</td><td>inHostPort</td><td>The host port.</td></tr> <tr> <td>ConstStringPtr</td><td>inHostAddress</td><td>The host address.</td></tr> <tr> <td>Boolean</td><td>inLookupNow</td><td>Whether to do an immediate lookup.</td></tr> <tr> <td>LInternetAddress&</td><td>inOriginal</td><td>The address to copy.</td></tr> </table>		UInt32	inHostAddress	The host address.	UInt16	inHostPort	The host port.	ConstStringPtr	inHostAddress	The host address.	Boolean	inLookupNow	Whether to do an immediate lookup.	LInternetAddress&	inOriginal	The address to copy.
UInt32	inHostAddress	The host address.															
UInt16	inHostPort	The host port.															
ConstStringPtr	inHostAddress	The host address.															
Boolean	inLookupNow	Whether to do an immediate lookup.															
LInternetAddress&	inOriginal	The address to copy.															

~LInternetAddress()

Purpose	The destructor destroys the LInternetAddress object.	
Access	Virtual, Public	
Prototype	<pre>virtual ~LInternetAddress();</pre>	

Clone()

Purpose	Create a new copy of the address using the new operator from this one.
Access	Virtual, Public
Prototype	<code>virtual LInternetAddress* Clone();</code>
Parameters	None
Return	A new copy of the internet address.

GetDNSAddress()

Purpose	Retrieve the DNS address string.			
Access	Virtual, Public			
Prototype	<code>virtual StringPtr GetDNSAddress(Str255outDescriptor);</code>			
Parameters	<div>The parameters for this method are:<table><tr><td>Str255</td><td>outDescriptor</td><td>The address string.</td></tr></table></div>	Str255	outDescriptor	The address string.
Str255	outDescriptor	The address string.		
Return	Returns the address string.			

GetDNSDescriptor()

Purpose	Return the name of the specified host (and port number if available) Use GetDNSAddress() if you don't care if we have tried to do a DNS lookup.
Access	Virtual, Public
Prototype	<code>StringPtr GetDNSDescriptor(Str255outDescriptor, Boolean withPort);</code>

LInternetAddress

Parameters	The parameters for this method are:		
	Str255	outDescriptor	The address string.
	Boolean	withPort	Whether to add the port to the string or not.
Return	A pointer to the descriptor.		

GetHostPort()

Purpose	Returns the value of mHostPort .
Access	Virtual, Public
Prototype	<code>virtual UInt16 GetHostPort();</code>
Parameters	None
Return	The value of mHostPort .

GetIPAddress()

Purpose	Return the 32-bit IP address of the host.
Access	Virtual, Public
Prototype	<code>virtual UInt32 GetIPAddress();</code>
Parameters	None
Return	The 32-bit address.

GetIPAddress()

Purpose	Return the 32-bit IP address of the host as well as dotted decimal format.	
Access	Virtual, Public	
Prototype	<code>virtual UInt32 GetIPAddress(Str255outDescriptor);</code>	
Parameters	The parameters for this method are:	
	Str255	outDescriptor The address string.
Return	The 32-bit address.	

GetIPDescriptor()

Purpose	Convert the IP address to dotted decimal format and return this string. If a port number is specified, optionally return that at the end of the string.	
Access	Virtual, Public	
Prototype	<code>virtual StringPtr GetIPDescriptor(Str255outDescriptor, Boolean withPort);</code>	
Parameters	The parameters for this method are:	
	Str255	outDescriptor The address string.
	Boolean	withPort Add the port to the string or not.
Return	A string pointer.	

InternalLookupAddress()

Purpose	Create mapper and lookup address from DNS name.
Access	Private
Prototype	<code>void InternalLookupAddress();</code>
Parameters	None
Return	None

InternalLookupName()

Purpose	Create mapper and lookup name from IP Address.
Access	Private
Prototype	<code>void InternalLookupName();</code>
Parameters	None
Return	None

MakeOTDNSAddress()

Purpose	Create an Open Transport (OT) DNS address.				
Access	Virtual, Public				
Prototype	<code>virtual void MakeOTDNSAddress(TNetbuf& outAddress);</code>				
Parameters	The parameters for this method are:				
	<table><tr><td>TNetbuf&</td><td>outAddress</td><td>The OT address.</td></tr></table>	TNetbuf&	outAddress	The OT address.	
TNetbuf&	outAddress	The OT address.			
Return	None				

MakeOTIPAddress()

Purpose	Make an Open Transport (OT) IP address.		
Access	Virtual, Public		
Prototype	<pre>virtual void MakeOTIPAddress(TNetbuf& outAddress);</pre>		
Parameters	The parameters for this method are:		
	TNetbuf&	outAddress	The OT address.
Return	None		

SetDNSAddress()

Purpose	Set the value of the mDNSAddress data member.		
Access	Virtual, Public		
Prototype	<pre>virtual void SetDNSAddress(ConstStringPtr inHostAddress);</pre>		
Parameters	The parameters for this method are:		
	ConstStringPtr	inHostAddress	The DNS address.
Return	None		

SetHostPort()

Purpose	Set the value of the mHostPort data member.
Access	Virtual, Public
Prototype	<pre>virtual void SetHostPort(UInt16 inHostPort);</pre>

LInternetAddress

Parameters The parameters for this method are:

UInt16	inHostPort	The host port.
--------	------------	----------------

Return None

SetIPAddress()

Purpose Set the value of the [mIPAddress](#) data member.

Access Virtual, Public

Prototype `virtual void SetIPAddress(UInt32 inHostAddress);`

Parameters The parameters for this method are:

UInt32	inHostAddress	The IP address.
--------	---------------	-----------------

Return None

mIPAddress

Purpose The IP address.

Access Protected

Prototype `UInt32 mIPAddress;`

mDNSAddress

Purpose The DNS address.

Access Protected

Prototype `LStr255 mDNSAddress;`

mHostPort

Purpose	The host port.
Access	Protected
Prototype	<code>UInt16 mHostPort;</code>

LIInternetMapper

Overview	LIInternetMapper encapsulates the services of a Domain Name System server (DNS) for remote computers.						
Methods	<p>The methods in this class are:</p> <table><tr><td>LIInternetMapper()</td><td>~LIInternetMapper()</td></tr><tr><td>AbortThreadOperation()</td><td>AddressToName()</td></tr><tr><td>GetLocalAddress()</td><td>NameToAddress()</td></tr></table>	LIInternetMapper()	~LIInternetMapper()	AbortThreadOperation()	AddressToName()	GetLocalAddress()	NameToAddress()
LIInternetMapper()	~LIInternetMapper()						
AbortThreadOperation()	AddressToName()						
GetLocalAddress()	NameToAddress()						
Data Members	There are no data members in this class.						
Operation	<p>You use an LIInternetMapper object to locate the address of another computer on the network. You can convert from a DNS address (like <code>www.metrowerks.com</code>) to an IP address (such as <code>127.0.0.1</code>) and vice-versa. All of these methods are pure virtual, forcing you to provide your own concrete implementations if you choose to not use <code>LMacTCPInetMapper</code> or <code>LOpenTptInetMapper</code>.</p> <p>NOTE: For most connection-oriented applications, it is not necessary to use the mapper interface. The LEndpoint interface can accept <code>LIInternetDNSAddress</code> objects in its <code>Connect</code> member function, which causes a name lookup to take place automatically.</p> <p>LIInternetMapper is an abstract base class (thus its constructor is declared protected). Use one of the predefined subclasses, such as <code>LMacTCPInetMapper</code> or <code>LOpenTptInetMapper</code>, or use the UNetworkFactory::CreateInternetMapper function to create the appropriate mapper for the system software that's installed on the user's machine.</p>						
Source files	<p>(Networking Classes)</p> <p><code>LIInternetMapper.h</code></p> <p><code>LIInternetMapper.cp</code></p>						
See also	<p>LEndpoint</p> <p>UNetworkFactory</p>						

LInternetMapper

LInternetMapper()

Purpose	The copy constructor creates the object. Don't use the default constructor.
Access	Private
Prototype	<code>LInternetMapper (LInternetMapper&) ;</code>
Parameters	None

~LInternetMapper()

Purpose	The destructor destroys the object.
Access	Public, Virtual
Prototype	<code>~LInternetMapper () ;</code>

AbortThreadOperation()

Purpose	Aborts a thread operation.		
Access	Pure virtual, Public		
Prototype	<code>virtual void AbortThreadOperation(LThread * inThread) = 0;</code>		
Parameters	The parameters for this method are:		
	LThread*	inThread	The thread.
Return	None		

AddressToName()

Purpose	Convert an address to a name.							
Access	Pure virtual, Public							
Prototype	<pre>virtual void AddressToName(UInt32 inHostIP, LStr255& outHostName) = 0;</pre>							
Parameters	The parameters for this method are:							
	<table> <tr> <td>UInt32</td><td>inHostIP</td><td>The host IP address.</td></tr> <tr> <td>LStr255&</td><td>outHostName</td><td>The host name.</td></tr> </table>	UInt32	inHostIP	The host IP address.	LStr255&	outHostName	The host name.	
UInt32	inHostIP	The host IP address.						
LStr255&	outHostName	The host name.						
Return	None							

GetLocalAddress()

Purpose	Obtain the local address.	
Access	Pure virtual, Public	
Prototype	<pre>virtual LInternetAddress* GetLocalAddress() = 0;</pre>	
Parameters	None	
Return	A pointer to the address.	

NameToAddress()

Purpose	Convert a name to an address.	
Access	Pure virtual, Public	
Prototype	<pre>UInt32 NameToAddress(ConstStringPtr inHostName) = 0;</pre>	
Parameters	The parameters for this method are:	

LInternetMapper

		ConstStringPtr	inHostName	The host name.
Return	None			

LIInterruptSafeList

Overview	LIInterruptSafeList is a PowerPlant class that is used for....
Methods	<div>The methods in this class are:</div> <div><div>LIInterruptSafeList()</div><div>~LIInterruptSafeList()</div><div>Append()</div><div>IsEmpty()</div><div>Remove()</div></div>
Data Members	<div>The data members in this class are:</div> <div><div>mQueue</div><div>mIteratorQueue</div></div>
Operation	This class implements a container class which can add and remove members at interrupt time. The current implementation uses the Mac OS Toolbox calls <code>Enqueue()</code> and <code>Dequeue()</code> , but this may change at a later time.
Source files	<div>(Networking Classes)</div> <div><code>LIInterruptSafeList.h</code></div> <div><code>LIInterruptSafeList.cp</code></div>
See also	LNetMessageQueue

LIInterruptSafeList()

Purpose	The constructor creates the object.
Access	Public
Prototype	<code>LIInterruptSafeList();</code>
Parameters	None

LInterruptSafeList

~LInterruptSafeList()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LInterruptSafeList();</code>

Append()

Purpose	Call to add an item to the end of a list. Note that there is currently no way to insert an item at any other position in a list. This method may be called at interrupt time.		
Access	Virtual, Public		
Prototype	<code>virtual void Append(LInterruptSafeListMember* inItem);</code>		
Parameters	The parameter for this method is:		
	<code>LInterruptSafeListMember*</code>	<code>inItem</code>	The pointer to the list item.
Return	None		

IsEmpty()

Purpose	Returns true if there are no entries in this list.
Access	Virtual, Public
Prototype	<code>virtual Boolean IsEmpty() const;</code>
Parameters	None
Return	Return true if list is empty.

Remove()

Purpose	Call to remove an item from a list. This method may be called at interrupt time.		
Access	Virtual, Public		
Prototype	virtual Boolean Remove(LInterruptSafeListMember* inItem);		
Parameters	The parameter for this method is:		
	LInterruptSafeListMember*	inItem	The pointer to the list item.
Return	Returns false if unable to remove the item from the list (i.e. it was not a member of the list, or another thread of execution removed the item first).		

mQueue

Purpose	The queue head.
Access	Private
Prototype	QHdr mQueue;

mIteratorQueue

Purpose	An iterator for the queue.
Access	Private
Prototype	QHdr mIteratorQueue;

LKeyScrollAttachment

Overview	LKeyScrollAttachment is a PowerPlant class that is used for handling scrolling of a View using the Home, End, PageUp, or PageDown keyboard navigation keys.
Methods	The methods in this class are: LKeyScrollAttachment() ExecuteSelf()
Data Members	The data members in this class are: mViewToScroll
Operation	<p>This class is for use only with <code>msg_KeyPress</code>. If you have a View that is also a Commander, you can attach a LKeyScrollAttachment to it to implement keyboard navigation.</p> <p>If your View is not a Commander, but you still want to implement keyboard navigation, you can attach a LKeyScrollAttachment to a SuperView that is a Commander (such as the Window containing the View). However, if you can delete the View independent of the Window, you must take care to delete the Attachment.</p>
Source files	(Utility Classes) <code>UAttachments.h</code> <code>UAttachments.cp</code>
See also	LAttachment LCommander LPane LView LWindow

LKeyScrollAttachment()

Purpose	The constructor creates the object using the passed-in parameters.	
Access	Public	
Prototype	<pre>LKeyScrollAttachment(LView *inViewToScroll); LKeyScrollAttachment(LStream *inStream);</pre>	
Parameters	The parameters for these constructors are:	
	<pre>LView* inViewToScroll</pre>	A pointer to a LView to set mViewToScroll to.
	<pre>LStream* inStream</pre>	A pointer to a stream object that contains the information to create the LBorderAttachment object.

ExecuteSelf()

Purpose	This method handles keyboard navigation scrolling. If the key that is passed to this method via <code>ioParam</code> is not Home, End, PageUp, or PageDown, then mExecuteHost will be set to <code>true</code> , else it is set to <code>false</code> . If <code>true</code> , this indicates that the host should handle the key event. This is an override of ExecuteSelf() in LAttachment .
---------	--

mViewToScroll

Purpose	A pointer to a LView that we want to scroll using keyboard navigational keys.
Access	Protected
Prototype	<pre>LView *mViewToScroll;</pre>

LLink

Overview	LLink is a PowerPlant class that is used for a simple linked list implementation.
Methods	The methods in this class are: LLink() GetLink() ~LLink() SetLink()
Data Members	The data members in this class are: mLink
Operation	Use SetLink() and GetLink() . Like duh, that's all that's here. You typically don't use this class directly. It serves as a base class for custom subclasses. A typical subclass of LLink adds data members that store data you want to pass between threads in an LQueue object.
Source files	(Thread Classes) LLink.h LLink.cp
See also	LQueue

LLink()

Purpose	The constructor creates the object.
Access	Public
Prototype	<code>LLink();</code> <code>LLink (LLink* inLinkP);</code>
Parameters	A pointer to a link.

LLink

~LLink()

Purpose	The destructor destroys the LLink object.
Access	Virtual
Prototype	<code>virtual ~LLink();</code>

GetLink()

Purpose	Returns the contents of the link field.
Access	Inline, Public
Prototype	<code>inline LLink* GetLink() const;</code>
Parameters	None
Return	The link

SetLink()

Purpose	Changes the link field.
Access	Inline, Public
Prototype	<code>inline void SetLink(LLink* inLinkP);</code>
Parameters	A pointer to the link to set.
Return	None

mLink

Purpose	The link
---------	----------

Access	Protected
Prototype	LLink* mLink;

LListBox

Overview LListBox is a PowerPlant class that is used for wrapping the Mac OS List Manager functionality.

Methods The methods in this class are:

<u>LListBox()</u>	<u>~LListBox()</u>
<u>ActivateSelf()</u>	<u>BeTarget()</u>
<u>ClickSelf()</u>	<u>DeactivateSelf()</u>
<u>DoNavigationKey()</u>	<u>DoTypeSelection()</u>
<u>DontBeTarget()</u>	<u>DrawSelf()</u>
<u>FindCommandStatus()</u>	<u>FocusDraw()</u>
<u>GetDescriptor()</u>	<u>GetDoubleClickMessage()</u>
<u>GetFocusBox()</u>	<u>GetLastSelectedCell()</u>
<u>GetMacListH()</u>	<u>GetValue()</u>
<u>HandleKeyPress()</u>	<u>HideSelf()</u>
<u>InitListBox()</u>	<u>MakeCellVisible()</u>
<u>MoveBy()</u>	<u>ObeyCommand()</u>
<u>ResizeFrameBy()</u>	<u>RestorePlace()</u>
<u>SavePlace()</u>	<u>SelectAllCells()</u>
<u>SelectOneCell()</u>	<u>SetDescriptor()</u>
<u>SetDoubleClickMessage()</u>	<u>SetValue()</u>
<u>ShowSelf()</u>	<u>UnselectAllCells()</u>

Data Members The data members in this class are:

<u>mMacListH</u>	<u>mDoubleClickMessage</u>
<u>mFocusBox</u>	<u>mTextTraitsID</u>
<u>mHasGrow</u>	

LListBox

Operation	For detailed information on the workings of this class, refer to <i>The PowerPlant Book</i> .
Source files	(Pane Classes) LListBox.h LListBox.cp
Ancestors	LAttachable LBroadcaster LCommander LPane

LListBox()

Purpose	The constructor creates objects from the passed-in parameters.	
Access	Public	
Prototype	<pre>LListBox(); LListBox(const LListBox &inOriginal); LListBox(const SPaneInfo &inPaneInfo, Boolean inHasHorizScroll, Boolean inHasVertScroll, Boolean inHasGrow, Boolean inHasFocusBox, MessageT inDoubleClickMessage, SInt16 inTextTraitsID, SInt16 inLDEFid, LCommander *inSuper); LListBox(LStream *inStream);</pre>	
Parameters	The parameters for these constructors are:	
const LListBox&	inOriginal	The object to be copied.
const SPaneInfo&	inPaneInfo	The reference to the Superpane.

Boolean	inHasHorizScroll	Set if horizontal scrolling.
Boolean	inHasVertScroll	Set if vertical scrolling.
Boolean	inHasGrow	Set if the box can grow.
Boolean	inHasFocusBox	Set if it has a focus box.
MessageT	inDoubleClickMessage	The double-click message.
SInt16	inTextTraitsID	The resource ID for the text traits.
SInt16	inLDEFid	The ID for the LDEF List Definition resource.
LCommander*	inSuper	The pointer to the Supercommander.
LStream*	inStream	The pointer to the stream object to read from.

~LListBox()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LListBox();</code>

ActivateSelf()

Purpose	Activate the ListBox. The Toolbox shows the selection and scroll bars. This is an override of ActivateSelf() in LPane .
---------	---

BeTarget()

Purpose	ListBox is becoming the Target. This is an override of BeTarget() in LCommander .
---------	---

ClickSelf()

Purpose Respond to Click inside an ListBox. This is an override of [ClickSelf\(\)](#) in [LPane](#).

DeactivateSelf()

Purpose Deactivate ListBox. The Toolbox hides the selection and scroll bars. This is an override of [DeactivateSelf\(\)](#) in [LPane](#).

DoNavigationKey()

Purpose Implements keyboard navigation by supporting selection change using the arrow keys, page up, page down, home, and end.

Access Virtual, Protected

Prototype `virtual void DoNavigationKey(
const EventRecord &inKeyEvent);`

Parameters The parameter for this method is:

<code>const EventRecord&</code>	<code>inKeyEvent</code>	The event record for the key press.
---	-------------------------	-------------------------------------

Return None

DoTypeSelection()

Purpose Change selection to the item beginning with the input characters.

Access Virtual, Protected

Prototype	<code>virtual void DoTypeSelection(const EventRecord& inKeyEvent);</code>
-----------	---

Parameters	The parameter for this method is:
------------	-----------------------------------

<code>const EventRecord&</code>	<code>inKeyEvent</code>	The event record for the key press.
---	-------------------------	--

Return	None
--------	------

DontBeTarget()

Purpose	Listbox is no longer the Target. This is an override of DontBeTarget() in LCommander .
---------	--

DrawSelf()

Purpose	Draw Listbox. This is an override of DrawSelf() in LPane .
---------	--

FindCommandStatus()

Purpose	Pass back the status of a Command. This is an override of FindCommandStatus() in LCommander .
---------	---

FocusDraw()

Purpose	Focus drawing. This is an override to use the default Pen state and to set the TextTraits used by the Listbox. This is an override of FocusDraw() in LPane .
---------	--

GetDescriptor()

Purpose	Return the descriptor of a ListBox, which is the text of the first selected cell. The descriptor is an empty string if there are no selected cells. This method assumes that the cell data is text. This is an override of GetDescriptor() in LPane .
---------	---

GetDoubleClickMessage()

Purpose	Retrieve the double-click message.
Access	Public
Prototype	<code>MessageT GetDoubleClickMessage() const;</code>
Parameters	None
Return	Returns the message for the double-click.

GetFocusBox()

Purpose	Retrieve the focus box from mFocusBox .
Access	Inline, Public
Prototype	<code>LFocusBox* GetFocusBox();</code>
Parameters	None
Return	Return the mFocusBox .

GetLastSelectedCell()

Purpose	Pass back the last selected Cell in a ListBox.
---------	--

Access	Virtual, Public			
Prototype	<code>virtual Boolean GetLastSelectedCell (Cell &outCell);</code>			
Parameters	<div>The parameter for this method is:</div> <table><tr><td>Cell&</td><td>outCell</td><td>The reference to the last selected Cell.</td></tr></table>	Cell&	outCell	The reference to the last selected Cell.
Cell&	outCell	The reference to the last selected Cell.		
Return	Returns false if no cells are selected.			

GetMacListH()

Purpose	Retrieve the Mac OS list handle.
Access	Public
Prototype	<code>ListHandle GetMacListH() const;</code>
Parameters	None
Return	Returns the list handle.

GetValue()

Purpose	Return the value of a ListBox. <p>The "value" of a ListBox is the row number of the first selected cell, with the first row being number 0 (the ListManager uses zero-based numbering). If no cells are selected, the value is -1. This "value" makes sense for a ListBox with one column, by far the most common case.</p> <p>This is an override of GetValue() in LPane.</p>
---------	---

LListBox

HandleKeyPress()

Purpose LListBox supports keyboard navigation and type selection. This is an override of [HandleKeyPress\(\)](#) in [LCommander](#).

HideSelf()

Purpose LListBox is becoming invisible. This is an override of [HideSelf\(\)](#) in [LPane](#).

InitListBox()

Purpose Private initializer

Access Private

Prototype `void InitListBox(
 Boolean inHasHorizScroll,
 Boolean inHasVertScroll,
 Boolean inHasGrow,
 Boolean inHasFocusBox,
 MessageT inDoubleClickMessage,
 SInt16 inTextTraitsID,
 SInt16 inLDEFid);`

Parameters The parameters for these constructors are:

Boolean	inHasHorizScroll	Set if horizontal scrolling.
Boolean	inHasVertScroll	Set if vertical scrolling.
Boolean	inHasGrow	Set if the box can grow.
Boolean	inHasFocusBox	Set if it has a focus box.
MessageT	inDoubleClickMessag e	The double-click message.

SInt16	inTextTraitsID	The resource ID for the text traits.
SInt16	inLDEFid	The ID for the LDEF List Definition resource.
Return	None	

MakeCellVisible()

Purpose	Scroll the ListBox as little as possible to move the specified Cell into view.	
Access	Virtual, Public	
Prototype	<code>virtual void MakeCellVisible(Cell inCell);</code>	
Parameters	The parameter for this method is:	
	<u>Cell& inCell</u>	<u>The reference to the Cell.</u>
Return	None	

MoveBy()

Purpose	Move the location of the Frame by the specified amounts. Both PowerPlant and the List Manager store a location for the ListBox, so we must make sure that both locations are in synch. This is an override of MoveBy() in LPane .
---------	---

ObeyCommand()

Purpose	Respond to Command message. This is an override of ObeyCommand() in LCommander .
---------	--

ResizeFrameBy()

Purpose Change the Frame size by the specified amounts. Both PowerPlant and the List Manager store a size for the ListBox, so we must make sure that both sizes are in synch. This is an override of [ResizeFrameBy\(\)](#) in [LPane](#).

RestorePlace()

Purpose Read size and location information stored in a Stream by the [SavePlace\(\)](#) function.

Access Public

Prototype `virtual void RestorePlace(LStream *inPlace);`

Parameters The parameter for this method is:

<code>LStream*</code>	<code>inPlace</code>	Reference to the stream to read from.
-----------------------	----------------------	---------------------------------------

Return None

SavePlace()

Purpose Write size and location information to a Stream for later retrieval by the [RestorePlace\(\)](#) function.

Access Public

Prototype `virtual void SavePlace(LStream *outPlace);`

Parameters The parameter for this method is:

<code>LStream*</code>	<code>outPlace</code>	The reference to the stream to write to.
-----------------------	-----------------------	--

Return None

SelectAllCells()

Purpose	Select all the cells in a ListBox.
Access	Virtual, Public
Prototype	<code>virtual void SelectAllCells();</code>
Parameters	None
Return	None

SelectOneCell()

Purpose	Select the specified Cell and deselect all others.			
Access	Virtual, Public			
Prototype	<code>virtual void SelectOneCell(Cell inCell);</code>			
Parameters	<div>The parameter for this method is:</div> <table><tr><td>Cell&</td><td>inCell</td><td>The reference to the Cell.</td></tr></table>	Cell&	inCell	The reference to the Cell.
Cell&	inCell	The reference to the Cell.		
Return	None			

SetDescriptor()

Purpose	Set the descriptor of a ListBox, which is the text of the first selected cell. Nothing happens if there are no selected cells.
	This function assumes that the cell data is text. This is an override of SetDescriptor() in LPane .

SetDoubleClickMessage()

Purpose	Set the double click message.	
Access	Public	
Prototype	<code>void SetDoubleClickMessage(MessageT inMessage);</code>	
Parameters	The parameter for this method is:	
	<code>MessageT inMessage</code>	The message to set for double click.
Return	None	

SetValue()

Purpose	Set the value of a ListBox. The "value" of a ListBox is the row number of the first selected cell, with the first row being number 0 (the ListManager uses zero-based numbering). If no cells are selected, the value is -1. This "value" makes sense for a ListBox with one column, by far the most common case. This function selects the cell in row "inValue" and column 1, deselecting any previously selected cells. This is an override of SetValue() in LPane .
---------	--

ShowSelf()

Purpose	ListBox is becoming visible. This is an override of ShowSelf() in LPane .
---------	---

UnselectAllCells()

Purpose	Unselect all the cells in a ListBox.
Access	Public
Prototype	<code>virtual void UnselectAllCells();</code>
Parameters	None
Return	None

mMacListH

Purpose	The Mac OS list handle.
Access	Protected
Prototype	<code>ListHandle mMacListH;</code>

mDoubleClickMessage

Purpose	Storage for the double click message.
Access	Protected
Prototype	<code>MessageT mDoubleClickMessage;</code>

mFocusBox

Purpose	Storage for the focus box.
Access	Protected
Prototype	<code>LFocusBox *mFocusBox;</code>

LListBox

mTextTraitsID

Purpose	The resource ID for the text traits.
Access	Protected
Prototype	ResIDT mTextTraitsID;

mHasGrow

Purpose	Indicates whether the box can grow.
Access	Protected
Prototype	Boolean mHasGrow;

LListener

Overview	LListener is a PowerPlant class that is used for enabling your objects to respond to messages broadcasted by LBroadcaster objects.								
Methods	<p>The methods in this class are:</p> <table><tr><td>LListener()</td><td>~LListener()</td></tr><tr><td>AddBroadcaster()</td><td>IsListening()</td></tr><tr><td>ListenToMessage()</td><td>RemoveBroadcaster()</td></tr><tr><td>StartListening()</td><td>StopListening()</td></tr></table>	LListener()	~LListener()	AddBroadcaster()	IsListening()	ListenToMessage()	RemoveBroadcaster()	StartListening()	StopListening()
LListener()	~LListener()								
AddBroadcaster()	IsListening()								
ListenToMessage()	RemoveBroadcaster()								
StartListening()	StopListening()								
Data Members	<p>The data members in this class are:</p> <table><tr><td>mBroadcasters</td><td>mIsListening</td></tr></table>	mBroadcasters	mIsListening						
mBroadcasters	mIsListening								
Operation	An object that inherits from LListener receives messages from Broadcaster objects. To use LListener with your objects, inherit LListener in your class definition, and provide an implementation for the ListenToMessage() pure virtual method.								
Source files	<p>(Feature Classes)</p> <p><code>LListener.h</code></p> <p><code>LListener.cp</code></p>								
See also	LBroadcaster								

LListener()

Purpose	The default constructor enables listening for the object by setting mIsListening to true. The copy constructor copies the state of mIsListening , but does not copy any links to Broadcaster objects.
Access	Public
Prototype	<pre>LListener(); LListener(const LListener &inOriginal);</pre>

LListener

Parameters	None
Return	None

```
~LListener()
```

Purpose	The destructor removes this Listener from all the Broadcaster objects that it registered with.
Access	Public, Virtual
Prototype	<code>virtual ~LListener();</code>
Parameters	None
Return	None

AddBroadcaster()

Purpose	<p>This method adds a link from a Broadcaster to a Listener. LBroadcaster::AddBroadcaster() will call this method.</p> <pre>theBroadcaster->AddListener(theListener); // correct theListener->AddBroadcaster(theBroadcaster); // incorrect!</pre>			
Access	Protected			
Prototype	<pre>void LListener::AddBroadcaster(LBroadcaster *inBroadcaster);</pre>			
Parameters	<table><tr><td>LBroadcaster*</td><td>inBroadcaster</td><td>a pointer to an LBroadcaster object</td></tr></table>	LBroadcaster*	inBroadcaster	a pointer to an LBroadcaster object
LBroadcaster*	inBroadcaster	a pointer to an LBroadcaster object		
Return	None			
Remarks	Generally, you should not call this function directly.			

IsListening()

Purpose	This method is an accessor for the mIsListening data member.
Access	Public
Prototype	<code>Boolean IsListening() const;</code>
Parameters	None
Return	A Boolean value of true indicates that the Listener object is currently listening.

ListenToMessage()

Purpose	This method provides the message processing behavior for an LListener-inherited object. Your object inherits from LListener, and must provide some behavior for handling broadcasted messages. This method is where that behavior should be implemented.		
Access	Public, Pure virtual		
Prototype	<code>virtual void ListenToMessage(MessageT inMessage, void *ioParam) = 0;</code>		
Parameters	MessageT	inMessage	This parameter is the message that the Broadcaster is sending to the Listener.
	void*	ioParam	This pointer is used to pass a parameter to the Listener message-processing code
Return	None		
Remarks	Your object that inherits from LListener must implement this method, since it is a pure virtual method.		

RemoveBroadcaster()

Purpose	This method removes a Broadcaster from the list maintained by the LListener object. <code>theBroadcaster->RemoveListener(theListener); //</code> correct <code>theListener->RemoveBroadcaster(theBroadcaster); //</code> incorrect!		
Access	Protected		
Prototype	<code>void LListener::RemoveBroadcaster(LBroadcaster *inBroadcaster)</code>		
Parameters	<code>LBroadcaster*</code>	<code>inBroadcaster</code>	a pointer to the LBroadcaster object that is removed from the list
Return	None		
Remarks	Normally, you should not call this method directly. LBroadcaster::RemoveListener() will call it.		

StartListening()

Purpose	This method causes the object to start listening to Broadcaster messages. This method is an accessor method for mIsListening .		
Access	Public		
Prototype	<code>void StartListening();</code>		
Parameters	None		
Return	None		
Remarks	This method sets the mIsListening data member to true.		

StopListening()

Purpose	This method causes the object to stop listening to Broadcaster messages. This method is an accessor method for mIsListening .
Access	Public
Prototype	<code>void StopListening();</code>
Parameters	None
Return	None
Remarks	This method sets the mIsListening data member to false.

mBroadcasters

Purpose	This data member is an array of pointers to Broadcaster objects.
Access	Protected
Prototype	TArray < LBroadcaster *> mBroadcasters;

mIsListening

Purpose	This data member tells whether the Listener is listening to Broadcasted messages.
Access	Protected
Prototype	<code>Boolean mIsListening;</code>

LLongComparator

Overview	LLongComparator is a PowerPlant class that is used for comparing long quantities.	
Methods	The methods in this class are: LLongComparator() Clone() GetComparator() ~LLongComparator() Compare() IsEqualTo()	
Data Members	The data members in this class are: sLongComparator	
Operation	For information on using this class, refer to LComparator .	
Source files	(Array Classes) LComparator.h LComparator.cp	
Ancestors	LComparator	

LLongComparator()

Purpose	The constructor is a placeholder, it doesn't do anything for this class.
Access	Public
Prototype	<code>LLongComparator() ;</code>
Parameters	None

LLongComparator

~LLongComparator()

Purpose	The destructor is a placeholder, it doesn't do anything for this class.
Access	Virtual, Public
Prototype	<code>~LLongComparator();</code>

Clone()

Purpose	This method behaves identically to Clone() in the base class LComparator .
---------	--

Compare()

Purpose	This method compares the contents of the two items to determine the comparison value.		
Access	Virtual, Public		
Prototype	<code>virtual SInt32 Compare(const void* inItemOne, const void* inItemTwo, UInt32 inSizeOne, UInt32 inSizeTwo) const;</code>		
Parameters	This method has the following parameters:		
	<code>const void*</code>	<code>inItemOne</code>	A data pointer to the first item.
	<code>const void*</code>	<code>inItemTwo</code>	A data pointer to the second item.
	<code>UInt32</code>	<code>inSizeOne</code>	Unused
	<code>UInt32</code>	<code>inSizeTwo</code>	Unused

Return `SInt32` that indicates one of the following:

- `< 0` — object/data item 1 is less than object/data item 2
- `0` — object/data item 1 is equal to object/data item 2
- `> 0` — object/data item 1 is greater than object/data item 2

GetComparator()

Purpose This method behaves identically to [GetComparator\(\)](#) in the base class [LComparator](#).

IsEqualTo()

Purpose This method detects whether two objects are equal.

Access Virtual, Public

Prototype `virtual Boolean IsEqualTo(
 const void* inItemOne,
 const void* inItemTwo,
 UInt32 inSizeOne,
 UInt32 inSizeTwo) const;`

Parameters This method has the following parameters:

<code>const void*</code>	<code>inItemOne</code>	A data pointer to the first item.
<code>const void*</code>	<code>inItemTwo</code>	A data pointer to the second item.
<code>UInt32</code>	<code>inSizeOne</code>	Unused
<code>UInt32</code>	<code>inSizeTwo</code>	Unused

Return Boolean indicating true if the items are equal, false otherwise.

LLongComparator

sLongComparator

Purpose This data member stores the pointer to the comparison “standard” object.

Access Protected

Prototype `static LLongComparator* sLongComparator;`

LMacTCPDNSOperation

Overview	LMacTCPDNSOperation is a PowerPlant class that is used for managing thread blocking for MacTCP DNS interface calls.	
Methods	The methods in this class are:	
	LMacTCPDNSOperation()	~LMacTCPDNSOperation()
	GetResultProc()	Int_DNSCompletionProc()
	Int_DNSCompletionProc2()	WaitForCompletion()
Data Members	The data members in this class are:	
	mOperationListMem	sMacTCPDNSCompletionProc
	sMacTCPDNSPendingOperations;	sDNSOperationDeleteQueue
Source files	(Networking Classes)	
	UMacTCPSupport.h	
	UMacTCPSupport.cp	
See also	StAsyncOperation	

LMacTCPDNSOperation()

Purpose	The constructor creates the object.
Access	Public
Prototype	<code>LMacTCPDNSOperation();</code>
Parameters	None

LMacTCPDNSOperation

~LMacTCPDNSOperation()

Purpose	The destructor destroys the object.
Access	Public
Prototype	<code>~LMacTCPDNSOperation();</code>

GetResultProc()

Purpose	Return the value of sMacTCPDNSCompletionProc .
Access	Inline, Public
Prototype	<code>inline ResultUPP GetResultProc();</code>
Parameters	None
Return	The value of sMacTCPDNSCompletionProc .

Int_DNSCompletionProc()

Purpose	The completion routine.		
Access	Static, Protected		
Prototype	<pre>pascal void Int_DNSCompletionProc(hostInfo* inHostInfoPtr, Ptr inUserDataPtr);</pre>		
Parameters	This method has the following parameters:		
	hostInfo*	inHostInfoPtr	The pointer to the host info.
	Ptr	inUserDataPtr	The user data pointer.
Return	None		

Int_DNSCompletionProc2()

Purpose	The DNS completion procedure.		
Access	Static, Protected		
Prototype	<pre>pascal void Int_DNSCompletionProc2(returnRec*inReturnRecPtr, Ptr inUserDataPtr);</pre>		
Parameters	This method has the following parameters:		
	returnRec*	inReturnRecPtr	The pointer to the return record.
	Ptr	inUserDataPtr	The user data pointer.
Return	None		

WaitForCompletion()

Purpose	Wait for the completion routine to execute.
Access	Public
Prototype	<pre>void WaitForCompletion();</pre>
Parameters	None
Return	None

mOperationListMem

Purpose	The operation list.
Access	Private
Prototype	<pre>LOperationListMember * mOperationListMem;</pre>

LMacTCPDNSOperation

sMacTCPDNSCompletionProc

Purpose	The DNS completion routine pointer.
Access	Static, Private
Prototype	<code>static ResultUPP sMacTCPDNSCompletionProc;</code>

sMacTCPDNSPendingOperations;

Purpose	The MacTCP DNS pending operation.
Access	Static, Private
Prototype	<code>static UInt16 sMacTCPDNSPendingOperations;</code>

sDNSOperationDeleteQueue

Purpose	The DNS operation deletion queue.
Access	Static, Private
Prototype	<code>LDNSOperationDeleteQueue * sDNSOperationDeleteQueue;</code>

LMacTCPTCPSendQueue

Overview	LMacTCPTCPSendQueue is a PowerPlant class that is used for queuing data for asynchronous style data sending for MacTCP.	
Methods	The methods in this class are: LMacTCPTCPSendQueue() Int_InternalSend() ~LMacTCPTCPSendQueue() NotifyRelease()	
Data Members	The data members in this class are: mWDS mOperation mMacTCPEndpoint	
Operation	This class is used primarily for Open Transport-style “acked” data transfers which are not native to MacTCP.	
Source files	(Networking Classes) LMacTCPTCPSendQueue.h LMacTCPTCPSendQueue.cp	
See also	LSendQueue	

LMacTCPTCPSendQueue()

Purpose	The constructor creates the object.	
Access	Public	
Prototype	LMacTCPTCPSendQueue (LMacTCPTCPEndpoint *inEndpoint);	
Parameters	This constructor has the following parameter: LMacTCPTCPEndpoint* inEndpoint This is the endpoint.	

~LMacTCPTCPSendQueue()

Purpose	The destructor destroys the object.
Access	Public
Prototype	<code>virtual ~LMacTCPTCPSendQueue();</code>

Int_InternalSend()

Purpose	Sends are chained via the completion routine. NOTE: This routine may be called at interrupt time.
Access	Virtual, Protected
Prototype	<code>virtual void Int_InternalSend();</code>
Parameters	None
Return	None

NotifyRelease()

Purpose	Returns the size of the pending data in the queue.			
Access	Virtual			
Prototype	<code>virtual void NotifyRelease(LSendData* inData);</code>			
Parameters	<div>This method has the following parameter:</div> <table><tr><td>LSendData*</td><td>inData</td><td>The data.</td></tr></table>	LSendData*	inData	The data.
LSendData*	inData	The data.		
Return	None			

mWDS

Purpose	Data storage.
Access	Protected
Prototype	<code>wdsEntry mWDS[5];</code>

mMacTCPEndpoint

Purpose	The MacTCP endpoint.
Access	Protected
Prototype	<code>LMacTCPEndpoint* mMacTCPEndpoint;</code>

mOperation

Purpose	The operation.
Access	Protected
Prototype	<code>StMacTCPTCPSendOperation * mOperation;</code>

LMacTCPUDPEndpoint

Overview	LMacTCPUDPEndpoint is a PowerPlant class that is an implementation of LTCPEndpoint that communicates with MacTCP.	
Methods	The methods in this class are:	
	LMacTCPUDPEndpoint()	~LMacTCPUDPEndpoint()
	AbortThreadOperation()	AckSends()
	Bind()	DontAckSends()
	GetLocalAddress()	GetState()
	Int HandleAsyncEvent()	Int UDPNotifyProc()
	IsAckingSends()	ReceiveFrom()
	SendPacketData()	Unbind()
Data Members	The data members in this class are:	
	mUDPStream	mLocalAddress
	mAckSends	mEndpointState
	mReceiveBuffer	mReceiveBufferSize
	mMessageQueue	mSharedPool
	mSendQueue	sUDPNotifyUPP
Operation	We recommend that you do not make a subclass of this class.	
Source files	(Networking Classes)	
	LMacTCPUDPEndpoint.h	
	LMacTCPUDPEndpoint.cp	
See also	LTCPEndpoint	
	LCleanupTask	

LMacTCPUDPEndpoint

LMacTCPUDPEndpoint()

Purpose	The constructor creates the object.		
Access	Public		
Prototype	<code>LMacTCPUDPEndpoint (UInt32 inReceiveBufferSize) ;</code>		
Parameters	This constructor has the following parameter:		
	UInt32	inReceiveBufferSize	This is the size of the receive buffer.

~LMacTCPUDPEndpoint()

Purpose	The destructor destroys the object.		
Access	Virtual, Public		
Prototype	<code>virtual ~LMacTCPUDPEndpoint() ;</code>		
Parameters	None		

AbortThreadOperation()

Purpose	Abort the thread operation.		
Access	Virtual, Public		
Prototype	<code>virtual void AbortThreadOperation(LThread * inThread) ;</code>		
Parameters	This method has the following parameter:		
	LThread*	inThread	This is the thread.
Return	None		

AckSends()

Purpose	Enable the sending of acknowledgements.
Access	Virtual, Public
Prototype	<code>virtual void AckSends();</code>
Parameters	None
Return	None

Bind()

Purpose	Bind an address.		
Access	Virtual, Public		
Prototype	<pre>virtual void Bind(LInternetAddress& inLocalAddress, UInt32 inListenQueueSize, Boolean inReusePort);</pre>		
Parameters	This method has the following parameter:		
	LInternetAddress&	inLocalAddress	The local address.
	UInt32	inListenQueueSize	The size of the listen queue.
	Boolean	inReusePort	Whether to reuse the port.
Return	None		

DontAckSends()

Purpose	Turn off the generation of acknowledgments.
---------	---

LMacTCPUDPEndpoint

Access	Virtual, Public
Prototype	<code>virtual void DontAckSends();</code>
Parameters	None
Return	None

GetLocalAddress()

Purpose	Obtain the local address, held in mLocalAddress .
Access	Virtual, Public
Prototype	<code>LInternetAddress* GetLocalAddress();</code>
Parameters	None
Return	The local address.

GetState()

Purpose	Retrieve the endpoint state, stored in mEndpointState .
Access	Virtual, Public
Prototype	<code>virtual EEndpointState GetState();</code>
Parameters	None
Return	The value of mEndpointState .

Int_HandleAsyncEvent()

Purpose	Called by NotifyProc to handle notifications for this endpoint. You will be notified via a Broadcast at primary task time of the event if necessary.
---------	--

Warning: This routine will probably be called at interrupt time.

Access	Protected		
Prototype	<pre>void Int_HandleAsyncEvent (UInt16 inEventCode, ICMPReport* inEventCode, StreamPtr inStreamPtr);</pre>		
Parameters	This method has the following parameters:		
	UInt16	inEventCode	The local address.
	ICMPReport*	inEventCode	The size of the listen queue.
	StreamPtr	inStreamPtr	Whether to reuse the port.
Return	None		

Int_UDPNotifyProc()

Purpose			
Access	Static, Protected		
Prototype	<pre>pascal static void Int_UDPNotifyProc (StreamPtr inStreamPtr, UInt16 inEventCode, Ptr inUserPtr, ICMPReport* inIcmpMsg);</pre>		
Parameters	This method has the following parameters:		
	StreamPtr	inStreamPtr	The stream pointer.
	ICMPReport*	inEventCode	The size of the listen queue.
	UInt16	inUserPtr	The pointer to the user area.
	ICMPReport*	inIcmpMsg	The ICMP msg info.
Return	None		

IsAckingSends()

Purpose	Determine whether sends are being acknowledged.
Access	Virtual, Public
Prototype	<code>virtual Boolean IsAckingSends();</code>
Parameters	None
Return	Returns the value of mAckSends .

ReceiveFrom()

Purpose			
Access	Virtual, Public		
Prototype	<code>virtual void ReceiveFrom(LInternetAddress& outRemoteAddress, void* outDataBuffer, UInt32& ioDataSize, UInt32 inTimeoutSeconds);</code>		
Parameters	This method has the following parameters:		
	LInternetAdd ress&	outRemoteAddress	The remote address.
	void*	outDataBuffer	The data output buffer.
	UInt32&	ioDataSize	The buffer size.
	UInt32	inTimeoutSeconds	The timeout period to wait for data.
Return	None		

SendPacketData()

Purpose	Send data.		
Access	Virtual, Public		
Prototype	<pre>virtual void SendPacketData(LInternetAddress& inRemoteHost, void* inData, UInt32 inDataSize);</pre>		
Parameters	This method has the following parameters:		
	LInternetAdd ress&	inRemoteHost	The remote address.
	void*	inData	The data input.
	UInt32	inDataSize	The buffer size.
Return	None		

Unbind()

Purpose	Unbind from a port.
Access	Virtual, Public
Prototype	<pre>virtual void Unbind();</pre>
Parameters	None
Return	None

mUDPStream

Purpose	The UDP stream.
Access	Protected

LMacTCPUDPEndpoint

Prototype `StreamPtr mUDPStream;`

mLocalAddress

Purpose `The local address.`

Access `Protected`

Prototype `LInternetAddress * mLocalAddress;`

mAckSends

Purpose `Whether to acknowledge sends or not.`

Access `Protected`

Prototype `Boolean mAckSends;`

mEndpointState

Purpose `The endpoint status.`

Access `Protected`

Prototype `UInt16 mEndpointState;`

mReceiveBuffer

Purpose `The receive buffer pointer.`

Access `Protected`

Prototype `Ptr mReceiveBuffer;`

mReceiveBufferSize

Purpose	The receive buffer size in bytes.
Access	Protected
Prototype	UInt32 mReceiveBufferSize;

mMessageQueue

Purpose	The message queue.
Access	Protected
Prototype	NetMessageQueue * mMessageQueue;

mSharedPool

Purpose	The shared memory pool.
Access	Protected
Prototype	SharedMemoryPool * mSharedPool;

mSendQueue

Purpose	The send queue.
Access	Protected
Prototype	MacTCPUDPSendQueue * mSendQueue;

sUDPNotifyUPP

Purpose	The procedure pointer for the UDP notification procedure.
Access	Static, Private
Prototype	UDPNotifyUPP sUDPNotifyUPP;

LMacTCPUDPSendQueue

Overview	LMacTCPUDPSendQueue is a PowerPlant class that is used for simplifying data sends.	
Methods	The methods in this class are:	
	LMacTCPUDPSendQueue()	~LMacTCPUDPSendQueue()
	Int InternalSend()	NotifyRelease()
Data Members	The data members in this class are:	
	mWDS	mMacTCPUDPEndpoint
	mOperation	
Source files	(Networking Classes)	
	<code>LMacTCPUDPSendQueue.h</code>	
	<code>LMacTCPUDPSendQueue.cp</code>	
See also	LSendQueue	

LMacTCPUDPSendQueue()

Purpose	The constructor creates the object. It inherits from the LSendQueue() constructor.	
Access	Public	
Prototype	<code>LMacTCPUDPSendQueue (LMacTCPUDPEndpoint * inEndpoint);</code>	
Parameters	This constructor has the following parameter:	
	<code>LMacTCPUDPEndpoint*</code>	<code>inEndpoint</code> This is the endpoint.

LMacTCPUDPSendQueue

~LMacTCPUDPSendQueue()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LMacTCPUDPSendQueue()</code>

Int_InternalSend()

Purpose	Sends are chained via the completion routine. NOTE: This routine may be called at interrupt time.
Access	Virtual, Protected
Prototype	<code>virtual void Int_InternalSend();</code>
Parameters	None
Return	None

NotifyRelease()

Purpose	Returns the size of the pending data in the queue.
Access	Virtual, Protected
Prototype	<code>void NotifyRelease(LSendData* inData);</code>
Parameters	This method has the following parameter: <div><div>LSendData*</div><div>inData</div><div>This is the data object.</div></div>
Return	None

mWDS

Purpose	Data storage.
Access	Protected
Prototype	<code>wdsEntry mWDS[2];</code>

mMacTCPUDPEndpoint

Purpose	The MacTCP endpoint.
Access	Protected
Prototype	<code>LMacTCPUDPEndpoint * mMacTCPUDPEndpoint;</code>

mOperation

Purpose	The operation.
Access	Protected
Prototype	<code>StMacTCPUDPSendOperation * mOperation;</code>

LMenu

Description	LMenu is a PowerPlant class that is used for managing a Mac OS menu. This class maintains a mapping between Menu items and Command numbers.	
Methods	The methods in this class are:	
	LMenu()	~LMenu()
	CommandFromIndex()	FindNextCommand()
	GetMacMenuH()	GetMenuID()
	GetNextMenu()	IndexFromCommand()
	InsertCommand()	IsInstalled()
	IsUsed()	ItemIsEnabled()
	ReadCommandNumbers()	RemoveCommand()
	RemoveItem()	SetCommand()
	SetInstalled()	SetNextMenu()
	SetUsed()	SyntheticCommandFromIndex()
Data Members	The data members in this class are:	
	mNextMenu	mMacMenuH
	mMENUId	mNumCommands
	mCommandNums	mIsInstalled
	mUsed	
Operation	To learn more about effectively using the LMenu class, refer to <i>The PowerPlant Book</i> for a lengthy discussion on the topic.	
Source Files	(Support Classes)	
	LMenu.h	
	LMenu.cp	

LMenu

LMenu()

Purpose	The constructors create objects from the passed-in parameters.
Access	Public
Prototypes	<pre>LMenu () ; LMenu (ResIDT inMENUid) ; LMenu (SInt16 inMENUid, Str255 inTitle) ;</pre>
Parameters	The parameters for these constructors are:

ResID T	inMENUid	The resource ID for the menu.
SInt1 6	inMENUid	The resource ID for the menu.
Str25 5	inTitle	The string to use for a title.

~LMenu()

Purpose	The destructor destroys the LMenu object.
Access	Public
Prototype	<pre>~LMenu () ;</pre>

CommandFromIndex()

Purpose	Return the command number for a particular Menu item.
Access	Public
Prototype	<pre>CommandT CommandFromIndex (SInt16 inIndex) const ;</pre>
Parameters	The parameter for this method is:

SInt16	inIndex	The index to use to retrieve the command number.
--------	---------	--

Return The command number.

FindNextCommand()

Purpose Retrieve the next command.

Access Public

Prototype `Boolean FindNextCommand(SInt16 &ioIndex, CommandT &outCommand) const;`

Parameters The parameters for this method are:

SInt16&	ioIndex	The index to use to retrieve the command.
---------	---------	---

CommandT&	outCommand	The command.
-----------	------------	--------------

Return Returns `true` if the command is found.

GetMacMenuH()

Purpose Retrieve OS menu handle.

Access Inline, Public

Prototype `MenuHandle GetMacMenuH() const;`

Parameters None

Return A handle to the menu.

GetMenuID()

Purpose	Retrieve the menu ID.
Access	Inline, Public
Prototype	<code>ResIDT GetMenuID() const;</code>
Parameters	None
Return	The resource ID of the menu.

GetNextMenu()

Purpose	Retrieve the next menu.
Access	Inline, Protected
Prototype	<code>LMenu* GetNextMenu() const;</code>
Parameters	None
Return	A pointer to the LMenu object.

IndexFromCommand()

Purpose	Return the Menu item index number for a particular command number Return 0 if the command number is not used for this Men		
Access	Public		
Prototype	<code>SInt16 IndexFromCommand(CommandT inCommand) const;</code>		
Parameters	The parameter for this method is:		
	CommandT&	inCommand	The command.
Return	SInt16 containing the index.		

InsertCommand()

Purpose	Insert an item with the specified text and command number after particular item.		
Access	Public		
Prototype	<pre>void InsertCommand(ConstStringPtr inItemText, CommandT inCommand, SInt16 inAfterItem);</pre>		
Parameters	The parameters for this method are:		
	ConstStringPtr	inItemText	The index to use to retrieve the command.
	CommandT&	inCommand	The command.
	SInt16	inAfterItem	The item to insert after.
Return	None		
Remarks	This function does not support adding multiple menu items using "Return" or "Semicolon" characters within inItemText.		

IsInstalled()

Purpose	Retrieves the value of mIsInstalled .
Access	Inline, Public
Prototype	Boolean IsInstalled() const;
Parameters	None
Return	The value of mIsInstalled .

IsUsed()

Purpose	Retrieves the value of mUsed .
Access	Inline, Public
Prototype	<code>Boolean IsUsed() const;</code>
Parameters	None
Return	The value of mUsed .

ItemIsEnabled()

Purpose	Return whether an item at the specified position is enabled.						
Access	Public						
Prototype	Boolean ItemIsEnabled(SInt16 inIndex) const;						
Parameters	<div>The parameter for this method is:</div> <table><tr><td>SInt16</td><td>inIndex</td><td>The index to check.</td></tr><tr><td>6</td><td></td><td></td></tr></table>	SInt16	inIndex	The index to check.	6		
SInt16	inIndex	The index to check.					
6							
Return	Returns true if enabled, else false.						

ReadCommandNumbers()

Purpose	Get command numbers from the 'Mcmd' resource. A 'Mcmd' resource with the same ID as the 'MENU' resource contains a list of command numbers. The format of a 'Mcmd' resource is: <ul style="list-style-type: none">• Number of Commands: n(2 bytes)• Command Num for Item 1(4 bytes)• Command Num for Item n(4 bytes)
---------	--

Access	Protected
Prototype	<code>void ReadCommandNumbers();</code>
Parameters	None
Return	None

RemoveCommand()

Purpose	Remove the Menu item with the specified command.	
Access	Public	
Prototype	<code>void RemoveCommand(CommandT inCommand);</code>	
Parameters	The parameter for this method is:	
	<code>CommandT</code>	<code>inCommand</code> The command for the menu item to remove.
Return	None	

RemoveItem()

Purpose	Remove the Menu item at the specified position.	
Access	Public	
Prototype	<code>void RemoveItem(SInt16 inItemToRemove);</code>	
Parameters	The parameter for this method is:	
	<code>SInt16</code>	<code>inItemToRemove</code> The menu item to remove.
Return	None	

SetCommand()

Purpose	Set the command number for a Menu item.	
Access	Public	
Prototype	<pre>void SetCommand(SInt16 inIndex, CommandT inCommand);</pre>	
Parameters	The parameters for this method are:	
	SInt16 inIndex	The index to use to set the command.
	CommandT& inComman d	The command.
Return	None	

SetInstalled()

Purpose	Sets the value of the mIsInstalled data member.	
Access	Public	
Prototype	<pre>void SetInstalled(Boolean inInstalled);</pre>	
Parameters	The parameter for this method is:	
	Boolean inInstalled	The value to set.
Return	None	

SetNextMenu()

Purpose	Set the next Menu.
---------	--------------------

Access	Inline, Protected	
Prototype	<code>void SetNextMenu(LMenu *inMenu);</code>	
Parameters	The parameter for this method is:	
	<code>LMenu*</code> <code>inMenu</code>	The pointer to the Menu to set.
Return	None	

SetUsed()

Purpose	This method sets the value of the mUsed data member.	
Access	Public	
Prototype	<code>void SetUsed(Boolean inUsed);</code>	
Parameters	The parameter for this method is:	
	<code>Boolean</code> <code>inUsed</code>	The value to set the data member to.
Return	None	

SyntheticCommandFromIndex()

Purpose	Retrieve the synthetic command number for a particular Menu item. A synthetic command number has the MENU id in the high 16 bits and the item number in the low 16 bits, and the resulting 32-bit number is negated (to distinguish it from regular command numbers). The synthetic command number is the negative of the value that would be returned by the Toolbox trap <code>MenuSelect()</code> for the menu item.
Access	Public
Prototype	<code>CommandT SyntheticCommandFromIndex(SInt16 inIndex) const;</code>

LMenu

Parameters	The parameter for this method is:		
	SInt16	inIndex	The index for the Menu item.
Return	The command for the Menu item.		

mNextMenu

Purpose	Storage for the next menu.
Access	Protected
Prototype	LMenu *mNextMenu;

mMacMenuH

Purpose	The Mac OS menu handle.
Access	Protected
Prototype	MenuHandle mMacMenuH;

mMENUId

Purpose	The menu ID.
Access	Protected
Prototype	ResIDT mMENUId;

mNumCommands

Purpose	The number of commands.
---------	-------------------------

Access	Protected
Prototype	SInt16 mNumCommands;

mCommandNums

Purpose	The commadn numbers.
Access	Protected
Prototype	CommandT **mCommandNums ;

mIsInstalled

Purpose	Storage for detecting installation.
Access	Protected
Prototype	Boolean mIsInstalled;

mUsed

Purpose	Indicates whether used or not.
Access	Protected
Prototype	Boolean mUsed;

LMenuBar

Description	LMenuBar is a PowerPlant class that is used for managing the menu bar.	
Methods	The methods in this class are:	
	LMenuBar()	~LMenuBar()
	CouldBeKeyCommand()	FetchMenu()
	FindCommand()	FindKeyCommand()
	FindMenuItem()	FindNextCommand()
	FindNextMenu()	GetCurrentMenuBar()
	InstallMenu()	MenuCommandSelection()
	RemoveMenu()	
Data Members	The data members in this class are:	
	sMenuBar	mMenuListHead
Operation	For detailed information on how to use this class, refer to <i>The PowerPlant Book</i> .	
Source files	(Support Classes)	
	LMenuBar.h	
	LMenuBar.cp	

LMenuBar()

Purpose	The constructor creates the menu bar.
Access	Public
Prototype	<code>LMenuBar (ResIDT inMBARid);</code>
Parameters	This constructor has the following parameter:

LMenuBar

ResIDT	inMBARid	The resource ID of the menu bar.
--------	----------	----------------------------------

~LMenuBar()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LMenuBar();</code>

CouldBeKeyCommand()

Purpose	Return whether the keystroke could be a key equivalent for a menu command. Overrid this to implement keyboard equivalents that use modifier keys other than just the Command key.			
Access	Virtual, Public			
Prototype	<code>virtual Boolean CouldBeKeyCommand(const EventRecord &inKeyEvent) const;</code>			
Parameters	This method has the following parameter: <table><tr><td>const EventRecord&</td><td>inKeyEvent</td><td>The event for the key down.</td></tr></table>	const EventRecord&	inKeyEvent	The event for the key down.
const EventRecord&	inKeyEvent	The event for the key down.		
Return	This function returns true if the command key is down.			

FetchMenu()

Purpose	Return the Menu object for the specified MENU resource ID Returns nil if there is no such Menu object
Access	Public

Prototype	<code>LMenu *FetchMenu(ResIDT inMENUid) const;</code>			
Parameters	<p>This method has the following parameter:</p> <table><tr><td>ResIDT</td><td>inMENUid</td><td>The menu resource ID.</td></tr></table>	ResIDT	inMENUid	The menu resource ID.
ResIDT	inMENUid	The menu resource ID.		
Return	A pointer to an LMenu.			

FindCommand()

Purpose	Retrieve the Command number corresponding to a Menu (ID, item) pair.						
Access	Public						
Prototype	CommandT FindCommand(ResIDT inMENUid, SInt16 inItem) const;						
Parameters	<div>This method has the following parameters:</div> <table><tr><td>ResIDT</td><td>inMENUid</td><td>The menu resource ID.</td></tr><tr><td>SInt16</td><td>inItem</td><td>The menu item.</td></tr></table>	ResIDT	inMENUid	The menu resource ID.	SInt16	inItem	The menu item.
ResIDT	inMENUid	The menu resource ID.					
SInt16	inItem	The menu item.					
Return	Returns the command number.						

FindKeyCommand()

Purpose	Return the Command number corresponding to a keystroke. Call this function when CouldBeKeyCommand() is true. You should override this to implement keyboard equivalents that use modifier keys other than just the Command key.
Access	Virtual, Public
Prototype	<code>virtual CommandT FindKeyCommand(const EventRecord &inKeyEvent, SInt32 &outMenuChoice) const;</code>

LMenuBar

Parameters	<div><div>const EventRecord &</div><div>SInt32&</div><div>inKeyEvent</div><div>outMenuChoice</div><div>The key down event.</div><div>The menu choice.</div></div>
Return	The command number. Returns cmd_Nothing if the keystroke is not a menu equivalent
Remarks	This function calls the Toolbox routine MenuKey to find the associated menu item, if any. Override this function (as well as CouldBeKeyCommand) to implement key equivalents that use other modifier keys, such as Option, Shift, or Control.

FindMenuItem()

Purpose	Passes back the MENU id, MenuHandle, and item number corresponding to a Command number. If the Command is not associated with any item in the MenuBar, outMENUid is 0, outMenuHandle is nil, and outItem is 0.		
Access	Public		
Prototype	<pre>void FindMenuItem(CommandT inCommand, ResIDT &outMENUid, MenuHandle &outMenuHandle, SInt16 &outItem);</pre>		
Parameters	This method has the following parameter:		
	ResIDT	outMENUid	The menu resource ID.
	SInt16	outItem	The menu item.
	CommandT	inComman d	The command.
	MenuHandle	outMenuHa ndle	A handle to the menu.
Return	None		

Remarks	None
---------	------

FindNextCommand()

Purpose	Find the next command.
---------	------------------------

Access	Public
--------	--------

Prototype	<pre>Boolean FindNextCommand(SInt16 &ioIndex, MenuHandle &ioMenuHandle, LMenu* &ioMenu, CommandT &outCommand);</pre>
-----------	--

Parameters	This method has the following parameters:
------------	---

LMenu*	ioMenu	The menu pointer.
SInt16&	ioIndex	The index.
CommandT&	outCommand	The command.
MenuHandle&	ioMenuHandle	A handle to the menu.

Return	Return true if the command is found, else return false.
--------	---

FindNextMenu()

Purpose	Find the next menu in the menu bar.
---------	-------------------------------------

Access	Public
--------	--------

Prototype	<pre>Boolean FindNextMenu(LMenu* &ioMenu) const;</pre>
-----------	--

Parameters	This method has the following parameter:
------------	--

LMenu*&	ioMenu	The menu pointer.
---------	--------	-------------------

Return	Return true if a menu was found, else return false.
--------	---

GetCurrentMenuBar()

Purpose	Retrieve the menu bar.
Access	Static, Public
Prototype	<code>static LMenuBar* GetCurrentMenuBar();</code>
Parameters	None
Return	The pointer to the menu bar.

InstallMenu()

Purpose	Install a Menu object in the MenuBar							
Access	Public							
Prototype	<code>void InstallMenu(LMenu *inMenuToInstall, ResIDT ioMenuHandle);</code>							
Parameters	This method has the following parameters:							
	<table><tr><td>LMenu *</td><td>inMenuToInstall</td><td>The menu to install in the menu bar.</td></tr><tr><td>ResID T</td><td>ioMenuHandle</td><td>The resource ID for the menu.</td></tr></table>	LMenu *	inMenuToInstall	The menu to install in the menu bar.	ResID T	ioMenuHandle	The resource ID for the menu.	
LMenu *	inMenuToInstall	The menu to install in the menu bar.						
ResID T	ioMenuHandle	The resource ID for the menu.						
Return	None							

MenuCommandSelection()

Purpose	Handle menu selection with the Mouse and return the command number for the item chosen. Override this method to implement alternative menu selection behavior.
---------	--

Access	Virtual, Public	
Prototype	<pre>virtual CommandT MenuCommandSelection(const EventRecord &inMouseEvent, SInt32 &outMenuChoice) const;</pre>	
Parameters	This method has the following parameters:	
	const EventRecord&	The mouse event information.
	SInt32& outMenuChoice	The menu choice.
Return	The command number.	

RemoveMenu()

Purpose	Remove a Menu object from the menu bar.	
Access	Public	
Prototype	<pre>void RemoveMenu(LMenu *inMenuToRemove);</pre>	
Parameters	This method has the following parameters:	
	LMenu* inMenuToRemove	The menu to remove.
Return	None	

sMenuBar

Purpose	Storage for the menu bar pointer.
Access	Protected
Prototype	<pre>static LMenuBar *sMenuBar;</pre>

LMenuBar

mMenuListHead

Purpose The menu list head.

Access Protected

Prototype LMenu *mMenuListHead;

LMouseTracker

Overview	LMouseTracker is a PowerPlant class that is used for periodically tracking the mouse.
Methods	The methods in this class are: LMouseTracker() SpendTime() ~LMouseTracker()
Data Members	There are no data members for this class.
Operation	You can use this class to do things based on the location of the mouse pointer.
Source files	(Support Classes) LMouseTracker.h LMouseTracker.cp
Ancestors	LPeriodical

LMouseTracker()

Purpose	The constructor is a placeholder (does nothing).
Access	Public
Prototype	<code>LMouseTracker () ;</code>

~LMouseTracker()

Purpose	The destructor is a placeholder (does nothing).
Access	Virtual, Public
Prototype	<code>virtual ~LMouseTracker () ;</code>

LMouseTracker

Parameters None

SpendTime()

Purpose Track the mouse. This is an override of [SpendTime\(\)](#) in [LPeriodical](#).

LMovieController

Overview	LMovieController is a PowerPlant class that is used for creating, drawing, and disposing of a standard Mac OS QuickTime movie controller.
Methods	The methods in this class are: LMovieController() DrawSelf() ~LMovieController() SpendTime()
Data Members	The data members in this class are: mMovie mMovieController
Operation	You would typically use this class in conjunction with the UQuickTime object if your application supports QuickTime movies. Since this class inherits from LPeriodical , it receives and can process every event retrieved by the event loop.
Source files	(Pane Classes) UQuickTime.h UQuickTime.cp
Ancestors	LPane LPeriodical

LMovieController()

Purpose	The constructors create objects from the passed-in parameters.
Access	Public
Prototype	<pre>LMovieController(); LMovieController(const SPaneInfo &inPaneInfo, Movie inMovie);</pre>
Parameters	This method has the following parameters:

LMovieController

const SPaneInfo&	inPaneInfo	The reference for the SuperView that the movie controller belongs to.
Movie	inMovie	The resource ID for the movie that will be displayed with the controller.

~LMovieController()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LMovieController();</code>

DrawSelf()

Purpose	This method is an override of the base class method DrawSelf() in LPane . It draws the movie controller.
---------	--

SpendTime()

Purpose	This method is an override of the base class method DrawSelf() in LPeriodical . It handles events in the movie controller.
---------	--

mMovie

Purpose	This data member stores the resource ID of the movie for the controller.
Access	Protected

Prototype `Movie mMovie;`

mMovieController

Purpose This data member stores the MovieController.

Access Protected

Prototype `MovieController mMovieController;`

LMutexSemaphore

Overview	LMutexSemaphore is a PowerPlant class that is used for implementing a mutually-exclusive semaphore.
Methods	The methods in this class are: LMutexSemaphore() Signal() ~LMutexSemaphore() Wait()
Data Members	The data members in this class are: mOwner mNestedWaits
Operation	If a mutually-exclusive semaphore is raised, only one thread may access the flagged data. When the semaphore is lowered, if there are waiting threads, only one waiting thread is returned to the ready state.
Source files	(Threads Classes) LMutexSemaphore.h LMutexSemaphore.cp
See also	LSemaphore

LMutexSemaphore()

Purpose	The constructor creates the object.
Access	Public
Prototype	LLMutexSemaphore(); LMutexSemaphore(Boolean owned);
Parameters	Indicate whether the semaphore is already owned or not. If owned is true, the semaphore is marked as belonging to the current thread.

LMutexSemaphore

~LMutexSemaphore()

Purpose	The destructor destroys the LMutexSemaphore object.
Access	Virtual
Prototype	<code>virtual ~LMutexSemaphore();</code>

Signal()

Purpose	This is an override of Signal() in LSemaphore . If the owning semaphore called <code>Wait()</code> more than once, this function merely decrements a usage count. Else, if any threads are waiting on the semaphore, one of them is unblocked. If the current thread doesn't own the semaphore, the <code>errSemaphoreNotOwner</code> exception is thrown.
---------	--

Wait()

Purpose	This is an override of Wait() in LSemaphore .
---------	---

mOwner

Purpose	The owner.
Access	Protected
Prototype	<code>LThread*mOwner;</code>

mNestedWaits

Purpose	The number of nested waits
Access	Protected
Prototype	<code>UInt32 mNestedWaits;</code>

LNetMessageQueue

Overview	LNetMessageQueue is a PowerPlant class that is used for queuing of messages, typically generated at interrupt time, that will be broadcast to registered listeners at primary task time via a periodical.	
Methods	The methods in this class are: LNetMessageQueue() SpendTime() ~LNetMessageQueue()	
Data Members	The data members in this class are: mBroadcaster	
Source files	(Networking Classes) LNetMessageQueue.h LNetMessageQueue.cp	
See also	LBroadcaster LPeriodical	

LNetMessageQueue()

Purpose	The constructor creates the object.	
Access	Public	
Prototype	LNetMessageQueue (LBroadcaster &inBroadcaster) ;	
Parameters	The parameter for this constructor is:	
	LBroadcaster&	inBroadcaster The reference to the Broadcaster.

LNetMessageQueue

~LNetMessageQueue()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LNetMessageQueue();</code>

SpendTime()

Purpose	This is an override of SpendTime() in LPeriodical .
---------	---

mBroadcaster

Purpose	The broadcaster.
Access	Protected
Prototype	<code>LBroadcaster * mBroadcaster;</code>

LOffscreenView

Overview	LOffscreenView is a PowerPlant class that is used for offscreen drawing. It creates a temporary GWorld everytime you draw.						
Methods	<p>The methods in this class are:</p> <table><tr><td>LOffscreenView()</td><td>~LOffscreenView()</td></tr><tr><td>Draw()</td><td>DrawOffscreen()</td></tr><tr><td>EstablishPort()</td><td>SubImageChanged()</td></tr></table>	LOffscreenView()	~LOffscreenView()	Draw()	DrawOffscreen()	EstablishPort()	SubImageChanged()
LOffscreenView()	~LOffscreenView()						
Draw()	DrawOffscreen()						
EstablishPort()	SubImageChanged()						
Data Members	<p>The data members in this class are:</p> <table><tr><td>mOffscreenWorld</td><td>mDrawingSelf</td></tr></table>	mOffscreenWorld	mDrawingSelf				
mOffscreenWorld	mDrawingSelf						
Operation	If you want a view or panes within the view to use a GWorld that lasts for more than one update, use LGWorld instead.						
Source files	<p>(Pane Classes)</p> <p>LOffscreenView.h</p> <p>LOffscreenView.cp</p>						
Ancestors	<p>LAttachable</p> <p>LPane</p> <p>LView</p>						

LOffscreenView()

Purpose	The constructors create objects from the passed-in parameters.
Access	Public
Prototype	<pre>LOffscreenView(); LOffscreenView(const SPaneInfo &inPaneInfo, const SViewInfo &inViewInfo); LOffscreenView(LStream *inStream);</pre>

LOffscreenView

Parameters	The parameters for these constructors are:		
	const SPaneInfo&	inPaneInfo	The Superpane info.
	const SViewInfo&	inViewInfo	The Superview info.
	LStream*	inStream	The stream to read from.

~LOffscreenView()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LOffscreenView();</code>

Draw()

Purpose	Draw contents of View offscreen and then copy them onscreen. First, we try to draw offscreen using temporary memory. If that fails, we try to draw offscreen using application memory. If all else fails, we call the inherited version to draw directly to the screen. This is an override of Draw() in LPane .
---------	--

DrawOffscreen()

Purpose	Draw contents of View to an Offscreen World, then copy bits all at once to the normal port. Use the given flags when creating the offscreen GWorld.
Access	Virtual, Protected

Prototype	<code>virtual void DrawOffscreen(RgnHandle inSuperDrawRgnH, GWorldFlags inFlags);</code>		
Parameters	This method has the following parameter:		
	RgnHandle	inSuperDrawRgnH	The region handle.
	GWorldFlags	inFlags	The flags for the GWorld creation.
Return	None		

EstablishPort()

Purpose	Set current port to the OffscreenView. If this View is being drawn, the current port should be the Offscreen GWorld. Otherwise, some subpane is trying to draw itself directly (not in response to an update event) so we call the inherited function to do the normal thing.		
Access	Virtual, Public		
Prototype	<code>virtual Boolean EstablishPort();</code>		
Parameters	None		
Return	None		

SubImageChanged()

Purpose	Notification that the Image of some SubView changed size, location, or scroll units. In this override, we pass the notification up the chain. This is because the LOffscreenView isn't really the logical superview of its subpanes; it's just in the hierarchy to draw offscreen. So, we pass the notification up to the logical superview of its subpanes.		
---------	--	--	--

LOffscreenView

Access	Virtual, Public				
Prototype	<code>virtual void SubImageChanged(LView* inSubView);</code>				
Parameters	This method has the following parameter:				
	<table><tr><td><code>LView*</code></td><td><code>inSubView</code></td><td>The pointer to the view.</td></tr></table>	<code>LView*</code>	<code>inSubView</code>	The pointer to the view.	
<code>LView*</code>	<code>inSubView</code>	The pointer to the view.			
Return	None				

mOffscreenWorld

Purpose	The GWorld pointer.
Access	Protected
Prototype	<code>GWorldPtr mOffscreenWorld;</code>

mDrawingSelf

Purpose	Storage for the information about self drawing.
Access	Protected
Prototype	<code>Boolean mDrawingSelf;</code>

LOperationListMember

Overview	LOperationListMember is a PowerPlant class that simplifies operation list management.
Methods	The methods in this class are: LOperationListMember() ~LOperationListMember()
Data Members	The data members in this class are: mOperation
Source files	(Networking Classes) LNetworking.h LNetworking.cp
See also	StAsyncOperation

LOperationListMember()

Purpose	The constructor creates the object.
Access	Public
Prototype	<pre>LOperationListMember(StAsyncOperation *inOperation);</pre>
Parameters	This constructor has the following parameter: StAsyncOperation* inOperation The operation.

~LOperationListMember()

Purpose	The destructor destroys the object.
Access	Virtual, Public

LOperationListMember

Prototype `virtual ~LOperationListMember();`

mOperation

Purpose The operation.

Access Static

Prototype `StAsyncOperation *mOperation;`

LOverlappingView

Overview	LOverlappingView is a PowerPlant class that allows you to create a container (superview) for another view. LOverlappingView is a View which draws properly when sibling views (views within the same superview) overlap its Frame.
Methods	The methods in this class are: LOverlappingView() ~LOverlappingView() FocusDraw()
Data Members	There are no data members in this class.
Operation	Use LOverlappingView as a container (superview) for another view.
Source files	(Pane Classes) LOverlappingView.h LOverlappingView.cp
Ancestors	LAttachable LPane LView

LOverlappingView()

Purpose	The constructor creates the object.		
Access	Public		
Prototype	LOverlappingView(); LOverlappingView(LStream *inStream);		
Parameters	These constructors have the following parameter:		
	LStream*	inStream	The stream to read from.

LOverlappingView

~LOverlappingView()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LOverlappingView();</code>

FocusDraw()

Purpose	Set up coordinates system and clipping region. An LOverlappingView sets the clipping region to the revealed portion of its Frame minus the Frames of any sibling Views that are in front of it. This method is an override of FocusDraw() in LPane .
---------	--

LPaintAttachment

Overview	LPaintAttachment is a PowerPlant class that is used for painting rectangles within the Frame of a Pane.
Methods	The methods in this class are: LPaintAttachment() ExecuteSelf()
Data Members	The data members in this class are: mPenState mForeColor mBackColor
Operation	<p>This objects allows you to paint a rectangle within the Frame of a Pane, for use only with the <code>msg_DrawOrPrint</code> message.</p> <p>The rectangle is painted using the specified <code>PenState</code> settings and foreground and background <code>RGBColor</code> values. The painted rectangle is inset from the Frame by the size of the <code>pnSize</code> field of the <code>PenState</code>. This lets you use an <code>LPaintAttachment</code> in conjunction with a LBorderAttachment to draw a filled rectangle.</p>
Source files	(Utility Classes) <code>UAttachments.h</code> <code>UAttachments.cp</code>
See also	LAttachment LBorderAttachment LPane

LPaintAttachment()

Purpose	The constructors provide two ways to create <code>LPaintAttachment</code> objects. You may create a stream-based object or pass the itemized values for each required value.
---------	--

LPaintAttachment

Access	Public		
Prototype	<pre>LPaintAttachment(PenState *inPenState, RGBColor *inForeColor, RGBColor *inBackColor, Boolean inExecuteHost); LPaintAttachment(LStream *inStream);</pre>		
Parameters	The parameters for these constructors are:		
	PenState*	inPenState	A pointer to a PenState.
	RGBColor*	inForeColor	A pointer to an RGBColor.
	RGBColor*	inBackColor	A pointer to an RGBColor.
	Boolean	inExecuteHost	A Boolean indicating the value to be set for mExecuteHost .
	LStream*	inStream	A pointer to a stream object that contains the information to create the LPaintAttachment object.

ExecuteSelf()

Purpose	This method paints a rectangle. This is an override of ExecuteSelf() in LAttachment .
---------	---

mPenState

Purpose	This member holds the pen information.
Access	Protected
Prototype	<pre>PenState mPenState;</pre>

mForeColor

Purpose	This member holds the foreground color information.
Access	Protected
Prototype	<code>RGBColor mForeColor;</code>

mBackColor

Purpose	This member holds the background color information.
Access	Protected
Prototype	<code>RGBColor mBackColor;</code>

LPane

Overview LPane is the fundamental visual class in PowerPlant. Every area you draw in, and everything you draw, descends from LPane. There are LPane derivative classes for text, pictures, lists, tables, controls and more.

Methods The methods in this class are:

<u>LPane()</u>	<u>~LPane()</u>
<u>Activate()</u>	<u>ActivateSelf()</u>
<u>AdaptToNewSurroundings()</u>	<u>AdaptToSuperFrameSize()</u>
<u>AdaptToSuperScroll()</u>	<u>AdjustCursor()</u>
<u>AdjustCursorSelf()</u>	<u>ApplyForeAndBackColors()</u>
<u>CalcLocalFrameRect()</u>	<u>CalcPortFrameRect()</u>
<u>Click()</u>	<u>ClickSelf()</u>
<u>ClickTimesAreClose()</u>	<u>Contains()</u>
<u>CountPanels()</u>	<u>Deactivate()</u>
<u>DeactivateSelf()</u>	<u>Disable()</u>
<u>DisableSelf()</u>	<u>DontRefresh()</u>
<u>Draw()</u>	<u>DrawSelf()</u>
<u>Enable()</u>	<u>EnableSelf()</u>
<u>EventMouseUp()</u>	<u>FindConstPaneByID()</u>
<u>FindDeepSubPaneContaining()</u>	<u>FindPaneByID()</u>
<u>FindShallowSubPaneContaining()</u>	<u>FindSubPaneHitBy()</u>
<u>FinishCreate()</u>	<u>FinishCreateSelf()</u>
<u>FocusDraw()</u>	<u>FocusExposed()</u>
<u>GetActiveState()</u>	<u>GetClickCount()</u>
<u>GetDefaultView()</u>	<u>GetDescriptor()</u>
<u>GetEnabledState()</u>	<u>GetFrameBinding()</u>

<u>GetFrameLocation()</u>	<u>GetFrameSize()</u>
<u>GetLastPaneClicked()</u>	<u>GetLastPaneMoused()</u>
<u>GetLocalUpdateRgn()</u>	<u>GetMacPort()</u>
<u>GetPaneID()</u>	<u>GetSuperView()</u>
<u>GetUserCon()</u>	<u>GetValue()</u>
<u>GetVisibleState()</u>	<u>GlobalToPortPoint()</u>
<u>Hide()</u>	<u>HideSelf()</u>
<u>InitPane()</u>	<u>InvalPortRect()</u>
<u>InvalPortRgn()</u>	<u>IsActive()</u>
<u>IsAreaInQDSpace()</u>	<u>IsEnabled()</u>
<u>IsHitBy()</u>	<u>IsVisible()</u>
<u>LocalToPortPoint()</u>	<u>MouseEnter()</u>
<u>MouseLeave()</u>	<u>MouseWithin()</u>
<u>MoveBy()</u>	<u>PlaceInSuperFrameAt()</u>
<u>PlaceInSuperImageAt()</u>	<u>PointIsInFrame()</u>
<u>PointsAreClose()</u>	<u>PortToGlobalPoint()</u>
<u>PortToLocalPoint()</u>	<u>PrintPanel()</u>
<u>PrintPanelSelf()</u>	<u>PutInside()</u>
<u>Refresh()</u>	<u>ResizeFrameBy()</u>
<u>ResizeFrameTo()</u>	<u>RestorePlace()</u>
<u>SavePlace()</u>	<u>ScrollToPanel()</u>
<u>SetDefaultView()</u>	<u>SetDescriptor()</u>
<u>SetForeAndBackColors()</u>	<u>SetFrameBinding()</u>
<u>SetLastPaneMoused()</u>	<u>SetPaneID()</u>
<u>SetRefreshAllWhenResized()</u>	<u>SetUserCon()</u>
<u>SetValue()</u>	<u>Show()</u>
<u>ShowSelf()</u>	<u>SuperActivate()</u>

SuperDeactivate()	SuperDisable()
SuperEnable()	SuperHide()
SuperPrintPanel()	SuperShow()
UpdateClickCount()	UpdatePort()
ValidPortRect()	ValidPortRgn()

Data Members The data members in this class are:

sDefaultView	sLastPaneClicked
sLastPaneMoused	sWhenLastMouseUp
sWhenLastMouseDown	sWhereLastMouseDown
sClickCount	mPaneID
mFrameSize	mFrameLocation
mFrameBinding	mUserCon
mSuperView	mVisible
mActive	mEnabled
mRefreshAllWhenResized	

Operation LPane is one of the central classes to the magic of PowerPlant. In order to learn how to use it effectively, you should read about it in *The PowerPlant Book* on your documentation CD.

Source files (Pane Classes)

`LPane.h`

`LPane.cp`

See also [LAttachable](#)

[LView](#)

LPane

LPane()

Purpose	The constructor creates the object using the passed-in parameters.		
Access	Public		
Prototype	<pre>LPane(); LPane(const LPane &inOriginal); LPane(const SPaneInfo &inPaneInfo); LPane(LStream *inStream);</pre>		
Parameters	The parameters for these constructors are:		
	const LPane&	inOriginal	The object to copy.
	const SPaneInfo&	inPaneInfo	The Superpane's info.
	LStream*	inStream	The stream to read from.

~LPane()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<pre>virtual ~LPane();</pre>

Activate()

Purpose	Activate a pane.
Access	Virtual, Public
Prototype	<pre>virtual void Activate();</pre>
Parameters	None

Return	None
--------	------

ActivateSelf()

Purpose	This is called when a pane is being activated. You should override this method for pane classes that change appearance when activated.
Access	Virtual, Protected
Prototype	<code>virtual void ActivateSelf();</code>
Parameters	None
Return	None

AdaptToNewSurroundings()

Purpose	Adjust the state of the pane when installed in a new parent pane.
Access	Virtual, Public
Prototype	<code>virtual void AdaptToNewSurroundings();</code>
Parameters	None
Return	None

AdaptToSuperFrameSize()

Purpose	Adjust the state of the pane when the size of the superview's frame changes by the specified amounts.
Access	Virtual, Public
Prototype	<code>virtual void AdaptToSuperFrameSize(SInt32 inSurrWidthDelta,</code>

```
SInt32 inSurrHeightDelta,  
Boolean inRefresh);
```

Parameters The parameters for this method are:

SInt32	inSurrWidthDelta	The change in width.
SInt32	inSurrHeightDelta	The change in height.
Boolean	inRefresh	Indicate whether to refresh or not.

Return None

AdaptToSuperScroll()

Purpose Adjust the state of the pane when its superview scrolls by the specified amounts.

Access Virtual, Public

Prototype

```
virtual void AdaptToSuperScroll(  
SInt32 inHorizScroll,  
SInt32 inVertScroll);
```

Parameters The parameters for this method are:

SInt32	inHorizScroll	The width amount to scroll.
SInt32	inVertScroll	The height amount to scroll.

Return None

AdjustCursor()

Purpose Wrapper function for setting the cursor shape.

Access Virtual, Public

Prototype	<pre>virtual void AdjustCursor(Point inPortPt, const EventRecord &inMacEvent);</pre>		
Parameters	The parameters for this method are:		
	Point	inPortPt	The point in Port coordinates.
	const EventRecord &	inMacEvent	The event record.
Return	None		

AdjustCursorSelf()

Purpose	Set the cursor shape when the cursor is inside a pane.		
Access	Virtual, Public		
Prototype	<pre>virtual void AdjustCursorSelf(Point inPortPt, const EventRecord& inMacEvent);</pre>		
Parameters	The parameters for this method are:		
	Point	inPortPt	The point in Port coordinates.
	const EventRecord &	inMacEvent	The event record. Use the inMacEvent->modifiers if the cursor shape depends on whether modifier keys (such as Option) are down.
Return	None		

ApplyForeAndBackColors()

Purpose	Set the foreground and background colors of the current port. The pane must already be focused.
---------	---

LPane

Access	Virtual, Protected
Prototype	<code>virtual void ApplyForeAndBackColors() const;</code>
Parameters	None
Return	None

CalcLocalFrameRect()

Purpose	Calculate the pane's frame rectangle in local (Image) coordinates.		
Access	Virtual, Public		
Prototype	<code>virtual Boolean CalcLocalFrameRect(Rect &outRect) const;</code>		
Parameters	The parameter for this method is:		
	Rect&	outRect	The Rect to calculate the pane's frame rectangle from.
Return	Returns <code>true</code> if the frame is within QuickDraw space (16-bit), and returns <code>false</code> if the frame is outside QuickDraw space with <code>outRect</code> is unchanged.		

CalcPortFrameRect()

Purpose	Calculate the pane's frame rectangle in Port coordinates.		
	Returns <code>true</code> if the frame is within QuickDraw space (16-bit) Returns <code>false</code> if the frame is outside QuickDraw space and <code>outRect</code> is unchanged		
Access	Virtual, Public		
Prototype	<code>virtual Boolean CalcPortFrameRect(Rect &outRect) const;</code>		
Parameters	The parameter for this method is:		

	Rect&	outRect	The Rect to calculate the pane's frame rectangle from.
Return	Returns <code>true</code> if the frame is within QuickDraw space (16-bit), and returns <code>false</code> if the frame is outside QuickDraw space with <code>outRect</code> is unchanged.		

Click()

Purpose	This is a wrapper method for handling a click inside a pane. This method does some bookkeeping, executes Attachments, then calls ClickSelf() to actually respond to the click.		
Access	Virtual, Public		
Prototype	<code>virtual void Click(SMouseDownEvent &inMouseDown);</code>		
Parameters	The parameter for this method is: SMouseDownEvent& inMouseDown The mouse down event.		
Return	None		

ClickSelf()

Purpose	Respond to click inside this pane. You should override this to do something when the user clicks inside a pane.		
Access	Virtual, Public		
Prototype	<code>virtual void ClickSelf(const SMouseDownEvent& inMouseDown);</code>		
Parameters	The parameter for this method is:		

	<code>SMouseDownEvent&</code>	<code>inMouseDown</code>	The mouse down event.
Return	None		

ClickTimesAreClose()

Purpose	Indicate whether the time between the specified time and the time of the last mouse down are close enough to be considered a double-click.		
Access	Virtual, Public		
Prototype	<code>virtual Boolean ClickTimesAreClose(UInt32 inLastClickTime) const;</code>		
Parameters	The parameter for this method is: <code>UInt32</code> <code>inLastClickTime</code> The mouse down event.		
Return	Returns true if a double-click, else false.		

Contains()

Purpose	Indicates whether a pane contains the specified point, which is in Port coordinates. Note: This function just calls PointIsInFrame() , but it exists so that you can override it to change how the FindDeepSubPaneContaining() and FindShallowSubPaneContaining() search for Panes.		
Access	Virtual, Public		
Prototype	<code>virtual Boolean Contains(SInt32 inHorizPort, SInt32 inVertPort) const;</code>		
Parameters	The parameters for this method are:		

	SInt32	inHorizPort	The horizontal coordinate.
	SInt32	inVertPort	The vertical coordinate.
Return	Returns true if the pane contains the point, else returns false.		

CountPanels()

Purpose	Indicate the number of horizontal and vertical panels. A panel is a "frameful" of a pane.		
Access	Virtual, Public		
Prototype	<pre>virtual void CountPanels(UInt32 &outHorizPanels, UInt32 &outVertPanels);</pre>		
Parameters	The parameters for this method are:		
	UInt32	outHorizPanels	The number of horizontal panels.
	UInt32	outVertPanels	The number of vertical panels.
Return	None		

Deactivate()

Purpose	Deactivate a pane.		
Access	Virtual, Public		
Prototype	<pre>virtual void Deactivate();</pre>		
Parameters	None		
Return	None		

DeactivateSelf()

Purpose	This is called when a pane is being deactivated. You should override this for pane classes that change appearance when deactivated.
Access	Virtual, Protected
Prototype	<code>virtual void DeactivateSelf();</code>
Parameters	None
Return	None

Disable()

Purpose	Disable a pane.
Access	Public
Prototype	<code>virtual void Disable();</code>
Parameters	None
Return	None

DisableSelf()

Purpose	Pane is being disabled. You should override this for pane classes that change appearance when disabled. To avoid flicker, a pane can draw immediately from this method rather than refreshing and drawing at the next update event.
Access	Virtual, Protected
Prototype	<code>virtual void DisableSelf();</code>
Parameters	None

Return None

DontRefresh()

Purpose	Validate the area occupied by a pane. This removes the pane area from the update region so that the pane won't be redrawn during the next update event.		
Access	Virtual, Public		
Prototype	<code>virtual void DontRefresh(Boolean inOKIfHidden);</code>		
Parameters	<p>The parameter for this method is:</p> <table border="0" style="margin-left: 40px;"> <tr> <td style="vertical-align: top; padding-right: 20px;"><code>Boolean inOKIfHidden</code></td> <td>Specifies whether to validate even if the pane is not visible. Pass true for this parameter to suppress the automatic refresh which occurs after hiding a pane.</td> </tr> </table>	<code>Boolean inOKIfHidden</code>	Specifies whether to validate even if the pane is not visible. Pass true for this parameter to suppress the automatic refresh which occurs after hiding a pane.
<code>Boolean inOKIfHidden</code>	Specifies whether to validate even if the pane is not visible. Pass true for this parameter to suppress the automatic refresh which occurs after hiding a pane.		
Return	None		

Draw()

Purpose	<p>Try to draw the contents of a pane.</p> <p>This is a wrapper function which calls DrawSelf() if it is proper for the pane to draw. This means that:</p> <ul style="list-style-type: none"> • pane is visible • pane's frame is in QuickDraw space • pane's frame intersects inSuperDrawRgnH • pane can be focused
Access	Virtual, Public
Prototype	<code>virtual void Draw(RgnHandle inSuperDrawRgnH);</code>

LPane

Parameters	The parameter for this method is:		
	<code>RgnHandle</code>	<code>inSuperDrawRgnH</code>	Specifies, in Port coordinates, the portion of the pane's superview that needs to be drawn. Specify <code>nil</code> to bypass the intersection test.
Return	None		

DrawSelf()

Purpose	<p>This draws a box around the pane and a diagonal line from the top left to the bottom right corner using the default foreground and background colors for the pane.</p> <p>On entry, the clipping region is the revealed area of the pane's superview. Therefore, it is possible for a pane to draw outside of its frame. You will not normally do this.</p> <p>The port, coordinate system, and clipping region are set on entry. They must be the same upon exit. You are responsible for setting the Pen state and text characteristics to the proper values for your pane.</p>		
Access	Virtual, Protected		
Prototype	<code>virtual void DrawSelf();</code>		
Parameters	None		
Return	None		

Enable()

Purpose	Enable a pane.
Access	Virtual, Public
Prototype	<code>virtual void Enable();</code>

Parameters	None
Return	None

EnableSelf()

Purpose	Pane is being enabled. You should override this for pane classes that change appearance when enabled. To avoid flicker, a pane can draw immediately from this function rather than refreshing and drawing at the next update event.		
Access	Virtual, Protected		
Prototype	<code>virtual void EnableSelf();</code>		
Parameters	None		
Return	None		

EventMouseUp()

Purpose	Respond to a mouse up event following a click (mouse down) inside a pane.		
Access	Virtual, Public		
Prototype	<code>virtual void EventMouseUp(const EventRecord &inMouseUp);</code>		
Parameters	The parameter for this method is:		
	const EventRecord&	inMouseUp	The event info for the mouse up.
Return	None		

FindConstPaneByID()

Purpose	Find the pane of a View which has the specified ID. This method searches all Panes contained within this View, not just direct subpanes. Returns nil if pane with the target ID is not found.			
	Use when you only want to inspect (and not change) the pane.			
Access	Virtual, Public			
Prototype	<pre>virtual const LPane* FindConstPaneByID(PaneIDT inPaneID) const;</pre>			
Parameters	The parameter for this method is: <table><tr><td>PaneIDT</td><td>inPaneID</td><td>The resource ID for the pane.</td></tr></table>	PaneIDT	inPaneID	The resource ID for the pane.
PaneIDT	inPaneID	The resource ID for the pane.		
Return	A pointer to the pane.			

FindDeepSubPaneContaining()

Purpose	Return the most deeply nested subpane which contains the specified point, which is in Port coordinates.								
Access	Virtual, Public								
Prototype	virtual LPane* FindDeepSubPaneContaining(SInt32 inHorizPort, SInt32 inVertPort) const;								
Parameters	The parameters for this method are: <table><tr><td>SInt32</td><td>inHorizPort</td><td>The horizontal coordinate.</td></tr><tr><td>SInt32</td><td>inVertPort</td><td>The vertical coordinate.</td></tr></table>			SInt32	inHorizPort	The horizontal coordinate.	SInt32	inVertPort	The vertical coordinate.
SInt32	inHorizPort	The horizontal coordinate.							
SInt32	inVertPort	The vertical coordinate.							
Return	Return nil if no subpane contains the point, else returns a pointer to the pane.								

FindPaneByID()

Purpose	Find the pane of a View which has the specified ID. This method searches all Panes contained within this View, not just direct subpanes.
Access	Virtual, Public
Prototype	<code>virtual LPane* FindPaneByID(PaneIDT inPaneID);</code>
Parameters	The parameter for this method is: <div style="margin-left: 40px;"> <code>PaneIDT</code> <code>inPaneID</code> The resource ID for the pane. </div>
Return	Returns <code>nil</code> if pane with the target ID is not found.

FindShallowSubPaneContaining()

Purpose	Return the immediate subpane which contains the specified point, which is in Port coordinates. Same as FindSubPaneHitBy() except that it tests using Contains() rather than IsHitBy() . <code>Contains()</code> does not check if the pane is enabled.
Access	Virtual, Public
Prototype	<code>virtual LPane* FindShallowSubPaneContaining(SInt32 inHorizPort, SInt32 inVertPort) const;</code>
Parameters	The parameters for this method are: <div style="margin-left: 40px;"> <code>SInt32</code> <code>inHorizPort</code> The horizontal coordinate. <code>SInt32</code> <code>inVertPort</code> The vertical coordinate. </div>
Return	Return <code>nil</code> if no subpane contains the point, else returns a pointer to the pane.

FindSubPaneHitBy()

Purpose	Find the subpane of this pane that is hit by the specified point.	
Access	Virtual, Public	
Prototype	<pre>virtual LPane* FindSubPaneHitBy(SInt32 inHorizPort, SInt32 inVertPort) const;</pre>	
Parameters	The parameters for this method are:	
	<pre>SInt32 inHorizPort</pre>	The horizontal coordinate.
	<pre>SInt32 inVertPort</pre>	The vertical coordinate.
Return	Return nil if no subpane is hit, else return a pointer to the pane.	
Remarks	<code>inHorizPort</code> and <code>inVertPort</code> are in Port coordinates.	

FinishCreate()

Purpose	Wrapper function for FinishCreateSelf() . You will rarely want to override this function. Subpanes are told to <code>FinishCreateSelf</code> <i>before</i> their superview. Therefore, a superview is assured that all its subpanes have finished creating when its <code>FinishCreateSelf</code> function gets called.	
Access	Virtual, Public	
Prototype	<pre>virtual void FinishCreate();</pre>	
Parameters	None	
Return	None	

FinishCreateSelf()

Purpose	Override this method to perform finishing touches that depend on the entire pane hierarchy being constructed. For example, if a View wants to store a pointer to a subpane, it should override this function to call FindPaneByID() for that pane. This saves the overhead of repeatedly calling <code>FindPaneByID()</code> when the View wants to access that subpane. You can't do this from a constructor because subpanes are created after their superview.
Access	Virtual, Protected
Prototype	<code>virtual void FinishCreateSelf();</code>
Parameters	None
Return	None

FocusDraw()

Purpose	Prepare for drawing in the pane. Throws an exception if the pane has no superview. A pane does not have its own coordinate system and clipping region. This method relies on its superview to set the focus. Even if FocusDraw() returns false, the pane's GrafPort and clipping region of its superview will be set (unless pane has no superview)
Access	Virtual, Public
Prototype	<code>virtual Boolean FocusDraw(LPane* inSubPane);</code>
Parameters	The parameter for this method is: LPane* inSubPane The pointer to the Subpane.
Return	Returns true if superview is focused.

FocusExposed()

Purpose Prepare for drawing in the pane.

A pane is exposed when all the following conditions are true:

- pane's "visible" flag is on
- pane has a superview
- superview is exposed
- pane's frame intersects exposed area of its superview

Since a pane does not have its own coordinate system, focusing a pane sets up the port, coordinate system, and clipping region of its superview.

Access Virtual, Public

Prototype `virtual Boolean FocusExposed(
Boolean inAlwaysFocus);`

Parameters The parameter for this method is:

<code>Boolean inAlwaysFocus</code>	Specifies whether to focus the pane even if it is not exposed. pane is not focused if it is not exposed <i>and</i> <code>inAlwaysFocus</code> is false.
------------------------------------	---

Return Returns `true` if a View is exposed, meaning that it is visible and that it is revealed through the Frames of all its SuperViews.

GetActiveState()

Purpose Indicate the state of the pane, stored in the [mActive](#) data member.

Access Public

Prototype `ETriState GetActiveState() const;`

Parameters None

Return	The state of the pane.
--------	------------------------

GetClickCount()

Purpose	Indicate the value of the sClickCount data member.
Access	Inline, Static, Public
Prototype	<code>static SInt16 GetClickCount();</code>
Parameters	None
Return	The value of sClickCount .

GetDefaultView()

Purpose	Returns the value of sDefaultView .
Access	Inline, Static, Public
Prototype	<code>static LView* GetDefaultView();</code>
Parameters	None
Return	A pointer to the LView.

GetDescriptor()

Purpose	Returns the descriptor string for a pane.			
Access	Virtual, Public			
Prototype	<code>virtual StringPtr GetDescriptor(Str255 outDescriptor) const;</code>			
Parameters	The parameter for this method is: <table><tr><td>Str255</td><td>outDescriptor</td><td>The descriptor string.</td></tr></table>	Str255	outDescriptor	The descriptor string.
Str255	outDescriptor	The descriptor string.		

Return A pointer to the descriptor string.

GetEnabledState()

Purpose Returns the value of the [mEnabled](#) data member.

Access Public

Prototype `ETriState GetEnabledState() const;`

Parameters None

Return Returns the value of [mEnabled](#).

GetFrameBinding()

Purpose Retrieves the value of [mFrameBinding](#).

Access Public

Prototype `void GetFrameBinding(
 SBooleanRect &outFrameBinding) const;`

Parameters The parameter for this method is:

 `SBooleanRect& outFrameBinding The frame Rect.`

Return None

GetFrameLocation()

Purpose Get the location of a pane's frame, in 32-bit Port coordinates.

Access Public

Prototype `void GetFrameLocation(
 SPoint32 &outLocation) const;`

Parameters	The parameter for this method is:		
	SPoint32&	outLocation	The location of the frame.
Return	None		

GetFrameSize()

Purpose	Get the width and height, in pixels, of the frame of a pane		
Access	Public		
Prototype	<pre>void GetFrameSize(SDimension16 &outSize) const;</pre>		
Parameters	The parameter for this method is:		
	SDimension16&	outSize	The size of the frame.
Return	None		

GetLastPaneClicked()

Purpose	Returns the value of the sLastPaneClicked data member.		
Access	Static, Public		
Prototype	<pre>static LPane* GetLastPaneClicked();</pre>		
Parameters	None		
Return	A pointer to the last pane clicked.		

GetLastPaneMoused()

Purpose	Returns the value of the sLastPaneMoused data member.		
Access	Static, Public		

LPane

Prototype	<code>static LPane* GetLastPaneMoused();</code>
Parameters	None
Return	A pointer to the last pane moused.

GetLocalUpdateRgn()

Purpose	Return the region being updated. The region will be empty if called outside of a Draw() operation
Access	Virtual, Public
Prototype	<code>virtual RgnHandle GetLocalUpdateRgn();</code>
Parameters	None
Return	A handle the the region.
Remarks	This region is non-empty only within the scope of a <code>Draw()</code> operation. Call this function from a DrawSelf() function if you want to know the actual area that requires redrawing. Most of the time, you will just draw everything and let the clipping region mask out unnecessary drawing. However, if your drawing is complex, your program might run faster if you only draw the portion which is inside the update region.

GetMacPort()

Purpose	Return the Mac OS GrafPort.
Access	Virtual, Public
Prototype	<code>virtual GrafPtr GetMacPort() const;</code>
Parameters	None
Return	A pointer to the Mac OS GrafPort.

GetPanelID()

Purpose	Returns the value of the mPanelID data member.
Access	Public
Prototype	<code>PaneIDT GetPanelID() const;</code>
Parameters	None
Return	The value of mPanelID .

GetSuperView()

Purpose	Returns the value of mSuperView .
Access	Public
Prototype	<code>LView* GetSuperView() const;</code>
Parameters	None
Return	A pointer to the LView held in mSuperView .

GetUserCon()

Purpose	Return the value of mUserCon .
Access	Virtual, Public
Prototype	<code>virtual SInt32 GetUserCon() const;</code>
Parameters	None
Return	Return the value of mUserCon .

GetValue()

Purpose	Returns the value for the pane. You should override this if you want this method to return something other than 0.
Access	Virtual, Public
Prototype	<code>virtual SInt32 GetValue() const;</code>
Parameters	None
Return	The pane value.

GetVisibleState()

Purpose	Return the value of the mVisible data member.
Access	Public
Prototype	<code>ETriState GetVisibleState() const;</code>
Parameters	None
Return	The value of mVisible .

GlobalToPortPoint()

Purpose	Convert a global point to a Port point.					
Access	Virtual, Public					
Prototype	<code>virtual void GlobalToPortPoint(Point &ioPoint) const;</code>					
Parameters	The parameter for this method is: <table><tr><td>Point&</td><td>ioPoint</td><td>The Point.</td></tr></table>			Point&	ioPoint	The Point.
Point&	ioPoint	The Point.				
Return	None					

Hide()

Purpose	Make a pane invisible.
Access	Virtual, Public
Prototype	<code>virtual void Hide();</code>
Parameters	None
Return	None

HideSelf()

Purpose	Pane is being made invisible. You should override this method for pane classes that need to be informed when they are made invisible
Access	Virtual, Protected
Prototype	<code>virtual void HideSelf();</code>
Parameters	None
Return	None

InitPane()

Purpose	Initializer for the pane.
Access	Protected
Prototype	<code>void InitPane(const SPaneInfo &inPaneInfo);</code>
Parameters	The parameter for this method is: <div style="display: flex; justify-content: space-between; align-items: center; margin-top: 5px;"> <div style="text-align: center;"> <code>const SPaneInfo&</code> </div> <div style="text-align: center;"> <code>inPaneInfo</code> </div> <div style="text-align: right;"> The Superpane info. </div> </div>
Return	None

InvalPortRect()

Purpose	Invalidate the Port rectangle. All rectangles and regions must be in Port coordinates.		
Access	Virtual, Public		
Prototype	<code>virtual void InvalPortRect(const Rect *inRect);</code>		
Parameters	The parameter for this method is:		
	<code>const Rect *</code>	<code>inRect</code>	The Rect.
Return	None		
Remarks	You should use these routines rather than the Toolbox traps <code>InvalRect</code> , <code>InvalRgn</code> , <code>ValidRect</code> , and <code>ValidRgn</code> . Those traps require that the current <code>GrafPort</code> be a Window. However, a pane could be in another kind of <code>GrafPort</code> , such as a Printer Port or <code>GWorld</code> , where calling one of those traps would cause a crash (when the Toolbox tries to access a nonexistent update region).		

InvalPortRgn()

Purpose	Invalidate the Port region.		
Access	Virtual, Public		
Prototype	<code>virtual void InvalPortRgn(RgnHandle inRgnH);</code>		
Parameters	The parameter for this method is:		
	<code>RgnHandle</code>	<code>inRgnH</code>	The region handle.
Return	None		

IsActive()

Purpose	Indicates whether a pane is active. A pane is "active" if its active flag is on and it is visible.
Access	Public
Prototype	<code>Boolean IsActive() const;</code>
Parameters	None
Return	Returns <code>true</code> if the pane is active, else <code>false</code> .

IsAreaInQDSpace()

Purpose	Indicates whether an area is inQuickDraw space.		
Access	Static, Public		
Prototype	<pre>static Boolean IsAreaInQDSpace(SInt32 inLeft, SInt32 inTop, SInt16 inWidth, SInt16 inHeight);</pre>		
Parameters	The parameters for this method are:		
	<code>SInt32</code>	<code>inLeft</code>	The left coordinate.
	<code>SInt32</code>	<code>inTop</code>	The top coordinate.
	<code>SInt16</code>	<code>inWidth</code>	The width.
	<code>SInt16</code>	<code>inHeight</code>	The height.
Return	Returns <code>true</code> if the area is in the space, else <code>false</code> .		

IsEnabled()

Purpose	Return whether a pane is enabled. A pane is "enabled" if its enabled flag is on and it is visible.
Access	Public
Prototype	<code>Boolean IsEnabled() const;</code>
Parameters	None
Return	Returns <code>true</code> if the pane is enabled, else returns <code>false</code> .

IsHitBy()

Purpose	Indicates whether a pane is hit by the specified point, which is in Port coordinates. A pane is hit if the point is inside its frame and the pane is enabled.								
Access	Virtual, Public								
Prototype	<pre>virtual Boolean IsHitBy(SInt32 inHorizPort, SInt32 inVertPort);</pre>								
Parameters	The parameters for this method are: <table><tr><td>SInt32</td><td>inHorizPort</td><td>The horizontal coordinate.</td></tr><tr><td>SInt32</td><td>inVertPort</td><td>The vertical coordinate.</td></tr></table>			SInt32	inHorizPort	The horizontal coordinate.	SInt32	inVertPort	The vertical coordinate.
SInt32	inHorizPort	The horizontal coordinate.							
SInt32	inVertPort	The vertical coordinate.							
Return	Returns true if the pane is hit by the specified point, else returns false.								

IsVisible()

Purpose	Indicates whether a pane is visible. A pane is "visible" if its visible flag is on.
---------	---

Access	Public
Prototype	<code>Boolean isVisible() const;</code>
Parameters	None
Return	Returns <code>true</code> if the pane is visible.

LocalToPortPoint()

Purpose	Convert point from Local to Port coordinates.		
Access	Virtual, Public		
Prototype	<code>virtual void LocalToPortPoint(Point &ioPoint) const;</code>		
Parameters	The parameter for this method is:		
	Point &	ioPoint	The point to convert (conversion is in place).
Return	None		

MouseEnter()

Purpose	This method could be overridden to provide useful functionality when the mouse pointer enters the pane.		
Access	Virtual, Public		
Prototype	<code>virtual void MouseEnter(Point inPortPt, const EventRecord& inMacEvent);</code>		
Parameters	The parameter for this method is:		
	const EventRecord&	inMouseUp	The event info for the mouse up.

Return	None
--------	------

MouseLeave()

Purpose	This method could be overridden to provide useful functionality when the mouse pointer leaves the pane.
Access	Virtual, Public
Prototype	<code>virtual void MouseLeave();</code>
Parameters	None
Return	None

MouseWithin()

Purpose	Override this method to provide useful functionality when the mouse pointer is within the pane.						
Access	Virtual, Public						
Prototype	<code>virtual void MouseWithin(Point inPortPt, const EventRecord& inMacEvent);</code>						
Parameters	The parameters for this method are: <table><tr><td>Point</td><td>inPortPt</td><td>The port point.</td></tr><tr><td>const EventRecord&</td><td>inMacEvent</td><td>The event info.</td></tr></table>	Point	inPortPt	The port point.	const EventRecord&	inMacEvent	The event info.
Point	inPortPt	The port point.					
const EventRecord&	inMacEvent	The event info.					
Return	None						

MoveBy()

Purpose	Move the location of the frame by the specified amounts. This specifies, in pixels, how far to move the frame (within its surrounding Image).		
	<code>inHorizDelta</code> and <code>inVertDelta</code> specify, in pixels, how far to move the frame (within its surrounding Image).		
Access	Virtual, Public		
Prototype	<pre>virtual void MoveBy(SInt32 inHorizDelta, SInt32 inVertDelta, Boolean inRefresh);</pre>		
Parameters	The parameters for this method are:		
	SInt32	<code>inHorizDelta</code>	Positive horizontal deltas move to the right, negative to the left.
	SInt32	<code>inVertDelta</code>	Positive vertical deltas move down, negative up.
	Boolean	<code>inRefresh</code>	Indicates whether to refresh or not.
Return	None		

PlaceInSuperFrameAt()

Purpose	Place the pane at a location relative to the frame of its superview.		
Access	Public		
Prototype	<pre>void PlaceInSuperFrameAt (SInt32 inHorizOffset, SInt32 inVertOffset, Boolean inRefresh);</pre>		
Parameters	The parameters for this method are:		

LPane

	SInt32	inHorizOffset	Specifies, in pixels, how far the left edge of the frame is from the left edge of its parent frame. Positive offsets are to the left, negative to the right.
	SInt32	inVertOffset	Specifies, in pixels, how far the top edge of the frame is from the top edge of its parent frame. Positive offsets are below, negative above.
	Boolean	inRefresh	This value is passed to MoveBy() .
Return	None		

PlaceInSuperImageAt()

Purpose	Place the pane within the Image of its superview.		
Access	Virtual, Public		
Prototype	<pre>virtual void PlaceInSuperImageAt(SInt32 inHorizOffset, SInt32 inVertOffset, Boolean inRefresh);</pre>		
Parameters	The parameters for this method are:		
	SInt32	inHorizOffset	Specify the distance of the top left of the frame from the top left of the Image of its superview.

	SInt32	inVertOffset	Specify the distance of the top left of the frame from the top left of the Image of its superview.
	Boolean	inRefresh	This value is passed to MoveBy() .
Return	None		

PointIsInFrame()

Purpose	This method determines whether a given coordinate is within the perimeter of the button.		
Access	Virtual, Public		
Prototype	virtual Boolean PointIsInFrame(SInt32 inHoriz, SInt32 inVert) const;		
Parameters	This method has the following parameters:		
	SInt32	inHoriz	The horizontal coordinate
	SInt32	inVert	The vertical coordinate
Return	Boolean of true if the point is within the perimeter, false otherwise.		

PointsAreClose()

Purpose	Return whether the two points are close enough to be part of a multi-click. Points are in Local coordinates.		
Access	Virtual, Public		
Prototype	virtual Boolean PointsAreClose(Point inFirstPt, Point inSecondPt) const;		
Parameters	This method has the following parameters:		

	Point	inFirstPt	The first point.
	Point	inSecondPt	The second point.

Return	Returns <code>true</code> if the points are close enough to be a multi-click, else <code>false</code> .		
--------	---	--	--

PortToGlobalPoint()

Purpose	Convert a point to global coordinates.		
Access	Virtual, Public		
Prototype	<code>virtual void PortToGlobalPoint(Point &ioPoint) const;</code>		
Parameters	The parameter for this method is:		
	Point&	ioPoint	The point to convert.

Return	None		
--------	------	--	--

PortToLocalPoint()

Purpose	Convert point from Port to Local coordinates.		
Access	Virtual, Public		
Prototype	<code>virtual void PortToLocalPoint(Point &ioPoint) const;</code>		
Parameters	The parameter for this method is:		
	Point&	ioPoint	The point to convert.

Return	None		
--------	------	--	--

PrintPanel()

Purpose Try to print a panel of a pane. Since a pane does not scroll, it just prints itself for every panel.

This is a wrapper function which calls [PrintPanelSelf\(\)](#) if it is proper for the pane to print. This means that:

- pane is visible
- pane's frame is in QuickDraw space
- pane's frame intersects inSuperDrawRgnH
- pane can be focused

Access Virtual, Public

Prototype `virtual void PrintPanel(
const PanelSpec &inPanel,
RgnHandle inSuperPrintRgnH);`

Parameters The parameters for this method are:

<code>const PanelSpec&</code>	<code>inPanel</code>	The panel info.
<code>RgnHandle</code>	<code>inSuperPrintRgnH</code>	Specifies, in Port coordinates, the portion of the pane's superview that needs to be printed. Specify <code>nil</code> to bypass the intersection test.

Return None

PrintPanelSelf()

Purpose Print a panel of a pane.

Access Virtual, Public

LPane

Prototype	<code>virtual void PrintPanelSelf(const PanelSpec& inPanel);</code>		
Parameters	The parameter for this method is:		
	<code>const</code>	<code>inPanel</code>	The panel info.
	<code>PanelSpec&</code>		
Return	None		

PutInside()

Purpose	Put pane inside the specified View. Location is unspecified. InitPane() calls PutInside() with <code>inOrient</code> set to <code>false</code> since the pane is not fully constructed at that point, meaning that any functions called by OrientSubPane() will be the ones for <code>LPane</code> , and not any overrides. This may result in the pane not being properly oriented. After creating a pane, you should call FinishCreate() , which will call OrientSubPane() for the fully constructed pane object.		
Access	Public		
Prototype	<code>void PutInside(LView *inView, Boolean inOrient);</code>		
Parameters	The parameters for this method are:		
	<code>Boolean</code>	<code>inOrient</code>	Specifies whether or not to orient the pane within its new superview. The default value is <code>true</code> , which is what you'll normally want.
	<code>LView*</code>	<code>inView</code>	The pointer to the View.
Return	None		

Refresh()

Purpose	Invalidate the area occupied by a pane. This forces an update event that, when processed, will redraw the pane. This method does nothing if the pane is not exposed.
Access	Public
Prototype	<code>virtual void Refresh();</code>
Parameters	None
Return	None

ResizeFrameBy()

Purpose	Change the frame size by the specified amounts. <code>inWidthDelta</code> and <code>inHeightDelta</code> specify, in pixels, how much larger to make the frame. Positive deltas increase the size, negative deltas reduce the size.		
Access	Virtual, Public		
Prototype	<code>virtual void ResizeFrameBy(SInt16 inWidthDelta, SInt16 inHeightDelta, Boolean inRefresh);</code>		
Parameters	The parameters for this method are:		
	SInt32	<code>inWidthDelta</code>	The width change.
	SInt32	<code>inHeightDelta</code>	The height change.
	Boolean	<code>inRefresh</code>	Indicates whether to refresh or not.
Return	None		

Remarks Note that this function only changes the frame size, not the bounds of any enclosing view. To change the actual size (on screen) of an [LWindow](#) object, call [ResizeWindowBy\(\)](#) or [ResizeWindowTo\(\)](#).

ResizeFrameTo()

Purpose Set the dimensions of the frame.

inWidthDelta and *inHeightDelta* specify, in pixels, how much larger to make the frame. Positive deltas increase the size, negative deltas reduce the size.

Access Public

Prototype `void ResizeFrameTo(
 SInt16 inWidth,
 SInt16 inHeight,
 Boolean inRefresh);`

Parameters The parameters for this method are:

SInt16	<i>inWidth</i>	The horizontal width change to resize the window by.
SInt16	<i>inHeight</i>	The vertical height change to resize the window by.
Boolean	<i>inRefresh</i>	Indicates whether to refresh or not.

Return None

RestorePlace()

Purpose Read size and location information stored in a Stream by the [SavePlace\(\)](#) function

Access Virtual, Public

Prototype	<code>virtual void RestorePlace(LStream *inPlace);</code>			
Parameters	The parameter for this method is: <table><tr><td><code>LStream*</code></td><td><code>inPlace</code></td><td>The stream to read from.</td></tr></table>	<code>LStream*</code>	<code>inPlace</code>	The stream to read from.
<code>LStream*</code>	<code>inPlace</code>	The stream to read from.		
Return	None			

SavePlace()

Purpose	Write size and location information to a Stream for later retrieval by the RestorePlace() function.			
Access	Virtual, Public			
Prototype	<code>virtual void SavePlace(LStream *outPlace);</code>			
Parameters	<div>The parameter for this method is:</div> <table><tr><td><code>LStream*</code></td><td><code>outPlace</code></td><td>The stream to write to.</td></tr></table>	<code>LStream*</code>	<code>outPlace</code>	The stream to write to.
<code>LStream*</code>	<code>outPlace</code>	The stream to write to.		
Return	None			

ScrollToPanel()

Purpose	Scroll pane to the specified panel. Panes do not scroll, so just return true indicating that the panel is always valid		
Access	Virtual, Public		
Prototype	<pre>virtual Boolean ScrollToPanel(const PanelSpec &inPanel);</pre>		
Parameters	The parameter for this method is:		
	<pre>const PanelSpec &</pre>	<pre>inPanel</pre>	The panel info.

Return	Return whether the specified panel exists. If it doesn't, View is not scrolled.
--------	---

SetDefaultView()

Purpose	Return the value of the sDefaultView data member.
Access	Static, Public
Prototype	<code>static void SetDefaultView(LView *inView);</code>
Parameters	None
Return	Return the value of the sDefaultView .

SetDescriptor()

Purpose	Set the descriptor string.			
Access	Virtual, Public			
Prototype	<code>virtual void SetDescriptor(ConstStringPtr inDescriptor);</code>			
Parameters	The parameter for this method is:			
	<table><tr><td><code>ConstStringPtr</code></td><td><code>inDescriptor</code></td><td>The string pointer to set for the descriptor.</td></tr></table>	<code>ConstStringPtr</code>	<code>inDescriptor</code>	The string pointer to set for the descriptor.
<code>ConstStringPtr</code>	<code>inDescriptor</code>	The string pointer to set for the descriptor.		
Return	None			

SetForeAndBackColors()

Purpose	Specify the foreground and/or background colors of a Window. Specify nil for inForeColor and/or inBackColor to leave that color trait unchanged
---------	---

Access	Virtual, Public		
Prototype	<pre>virtual void SetForeAndBackColors(const RGBColor *inForeColor, const RGBColor *inBackColor);</pre>		
Parameters	The parameters for this method are:		
	const RGBColor*	inForeColor	The foreground color.
	const RGBColor*	inBackColor	The background color.
Return	None		

SetFrameBinding()

Purpose	This method sets the value of the mFrameBinding data member.		
Access	Public		
Prototype	<pre>void SetFrameBinding(const SBooleanRect &inFrameBinding);</pre>		
Parameters	The parameter for this method is:		
	const SBooleanRect &	inFrameBinding	The rect for the frame.
Return	None		

SetLastPaneMoused()

Purpose	This data member sets the value of the sLastPaneMoused data member.		
Access	Static, Public		

LPane

Prototype	<code>static void SetLastPaneMoused(LPane *inPane);</code>	
Parameters	The parameter for this method is:	
	<code>LPane*</code>	<code>inPane</code> The rect for the frame.
Return	None	

SetPaneID()

Purpose	Set the value of the mPaneID data member.	
Access	Public	
Prototype	<code>void SetPaneID(PaneIDT inPaneID);</code>	
Parameters	The parameter for this method is:	
	<code>PaneIDT</code>	<code>inPaneID</code> The resource ID for the pane.
Return	None	

SetRefreshAllWhenResized()

Purpose	This method sets the value of the mRefreshAllWhenResized data member.	
Access	Public	
Prototype	<code>void SetRefreshAllWhenResized(Boolean inRefreshAll);</code>	
Parameters	The parameter for this method is:	
	<code>Boolean</code>	<code>inRefreshAll</code> Whether to refresh all panes when resized.
Return	None	

SetUserCon()

Purpose	Set the mUserCon data member.	
Access	Virtual, Public	
Prototype	<code>virtual void SetUserCon(SInt32 inUserCon);</code>	
Parameters	The parameter for this method is:	
	SInt32	inUserCon
	The value to set for the data member.	
Return	None	

SetValue()

Purpose	Override this method to provide useful functionality for setting the value of a pane.	
Access	Virtual, Public	
Prototype	<code>virtual void SetValue(SInt32 inValue);</code>	
Parameters	The parameter for this method is:	
	SInt32	inValue
	The value to set.	
Return	None	

Show()

Purpose	Make a pane visible.
Access	Virtual, Public
Prototype	<code>virtual void Show();</code>

LPane

Parameters	None
Return	None

ShowSelf()

Purpose	Pane is being made visible. You should override this method for pane classes that need to be informed when they are made visible.
Access	Virtual, Protected
Prototype	<code>virtual void ShowSelf();</code>
Parameters	None
Return	None

SuperActivate()

Purpose	The superview of a pane has been activated.
Access	Virtual, Protected
Prototype	<code>virtual void SuperActivate();</code>
Parameters	None
Return	None

SuperDeactivate()

Purpose	The superview of a pane has been deactivated.
Access	Virtual, Protected
Prototype	<code>virtual void SuperDeactivate();</code>
Parameters	None

Return	None
--------	------

SuperDisable()

Purpose	The superview of a pane has been disabled.
Access	Virtual, Protected
Prototype	<code>virtual void SuperDisable();</code>
Parameters	None
Return	None

SuperEnable()

Purpose	The superview of a pane has been enabled.
Access	Virtual, Protected
Prototype	<code>virtual void SuperEnable();</code>
Parameters	None
Return	None

SuperHide()

Purpose	The superview of a pane has been hidden.
Access	Virtual, Protected
Prototype	<code>virtual void SuperHide();</code>
Parameters	None
Return	None

SuperPrintPanel()

Purpose	superview is printing a panel. The View is not in control of pagination. In general, it is not clear how to print nested scrolling views. This function just prints the View at its current location, without scrolling to a particular panel.		
Access	Virtual, Public		
Prototype	<pre>virtual void SuperPrintPanel(const PanelSpec &inSuperPanel, RgnHandle inSuperPrintRgnH);</pre>		
Parameters	The parameters for this method are:		
	const PanelSpec&	inSuperPanel	The panel info.
	RgnHandle	inSuperPrintRgnH	Specifies, in Port coordinates, the portion of the pane's superview that needs to be printed. Specify <i>nil</i> to bypass the intersection test.
Return	None		

SuperShow()

Purpose	The superview of a pane has become visible.
Access	Virtual, Protected
Prototype	<pre>virtual void SuperShow();</pre>
Parameters	None
Return	None

UpdateClickCount()

Purpose	Determine if the mouse down is part of a multi-click and set internal counters.						
Access	Virtual, Public						
Prototype	<pre>virtual void UpdateClickCount(const SMouseDownEvent &inMouseDown);</pre>						
Parameters	<p>The parameter for this method is:</p> <table><tr><td><pre>const</pre></td><td><pre>inMouseDown</pre></td><td>The event info.</td></tr><tr><td><pre>EventRecord&</pre></td><td></td><td></td></tr></table>	<pre>const</pre>	<pre>inMouseDown</pre>	The event info.	<pre>EventRecord&</pre>		
<pre>const</pre>	<pre>inMouseDown</pre>	The event info.					
<pre>EventRecord&</pre>							
Return	None						

UpdatePort()

Purpose	<p>Redraw invalidated area of the Port containing the pane.</p> <p>For Panes that are in a Window Port (i.e., the ultimate super view is a Window), this forces an immediate redraw of the update region of the Window. Since this message is really directed at the Port containing the pane (rather than the pane itself), the update occurs even if the pane is not visible.</p> <p>Panes that maintain a Mac GrafPort must override this function if they support updating.</p>
Access	Virtual, Public
Prototype	<pre>virtual void UpdatePort();</pre>
Parameters	None
Return	None

ValidPortRect()

Purpose	Sets the port rect.		
Access	Virtual, Public		
Prototype	<code>virtual void ValidPortRect(const Rect *inRect);</code>		
Parameters	The parameter for this method is:		
	<code>const Rect*</code>	<code>inRect</code>	The Rect.
Return	None		

ValidPortRgn()

Purpose	Sets the port region.		
Access	Virtual, Public		
Prototype	<code>virtual void ValidPortRgn(RgnHandle inRgnH);</code>		
Parameters	The parameter for this method is:		
	<code>RgnHandle</code>	<code>inRgnH</code>	The region handle.
Return	None		

sDefaultView

Purpose	The default view pointer storage.
Access	Static, Protected
Prototype	<code>static LView *sDefaultView;</code>

sLastPaneClicked

Purpose	The storage for the last pane clicked.
Access	Static, Protected
Prototype	<code>static LPane *sLastPaneClicked;</code>

sLastPaneMoused

Purpose	The storage for the last pane moused.
Access	Static, Protected
Prototype	<code>static LPane *sLastPaneMoused;</code>

sWhenLastMouseUp

Purpose	Storage for when the last mouse up occurred.
Access	Static, Protected
Prototype	<code>static UInt32 sWhenLastMouseUp</code>

sWhenLastMouseDown

Purpose	Storage for when the last mouse down occurred.
Access	Static, Protected
Prototype	<code>static UInt32 sWhenLastMouseDown;</code>

sWhereLastMouseDown

Purpose	Storage for where the last mouse down occurred.
Access	Static, Protected
Prototype	<code>static Point sWhereLastMouseDown;</code>

sClickCount

Purpose	Storage for the click count.
Access	Static, Protected
Prototype	<code>static SInt16 sClickCount;</code>

mPaneID

Purpose	The storage for the pane ID.
Access	Protected
Prototype	<code>PaneIDT mPaneID;</code>

mFrameSize

Purpose	The storage for the frame size.
Access	Protected
Prototype	<code>SDimension16 mFrameSize;</code>

mFrameLocation

Purpose	Storage for the frame location.
Access	Protected
Prototype	<code>SPoint32 mFrameLocation;</code>

mFrameBinding

Purpose	Storage for the frame binding.
Access	Protected
Prototype	<code>SBooleanRect mFrameBinding;</code>

mUserCon

Purpose	Storage for the UserCon storage area.
Access	Protected
Prototype	<code>SInt32 mUserCon;</code>

mSuperView

Purpose	Storage for the superview.
Access	Protected
Prototype	<code>LView *mSuperView;</code>

LPane

mVisible

Purpose	Storage for whether the pane is visible or not.
Access	Protected
Prototype	<code>ETriState mVisible;</code>

mActive

Purpose	This is storage for whether the pane is active or not.
Access	Protected
Prototype	<code>ETriState mActive;</code>

mEnabled

Purpose	This is storage for whether the pane is enabled or not.
Access	Protected
Prototype	<code>ETriState mEnabled;</code>

mRefreshAllWhenResized

Purpose	Storage for whether to resize all when the pane is resized.
Access	Protected
Prototype	<code>Boolean mRefreshAllWhenResized;</code>

LPeriodical

Overview LPeriodical is an abstract PowerPlant base class that is multiply inherited by other framework objects. It is used for giving objects the ability to do work at periodic intervals. In PowerPlant, an object that receives processor time on a periodic basis is said to be a Periodical.

Since LPeriodical is an abstract base class, you can't instantiate an LPeriodical object. Instead, you inherit from LPeriodical in your object's class definition, and define any overrides of the base class methods needed by your object.

This class maintains two static queues, an Idler queue and a Repeater queue. It is up to the caller to determine the meaning of each queue and when to devote time to the Periodicals in them. The PowerPlant [LApplication](#) class devotes time to Idlers at Null Event time, and devotes time to Repeaters after every event.

Methods The methods in this class are:

LPeriodical()	~LPeriodical()
DeleteIdlerAndRepeaterQueues()	DevoteTimeToIdlers()
DevoteTimeToRepeaters()	SpendTime()
StartIdling()	StartRepeating()
StopIdling()	StopRepeating()

Data Members The data members in this class are:

sIdlerQ	sRepeaterQ
-------------------------	----------------------------

Operation Many different objects in your program may require processor time on a periodic basis. For example, changing the text edit insertion caret in [LEditField](#) requires that some periodic operations are performed. Another example is [LGrowZone\(\)](#), which requires periodic polling of memory conditions in order to be effective at handling low-memory situations.

There are two different kinds of periodical objects that you create:

- Repeaters

LPeriodical

- Idlers

An Idler object is an object that gets processing time when the application gets a null event. In other words, when the application thread has nothing else to do, it will process the Idler queue.

A Repeater object is an object that gets processing time after every event.

Source files (Feature Classes)

LPeriodical.h

LPeriodical.cp

See also [LArray](#)

[TArray](#)

LPeriodical()

Purpose	The constructor for LPeriodical doesn't do anything.
Access	Public
Prototype	<code>LPeriodical();</code>
Parameters	None
Return	None

~LPeriodical()

Purpose	The destructor for LPeriodical stops the Idler and Repeater queue processing.
Access	Public
Prototype	<code>virtual ~LPeriodical();</code>
Parameters	None

Return None

DeleteIdlerAndRepeaterQueues()

Purpose	This method deletes the internal queues used to store pointers to Idler and Repeater objects. This method does not delete the Idler and Repeater objects themselves.
	You don't need to call this when quitting an application, since the queues will disappear when the System deallocates the application heap. However, you may want to call this if you like deleting every object that is created via the new operator.
Access	Public, Static
Prototype	<code>static void DeleteIdlerAndRepeaterQueues();</code>
Parameters	None
Return	None
Remarks	Normally, you will only use this routine to clean up memory when terminating a code resource or fragment. You do not normally need to call this method in from application code.

DevoteTimeToIdlers()

Purpose	This method calls SpendTime() for each Periodical object in the Idler queue. This gives each object some procesing time.		
Access	Public, Static		
Prototype	<code>static void DevoteTimeToIdlers(const EventRecord &inMacEvent);</code>		
Parameters	This method has the following parameter:		
	<code>const EventRecord&</code>	<code>inMacEvent</code>	A struct containing the parameters from the last event that occurred.

Return None

DevoteTimeToRepeaters()

Purpose	This method calls SpendTime() for each Periodical object in the Repeater queue. This gives each object some procesing time.	
Access	Public, Static	
Prototype	<pre>static void DevoteTimeToRepeaters(const EventRecord &inMacEvent);</pre>	
Parameters	This method has the following parameter:	
	<pre>const inMacEvent EventRecord&</pre>	A struct containing the parameters from the last event that occurred.
Return	None	

SpendTime()

Purpose	This method is a pure virtual method. You must override it in your concrete classes. You provide code that implements the operations that are required for your object to perform when it is given processing time.	
Access	Virtual, Public	
Prototype	<pre>virtual void SpendTime(const EventRecord &inMacEvent) = 0;</pre>	
Parameters	This method has the following parameter:	
	<pre>const inMacEvent EventRecord&</pre>	A struct containing the parameters from the last event that occurred.
Return	None	

StartIdling()

Purpose	This method creates an Idler queue if one doesn't already exist. Then, it adds this Periodical object to the end of the queue if it isn't already in there.
Access	Public, Virtual
Prototype	<code>virtual void StartIdling();</code>
Parameters	None
Return	None

StartRepeating()

Purpose	This method creates a Repeater queue if one doesn't already exist. Then, it adds this Periodical object to the end of the queue if it isn't already in there.
Access	Public, Virtual
Prototype	<code>virtual void StartRepeating();</code>
Parameters	None
Return	None

StopIdling()

Purpose	Remove this Periodical object from the Idler queue.
Access	Public, Virtual
Prototype	<code>virtual void StopIdling();</code>
Parameters	None
Return	None

StopRepeating()

Purpose	Remove this Periodical object from the Repeater queue.
Access	Public, Virtual
Prototype	<code>virtual void StopRepeating();</code>
Parameters	None
Return	None

sIdlerQ

Purpose	This data member is a pointer to a TArray of pointers to LPeriodical objects. In short, this data member is a pointer to a list of objects in the Idler queue.
Access	Static, Protected
Prototype	<code>static TArray<LPeriodical*> *sIdlerQ;</code>
Remarks	In general, you will want to use the methods in the LPeriodical class to manipulate the sIdlerQ list, rather than accessing it directly.

sRepeaterQ

Purpose	This data member is a pointer to a TArray of pointers to LPeriodical objects. In short, this data member is a pointer to a list of objects in the Repeater queue.
Access	Static, Protected
Prototype	<code>static TArray<LPeriodical*> *sRepeaterQ;</code>
Remarks	In general, you will want to use the methods in the LPeriodical class to manipulate the sRepeaterQ list, rather than accessing it directly.

LPicture

Overview	LPicture is a PowerPlant class that is used for displaying PICT resources.
Methods	<p>The methods in this class are:</p> <p>LPicture()DrawSelf() GetPictureID()InitPicture() SetPictureID()</p>
Data Members	<p>The data members in this class are:</p> <p>mPICTid</p>
Operation	This is a simple extension of the LView class.
Source files	<p>Pane Classes)</p> <p><code>LPicture.h</code> <code>LPicture.cp</code></p>
Ancestors	<p>LAttachable LPane LView</p>

LPicture()

Purpose	The constructors create objects from the passed-in parameters.
Access	Public
Prototype	<pre>LPicture(); LPicture(const LPicture &inOriginal); LPicture(const SPaneInfo &inPaneInfo, const SViewInfo &inViewInfo, ResIDTinPICTid); LPicture(LStream *inStream);</pre>

LPicture

```
LPicture( ResIDT inPictureID );
```

Parameters The parameters for these constructors are:

const LPicture&	inOriginal	A reference to the picture to copy.
const SPaneInfo&	inPaneInfo	The Superpane info.
const SViewInfo&	inViewInfo	The Superview info.
ResIDT	inPICTid	The resource ID for the PICT resource.
LStream*	inStream	The stream to read from.
ResIDT	inPictureID	The resource ID for the PICT resource.

DrawSelf()

Purpose Draw the picture. This is an override of [DrawSelf\(\)](#) in [LPane](#).

GetPictureID()

Purpose Retrieve the picture ID.

Access Public

Prototype ResIDT GetPictureID() const;

Parameters None

Return A resource ID for the picture.

InitPicture()

Purpose	Private Initializer. Assumes mPICTid is set.
Access	Private
Prototype	<code>void InitPicture();</code>
Parameters	None
Return	None

SetPictureID()

Purpose	Set the PICT Resource ID associated with a picture. This method changes the size of the image to match the bounds of the PICT.		
Access	Public		
Prototype	<code>void SetPictureID(ResIDT inPictureID);</code>		
Parameters	This method has the following parameter:		
	ResIDT	inPictureID	The resource ID for the PICT resource.
Return	None		

mPICTid

Purpose	The picture resource ID.
Access	Protected
Prototype	<code>ResIDT mPICTid;</code>

LPlaceholder

Overview	LPlaceholder is a PowerPlant class that is designed to assist the printing process. The purpose of a placeholder is to allow you to print a pane at a size and/or location that is different from the pane's characteristics when displayed in a window.	
Methods	The methods in this class are: LPlaceholder() CountPanels() RemoveOccupant() ~LPlaceholder() InstallOccupant() ScrollToPanel()	
Data Members	The data members in this class are: mOccupant mOccupantPlaceH mOccupantSuperView mOccupantAlignment	
Operation	A placeholder has two characteristics that differentiate it from other views. It has an occupant, and the occupant has an alignment. For more details on this class, refer to <i>The PowerPlant Book</i> .	
Source files	(Pane Classes) LPlaceholder.h LPlaceholder.cp	
Ancestors	LAttachable LPane LView	

LPlaceholder()

Purpose	The constructor creates objects from the passed-in parameters.
Access	Public

LPlaceholder

Prototype	<pre>LPlaceholder(); LPlaceholder(const LPlaceholder &inOriginal); LPlaceholder(const SPaneInfo &inPaneInfo, const SViewInfo &inViewInfo, UInt16 inOccupantAlignment); LPlaceholder(LStream *inStream);</pre>		
Parameters	The constructors take the following parameters:		
	const LPlaceholder&	inOriginal	The object to copy.
	const SPaneInfo&	inPaneInfo	The Superpane.
	const SViewInfo&	inViewInfo	The Superview.
	UInt16	inOccupant -Alignment	The alignment.
	LStream*	inStream	The stream to read from.

~LPlaceholder()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<pre>virtual ~LPlaceholder();</pre>

CountPanels()

Purpose	Return the number of horizontal and vertical Panels. A Panel is a "frameful" of a View's Image. For an LPlaceholder, the number of Panels is the number of Panels in its occupant Pane.
Access	Virtual, Public

Prototype `virtual void CountPanels(UInt32 &outHorizPanels, UInt32 &outVertPanels);`

Parameters This method has the following parameters:

UInt32&	outHorizPanels	The number of horizontal panels.
UInt32&	outVertPanels	The number of vertical panels.

Return None

InstallOccupant()

Purpose Install a pane inside an LPlaceholder.

Use the Mac OS Toolbox `IconAlignmentType` values (in `<Icons.h>`) for `inAlignment`. If you don't specify a horizontal alignment, the Occupant width is set to the Placeholder width. Similarly, if you don't specify a vertical alignment, the Occupant height is set to the Placeholder height.

Access Public

Prototype `void InstallOccupant(LPane *inOccupant, UInt16 inAlignment);`

Parameters This method has the following parameter:

LPane *	inOccupant	The occupant.
UInt16	inAlignmen t	Mac OS Toolbox <code>IconAlignmentType</code> values (in <code><Icons.h></code>). This is an optional parameter, if not specified, the default value (of -1) means to use the alignment stored in the Placeholder.

Return None

RemoveOccupant()

Purpose	Remove occupant from a Placeholder, restoring the occupant to its original state.
Access	Public
Prototype	<code>LPane* RemoveOccupant () ;</code>
Parameters	None
Return	The pointer to the occupant.

ScrollToPanel()

Purpose	Scroll view image to the specified panel. For a Placeholder, scroll occupant pane to the specified panel.		
Access	Virtual, Public		
Prototype	<code>virtual Boolean ScrollToPanel(const PanelSpec &inPanel) ;</code>		
Parameters	This method takes the following parameters:		
	<code>const</code>	<code>inPanel</code>	The panel to scroll.
	<code>PanelSpec&</code>		
Return	None		

mOccupant

Purpose	The occupant pointer.
Access	Protected
Prototype	<code>LPane *mOccupant ;</code>

mOccupantSuperView

Purpose	The occupant superview pointer.
Access	Protected
Prototype	LView *mOccupantSuperView;

mOccupantPlaceH

Purpose	The occupant handle.
Access	Protected
Prototype	Handle mOccupantPlaceH;

mOccupantAlignment

Purpose	Occupant alignment parameters.
Access	Protected
Prototype	UInt16 mOccupantAlignment;

LPreferencesFile

Overview	LPreferencesFile is a PowerPlant class that is used for creating preferences files for your application.
Methods	The methods in this class are: LPreferencesFile() OpenOrCreateResourceFork()
Data Members	There are no data members for this class.
Operation	This is a very simple class.
Source files	(File & StreamClasses) LPreferencesFile.h LPreferencesFile.cp
Ancestors	LFile

LPreferencesFile()

Purpose	The constructors create objects from the passed-in values.	
Access	Public	
Prototype	<pre>LPreferencesFile(); LPreferencesFile(ConstStr255Param inFileName, Boolean inCreateFolder); LPreferencesFile(FSSpec& inFileSpec); LPreferencesFile(AliasHandle inAlias, Boolean& outWasChanged, FSSpec* inFromFile);</pre>	
Parameters	The parameters for these constructors are:	
ConstStr255Param	inFileName	The name of the file to create.
Boolean	inCreateFolder	This is passed to FindFolder().
FSSpec&	inFileSpec	The file spec for the file to use.

LPReferencesFile

AliasHandle	inAlias	A handle to an alias to use.
Boolean	outWasChanged	Indicates if the AliasHandle was changed during resolution.
FSSpec*	inFromFile	A File Specifier for the starting point for a relative search. If nil, an absolute search is performed.

OpenOrCreateResourceFork()

Purpose	.Opens the resource fork of the preferences file, if it exists. If the resource fork doesn't exist, it creates one. If the file doesn't exist, it creates the file with the designated type and creator codes.		
Access	Virtual, Public		
Prototype	<pre>virtual Int16 OpenOrCreateResourceFork(Int16 inPrivileges, OSType inCreator, OSType inFileType, ScriptCode inScriptCode);</pre>		
Parameters	The parameters for these constructors are:		
	Int16	inPrivileges	The privileges to use.
	OSType	inCreator	The creator type to use.
	OSType	inFileType	The file type to use.
	ScriptCode	inScriptCode	The script code to use.
Return	Returns the RefNum of the resource fork for the file created.		

LPrintout

Overview	LPrintout is a PowerPlant class that is used for support of printing in your application.	
Methods	The methods in this class are:	
	LPrintout()	~LPrintout()
	ApplyForeAndBackColors()	CountPanels()
	CreatePrintout()	DoPrintJob()
	EstablishPort()	FinishCreateSelf()
	GetMacPort()	GetPrintJobSpecs()
	GetPrintRecord()	HasAttribute()
	InitPrintout()	PageToPanel()
	PrintCopiesOfPages()	PrintPanel()
	PrintPanelRange()	SetAttribute()
	SetForeAndBackColors()	SetPrintRecord()
Data Members	The data members in this class are:	
	mAttributes	mPrintRecordH
	mPrinterPort	mWindowPort
	mHorizPanelCount	mVertPanelCount
	mForeColor	mBackColor
Operation	To learn more about working with LPrintout, refer to <i>The PowerPlant Book</i> .	
Source files	(Pane Classes)	
	LPrintout.h	
	LPrintout.cp	
Ancestors	LAttachable	
	LPane	

[LView](#)

LPrintout()

Purpose	The constructors create objects from the passed-in parameters.	
Access	Public	
Prototype	<pre>LPrintout(); LPrintout(THPrint inPrintRecordH); LPrintout(LStream *inStream);</pre>	
Parameters	The parameters for these constructors are:	
	THPrint inPrintRecordH	The print record handle.
	LStream* inStream	The stream to read from.

~LPrintout()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<pre>virtual ~LPrintout();</pre>

ApplyForeAndBackColors()

Purpose	Set the foreground and background colors of the current port. The Printout or one of its subpanes must already be focused. This is an override of ApplyForeAndBackColors() in LPane .
---------	---

CountPanels()

Purpose Count the number of Panels in a Printout. This is an override of [CountPanels\(\)](#) in [LPane](#).

CreatePrintout()

Purpose Return a new Printout object (and its SubPanels) from the data in a 'PPob' resource.

Access Static, Public

Prototype `static LPrintout* CreatePrintout(ResIDT inPrintoutID);`

Parameters The parameter for this method is:
`ResIDT inPrintoutID` The resource ID for the printout.

Return The pointer to the new object.

DoPrintJob()

Purpose Print the job.

Access Virtual, Public

Prototype `virtual void DoPrintJob();`

Parameters None

Return None

EstablishPort()

Purpose	Make Printout the current port.
Access	Virtual, Public
Prototype	<code>virtual Boolean EstablishPort();</code>
Parameters	None
Return	Boolean indicating whether the port was successfully set, false if not.

FinishCreateSelf()

Purpose	This is an override of FinishCreateSelf() in LPane .
---------	--

GetMacPort()

Purpose	Return the GrafPort associated with a Printout. This is an override of GetMacPort() in LPane .
---------	--

GetPrintJobSpecs()

Purpose	Extract information about the print job from the Mac OS Toolbox PrintRecord. <ul style="list-style-type: none">• First Page to print (mapped to a PanelSpec)• Last Page to print (mapped to a PanelSpec)• Number of copies to print
Access	Virtual, Protected

Prototype	<pre>virtual void GetPrintJobSpecs(PanelSpec &outFirstPanel, PanelSpec &outLastPanel, UInt16 &outCopyCount);</pre>		
Parameters	The parameters for this method are:		
	PanelSpec&	outFirstPanel	The first panel.
	PanelSpec&	outLastPanel	The last panel.
	UInt16&	outCopyCount	The number of copies.
Return	None		

GetPrintRecord()

Purpose	Retrieve the PrintRecord.
Access	Public
Prototype	<pre>THPrint GetPrintRecord() const;</pre>
Parameters	None
Return	A handle to the PrintRecord.
Remarks	None

HasAttribute()

Purpose	Return whether a Printout has the specified attribute.		
Access	Public		
Prototype	<pre>Boolean HasAttribute(EPrintAttr inAttribute) const;</pre>		
Parameters	The parameter for this method is:		
	EPrintAttr	inAttribute	The attribute to check for.

LPrintout

Return	Boolean indicating whether the attribute is present (<code>true</code>) or not (<code>false</code>).
--------	--

InitPrintout()

Purpose	Initialize data members to default values.
Access	Private
Prototype	<code>void InitPrintout();</code>
Parameters	None
Return	None

PageToPanel()

Purpose	Fill in <code>PanelSpec</code> for a given page number.						
Access	Virtual, Protected						
Prototype	<code>virtual void PageToPanel(UInt32 inPageNumber, PanelSpec &outPanel) const;</code>						
Parameters	The parameters for this method are: <table><tr><td><code>UInt32</code></td><td><code>inPageNumber</code></td><td>The page number to check on.</td></tr><tr><td><code>PanelSpec&</code></td><td><code>outPanel</code></td><td>The filled-in <code>panelspec</code>.</td></tr></table>	<code>UInt32</code>	<code>inPageNumber</code>	The page number to check on.	<code>PanelSpec&</code>	<code>outPanel</code>	The filled-in <code>panelspec</code> .
<code>UInt32</code>	<code>inPageNumber</code>	The page number to check on.					
<code>PanelSpec&</code>	<code>outPanel</code>	The filled-in <code>panelspec</code> .					
Return	None						

PrintCopiesOfPages()

Purpose	Print copies of the specified range of Panels. This is the main "print loop."
---------	---

Access	Virtual, Protected		
Prototype	<pre>virtual void PrintCopiesOfPages(const PanelSpec &inFirstPanel, const PanelSpec &inLastPanel, UInt16 inCopyCount);</pre>		
Parameters	The parameters for this method are:		
	UInt16	inCopyCount	The page number to check on.
	PanelSpec&	inFirstPanel	The first panel.
	PanelSpec&	inLastPanel	The last panel.

PrintPanel()

Purpose	Print the specified Panel		
Access	Virtual, Public		
Prototype	<pre>virtual void PrintPanel(const PanelSpec &inPanel, RgnHandle inSuperPrintRgnH);</pre>		
Parameters	The parameters for this method are:		
	const PanelSpec&	inPanel	The panel spec.
	RgnHandle	inSuperPrintRgnH	The region handle.

PrintPanelRange()

Purpose	Print the range of panels.		
Access	Virtual, Public		
Prototype	<pre>virtual void PrintPanelRange(const PanelSpec &inFirstPanel,</pre>		

LPrintout

```
const PanelSpec &inLastPanel,  
UInt16 inCopyCount );
```

Parameters	The parameters for this method are:		
	const PanelSpec&	inFirstPanel	The first panel.
	const PanelSpec&	inLastPanel	The last panel.
	RgnHandle	inSuperPrintRgnH	The region handle.
Return	None		

SetAttribute()

Purpose	Specify attributes for a Printout.		
Access	Public		
Prototype	void SetAttribute(EPrintAttr inAttribute);		
Parameters	The parameter for this method is:		
	EPrintAttr	inAttribute	The region handle.
Return	None		

SetForeAndBackColors()

Purpose	Specify the foreground and/or background colors of a Printout. Specify nil for inForeColor and/or inBackColor to leave that color trait unchanged. This is an override of SetForeAndBackColors() in LPane .
---------	---

SetPrintRecord()

Purpose	Set the Toolbox PrintRecord for a Printout.		
Access	Public		
Prototype	<code>void SetPrintRecord(THPrint inPrintRecordH);</code>		
Parameters	The parameter for this method is:		
	THPrint	inPrintRecordH	The handle to print record.
Return	None		

mAttributes

Purpose	The attributes.
Access	Protected
Prototype	<code>UInt32 mAttributes;</code>

mPrintRecordH

Purpose	The handle PrintRecord.
Access	Protected
Prototype	<code>THPrint mPrintRecordH;</code>

mPrinterPort

Purpose	The printer port.
Access	Protected

LPrintout

Prototype `TPPrPort mPrinterPort;`

mWindowPort

Purpose The window pointer.

Access Protected

Prototype `WindowPtr mWindowPort;`

mHorizPanelCount

Purpose

Access Protected

Prototype `UInt32 mHorizPanelCount;`

mVertPanelCount

Purpose The vertical panel count.

Access Protected

Prototype `UInt32 mVertPanelCount;`

mForeColor

Purpose The foreground color.

Access Protected

Prototype `RGBColor mForeColor;`

mBackColor

Purpose	The background color.
Access	Protected
Prototype	<code>RGBColor mBackColor;</code>

LQueue

Overview	LQueue is a PowerPlant class that is used for implementing full linked list behavior for a list of LLink objects.	
Methods	The methods in this class are:	
	LQueue()	~LQueue()
	DoForEach()	GetSize()
	IsEmpty()	NextGet()
	NextPut()	Remove()
	operator =()	
Data Members	The data members in this class are:	
	mFirst	mLast
	mSize	
Operation	Every element in LQueue is an LLink object. The methods also work on LLink objects. LQueue implements a first-in, first-out (FIFO) queue. You add elements to the end of the list, and you may only get items from the head of the list.	
Source files	(Threads Classes)	
	LQueue.h	
	LQueue.cp	
See also	LLink	

LQueue()

Purpose	The constructor creates the object.
Access	Public
Prototype	<code>LQueue () ;</code>

LQueue

Parameters None

~LQueue()

Purpose The destructor destroys the LQueue object.

Access Virtual

Prototype `virtual ~LQueue();`

DoForEach()

Purpose Execute a user-supplied function for each element in the queue.

Access Virtual, Public

Prototype `virtual void DoForEach(LQueueIterator proc,
 void* arg);`

Parameters The parameters for this method are:

LQueueIterator	proc	The queue iterator.
void*	arg	The function to perform.

Return None

GetSize()

Purpose Returns the number of elements in the queue.

Access Virtual, Public

Prototype `virtual UInt32 GetSize() const;`

Parameters None

Return The number of items in the queue.

IsEmpty()

Purpose Indicates if the queue not contain any elements.

Access Virtual, Public

Prototype `virtual Boolean IsEmpty() const;`

Parameters None

Return Return true if the queue is empty.

NextGet()

Purpose Removes and returns the first element in the queue.

Access Virtual, Public

Prototype `virtual LLink* NextGet();`

Parameters None

Return A [LLink](#) pointer to the element in the queue.

NextPut()

Purpose Adds the given element to the end of the queue.

Access Virtual, Public

Prototype `virtual void NextPut(LLink* inLinkP);`

Parameters A pointer to an [LLink](#)

Return None

Remove()

Purpose	Remove an arbitrary element from the queue. This function traverses the entire queue, looking for the given queue element. If the element is found, it is removed from the queue.
Access	Virtual, Public
Prototype	<code>virtual Boolean Remove(LLink *inLinkP);</code>
Parameters	A pointer to an LLink
Return	Returns a Boolean indicating if the element was found.

operator =()

Purpose	The assignment operator
Access	Private
Prototype	<code>LQueue& operator = (const LQueue&);</code>
Parameters	An LQueue object to assign
Return	Assigned object

mFirst

Purpose	The head of the queue.
Access	Protected
Prototype	<code>LLink* mFirst;</code>

mLast

Purpose	The tail of the queue.
Access	Protected
Prototype	<code>LLink* mLast;</code>

mSize

Purpose	The size of the queue in elements.
Access	Protected
Prototype	<code>UInt32 mSize;</code>

LRadioGroup

Overview	LRadioGroup is a PowerPlant class that is used for managing a group of Controls by ensuring that only one Control in a group is "on" at any time. This is the standard behavior of a set of Radio Buttons.	
Methods	The methods in this class are: LRadioGroup() AddRadio() ListenToMessage() ~LRadioGroup() GetCurrentRadioID()	
Data Members	The data members in this class are: mCurrentRadio	
Operation	Although you will normally use this class with StdRadioButton objects, the group members can be any kind of Control. Therefore, you can group your own custom Control objects. This class assumes the "off" value is zero, and the "on" value is one. RadioGroup and its Controls have a Listener/Broadcaster relationship.	
Source files	(Support Classes) LRadioGroup.h LRadioGroup.cp	
See also	LControl LListener	

LRadioGroup()

Purpose	The constructors create objects from the passed-in parameters.
Access	Public

LRadioGroup

Prototype `LRadioGroup() ;`
 `LRadioGroup(LStream *inStream) ;`

Parameters The stream constructor has the following parameter:

<code>LStream*</code>	<code>inStream</code>	A pointer to a stream object that contains the information to create the LRadioGroup object.
-----------------------	-----------------------	--

~LRadioGroup()

Purpose The destructor destroys the object.

Access Virtual, Public

Prototype `virtual ~LRadioGroup() ;`

AddRadio()

Purpose Adds a Control to a Radio Group.

If the Control is "on" it becomes the current radio, turning off the former current radio.

Access Virtual, Public

Prototype `virtual void AddRadio(LControl *inRadio) ;`

Parameters The parameter for this method is:

<code>LControl</code>	<code>inRadio</code>	The pointer to the LControl object that we wish to add to.
<code>*</code>		

Return None

GetCurrentRadioID()

Purpose	Return the PaneID of the current RadioButton in the Group.
Access	Virtual, Public
Prototype	<code>virtual PaneIDT GetCurrentRadioID();</code>
Parameters	None
Return	PaneIDT

ListenToMessage()

Purpose	React to Messages from Broadcasters, which must be the Controls in the group. This is a pure virtual method (ListenToMessage()) in the base class which you must override. Refer to the description there for more information.
---------	---

mCurrentRadio

Purpose	This data member holds the pointer to the current radio button.
Access	Protected
Prototype	<code>LControl *mCurrentRadio;</code>

LReentrantMemoryPool

Overview	LReentrantMemoryPool is a PowerPlant class that is areentrant memory manager for use by interrupt callbacks. Based on the NEWMODE_FAST implementation of operator new.	
Methods	The methods in this class are:	
	LReentrantMemoryPool()	~LReentrantMemoryPool()
	AddPool()	AllocFrom()
	DisposePtr()	FreeMem()
	GetPtrSize()	MakeFreeBlock()
	NewPtr()	NewPtrClear()
	TotalMem()	
Data Members	The data members in this class are:	
	mMemoryPools	mFreeBlocks;
Source files	(Networking Classes)	
	LReentrantMemoryPool.h	
	LReentrantMemoryPool.cp	
See also	LSharedMemoryPool	

LReentrantMemoryPool()

Purpose	The constructor creates the object, specifying the initial size of the memory pool. IMPORTANT: MUST NOT BE CALLED AT INTERRUPT TIME!
Access	Public
Prototype	<pre>LReentrantMemoryPool (UInt32 inInitialSize); LReentrantMemoryPool(); // do not use LReentrantMemoryPool (LReentrantMemoryPool&);</pre>

LReentrantMemoryPool

Parameters	The parameters for this constructor are:		
	UInt32	inInitialSize	The initial size of the pool.

~LReentrantMemoryPool()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LReentrantMemoryPool();</code>

AddPool()

Purpose	Add a memory pool.		
Access	Public		
Prototype	<code>void AddPool (UInt32 inPoolSize);</code>		
Parameters	The parameters for this method are:		
	UInt32	inPoolSize	The size of the pool.
Return	None		

AllocFrom()

Purpose	Allocate memory from the pool.		
Access	Private		
Prototype	<code>void* AllocFrom (SFreeBlockInfo* inBlockInfo, UInt32 inByteCount);</code>		
Parameters	The parameters for this method are:		

	SFreeBlockInfo*	inBlockInfo	The block information.
	UInt32	inByteCount	The number of bytes.
Return	The buffer pointer.		

DisposePtr()

Purpose	Free a non-relocatable block from this memory pool. May be called at interrupt time.		
Access	Public		
Prototype	void DisposePtr (void* inPtr);		
Parameters	The parameters for this method are:		
	void*	inPtr	The pointer.
Return	None		

FreeMem()

Purpose	Return the total amount of free memory in this memory pool. May be called at interrupt time.
Access	Public
Prototype	<code>UInt32 FreeMem (void);</code>
Parameters	None
Return	The amount of free memory in the pool.

GetPtrSize()

Purpose	Return the size of the non-relocatable block at <code>inPtr</code> . NOTE: The size will be rounded up to the nearest multiple of 4. May be called at interrupt time.	
Access	Public	
Prototype	<code>UInt32 GetPtrSize (void* inPtr);</code>	
Parameters	The parameters for this method are:	
	<code>void*</code>	<code>inPtr</code> The pointer.
Return	The size of the memory block.	

MakeFreeBlock()

Purpose	Create a free block.	
Access	Private	
Prototype	<code>void MakeFreeBlock (void* inBlock, UInt32 inByteCount);</code>	
Parameters	The parameters for this method are:	
	<code>void*</code>	<code>inBlock</code> The pointer.
	<code>UInt32</code>	<code>inByteCount</code> The byte count.
Return	None	

NewPtr()

Purpose	Allocate a new non-relocatable block within the reentrant memory pool. May be called at interrupt time.
---------	---

Access	Public			
Prototype	<code>void* NewPtr (UInt32 inByteCount);</code>			
Parameters	<div>The parameters for this method are:<table><tr><td>UInt32</td><td>inByteCount</td><td>The byte count.</td></tr></table></div>	UInt32	inByteCount	The byte count.
UInt32	inByteCount	The byte count.		
Return	The pointer.			

NewPtrClear()

Purpose	Allocate a new non-relocatable block and clear it. May be called at interrupt time.		
Access	Public		
Prototype	<code>void* NewPtrClear (UInt32 inByteCount);</code>		
Parameters	The parameters for this method are:		
	UInt32	inByteCount	The byte count.
Return	The pointer.		

TotalMem()

Purpose	Return the total amount of memory used by this memory pool. May be called at interrupt time.
Access	Public
Prototype	<code>UInt32 TotalMem (void);</code>
Parameters	None
Return	The total memory in the pool.

LReentrantMemoryPool

mMemoryPools

Purpose	The pools.
Access	Private
Prototype	<code>LInterruptSafeList mMemoryPools;</code>

mFreeBlocks;

Purpose	The free blocks in the pool.
Access	Private
Prototype	<code>LInterruptSafeList mFreeBlocks;</code>

LScroller

Overview LScroller is a PowerPlant class that is used for scrolling views. Scroller views implement all the functionality of scroll bars, including creation, resizing, moving, hiding, showing, enabling and other operations. Sometimes you will see scrolling views referred to simply as “Scrollers” in the PowerPlant documentation.

The only functional difference between [LActiveScroller](#) and [LScroller](#) is that LActiveScroller supports dynamic scrolling of the view

Methods The methods in this class are:

LScroller()	~LScroller()
ActivateSelf()	AdjustScrollBars()
DeactivateSelf()	DrawSelf()
ExpandSubPane()	FinishCreateSelf()
HasHorizontalScrollBar()	HasVerticalScrollBar()
HorizSBarAction()	HorizScroll()
InstallView()	ListenToMessage()
MakeScrollBars()	ResizeFrameBy()
SubImageChanged()	VertSBarAction()
VertScroll()	

Data Members The data members in this class are:

mScrollingView	mVerticalBar
mHorizontalBar	mScrollingViewID

Operation The most important feature of a Scroller view is that it has only one subpane, which is a view. The subview may contain an arbitrary number of panes and views. The net effect is that the Scroller view may contain any number of panes of any type.

Source files (Pane Classes)

LScroller.h

LScroller

LScroller.cp

Ancestors [LListener](#)

[LPane](#)

[LView](#)

LScroller()

Purpose The constructor creates objects from the passed-in parameters.

Access Public

Prototype

```
LScroller();  
LScroller( const LScroller &inOriginal );  
LScroller( const SPaneInfo &inPaneInfo,  
const SViewInfo &inViewInfo,  
SInt16 inHorizBarLeftIndent,  
SInt16 inHorizBarRightIndent,  
SInt16 inVertBarTopIndent,  
SInt16 inVertBarBottomIndent,  
LView *inScrollingView );  
LScroller( LStream *inStream );
```

Parameters These constructors have the following parameters:

const LScroller&	inOriginal	A reference to the LScroller object you want to copy.
const SPaneInfo&	inPaneInfo	A reference to the SPaneInfo object that is the super view.
const SViewInfo&	inViewInfo	A reference to the SViewInfo object that contains information about the SuperView.
SInt16	inHorizBarLeft Indent	The indentation to use on the left side for the horizontal scrolling. A good initial value for this parameter is 15.
SInt16	inHorizBar- RightIndent	The indentation to use on the right side for the horizontal scrolling. A good initial value for this parameter is 15.

SInt16	inVertBarTop- Indent	The indentation to use on the top for the vertical scrolling. A good initial value for this parameter is 15.
SInt16	inVertBarBottomIndent	The indentation to use on the bottom for the vertical scrolling. A good initial value for this parameter is 15.
LView	inScrollingView	A pointer to the view that corresponds to this Scroller.
LStream*	inStream	A pointer to a stream object that contains the information to create the LScroller object.

~LScroller()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LScroller();</code>

ActivateSelf()

Purpose	This method activates the Scroller, and show scroll bars that were hidden when deactivated. This method is an override of ActivateSelf() in the LPane class.
---------	--

AdjustScrollBars()

Purpose	This method adjusts the scroll bars (value, min, and max) according to the current state of the Scroller and ScrollingView.
Access	Virtual, Public
Prototype	<code>virtual void AdjustScrollBars();</code>

LScroller

Parameters	None
------------	------

Return	None
--------	------

DeactivateSelf()

Purpose	This method deactivates the Scroller. According to Mac OS Human Interface Guidelines, scroll bars in inactive windows are hidden. This method is an override of DeactivateSelf() in the LPane class.
---------	--

DrawSelf()

Purpose	This method draws a Scroller. This method is an override of DrawSelf() in the LPane class.
---------	--

ExpandSubPane()

Purpose	Expand a SubPane, which should be the Scroller's ScrollingView, to fill the interior of a Scroller. This method is an override of ExpandSubPane() in the LView class.
---------	---

FinishCreateSelf()

Purpose	Finish creation of a Scroller by installing its ScrollingView. This method is an override of FinishCreateSelf() in the LPane class.
---------	---

HasHorizontalScrollBar()

Purpose	This method returns a Boolean indicating whether mHorizontalBar is nil or not.
Access	Inline, Public
Prototype	<code>Boolean HasHorizontalScrollBar();</code>
Parameters	None
Return	Boolean indicating mHorizontalBar != nil

HasVerticalScrollBar()

Purpose	This method returns a Boolean indicating whether mVerticalBar is nil or not.
Access	Inline, Public
Prototype	<code>Boolean HasVerticalScrollBar();</code>
Parameters	None
Return	Boolean indicating mVerticalBar != nil

HorizSBarAction()

Purpose	Mac OS Toolbox callback function for the action to take while tracking a mouse click in a horizontal scroll bar.
Access	Static, Protected
Prototype	<code>static pascal void HorizSBarAction(ControlHandle inMacControl, SInt16 inPart);</code>
Parameters	This method has the following parameters:

LScroller

	ControlHandle	inMacControl	This is the handle to the Mac OS control.
	SInt16	inPart	This is an enumeration, like kControl- UpButtonPart. See the Mac OS system header file named Controls.h for a list of these.
Return	None		

HorizScroll()

Purpose	This method is called to scroll horizontally while clicking and holding inside the horizontal scroll bar.		
Access	Virtual, Public		
Prototype	virtual void HorizScroll(SInt16 inPart);		
Parameters	This method has the following parameter:		
	SInt16	inPart	This is an enumeration, like kControl- UpButtonPart. See the Mac OS system header file named Controls.h for a list of these.
Return	None		

InstallView()

Purpose	This method installs a Scrolling View within this Scroller.
Access	Virtual, Public

Prototype	<code>virtual void InstallView(LView *inScrollingView);</code>	
Parameters	This method has the following parameter:	
	<code>LView *</code>	<code>inScrollingView</code> This is a pointer to the scrolling view to install in the scroller.
Return	None	

ListenToMessage()

Purpose	This method responds to messages from Broadcasters. The ScrollBars of a Scroller broadcast a message after the user drags the thumb. This method is an override of the pure virtual method ListenToMessage() in the LListener class.	
Access	Virtual, Public	
Prototype	<code>virtual void ListenToMessage(MessageT inMessage, void *ioParam);</code>	
Parameters	This method has the following parameters:	
	<code>MessageT</code>	<code>inMessage</code> This is the message that is broadcast from the framework. If the message is <code>msg_ThumbDragged</code> then processing is required.
	<code>void*</code>	<code>ioParam</code> This is a data block pointer that accompanies the message.
Return	None	
Remarks	You must supply an implementation for this method since it is a pure virtual in the base class.	

MakeScrollBars()

Purpose	Create standard Mac OS control objects for the horizontal and/or vertical scroll bars of a Scroller.	
Access	Private	
Prototype	<pre>void MakeScrollBars(SInt16 inHorizBarLeftIndent, SInt16 inHorizBarRightIndent, SInt16 inVertBarTopIndent, SInt16 inVertBarBottomIndent);</pre>	
Parameters	This method has the following parameters:	
	SInt16 inHorizBarLeftIndent	The indentation to use on the left side for the horizontal scrolling. A good initial value for this parameter is 15.
	SInt16 inHorizBarRightIndent	The indentation to use on the right side for the horizontal scrolling. A good initial value for this parameter is 15.
	SInt16 inVertBarTopIndent	The indentation to use on the top for the vertical scrolling. A good initial value for this parameter is 15.
	SInt16 inVertBarBottomIndent	The indentation to use on the bottom for the vertical scrolling. A good initial value for this parameter is 15.
Return	None	

ResizeFrameBy()

Purpose	Change the Frame size by the specified amounts. This method is an override of ResizeFrameBy() in the LPane class.
---------	---

SubImageChanged()

Purpose	Adjust state when the Image of the ScrollingView changes. Scroll bar settings depend on the ScrollingView Image, so they must be adjusted to match the current state. This method is an override of SubImageChanged() in the LView class.			
Access	Virtual, Public			
Prototype	<code>virtual void SubImageChanged(LView *inSubView);</code>			
Parameters	<div>This method has the following parameter:</div> <table><tr><td>LView*</td><td>inSubView</td><td>This is a pointer to the scrolling view to install in the scroller.</td></tr></table>	LView*	inSubView	This is a pointer to the scrolling view to install in the scroller.
LView*	inSubView	This is a pointer to the scrolling view to install in the scroller.		
Return	None			

VertSBarAction()

Purpose	Mac OS Toolbox callback function for the action to take while tracking a mouse click in a vertical scroll bar.
Access	Static, Protected
Prototype	<code>static pascal void VertSBarAction(ControlHandle inMacControl, SInt16 inPart);</code>
Parameters	This method has the following parameters:

LScroller

	ControlHandle	inMacControl	This is the handle to the Mac OS control.
	SInt16	inPart	This is an enumeration, like kControlUpButtonPart. See the Mac OS system header file named Controls.h for a list of these.
Return	None		

VertScroll()

Purpose	This method is called to scroll vertically while clicking and holding inside the vertical scroll bar.		
Access	Virtual, Public		
Prototype	virtual void VertScroll(SInt16 inPart);		
Parameters	This method has the following parameter:		
	SInt16	inPart	This is an enumeration, like kControlUpButtonPart. See the Mac OS system header file named Controls.h for a list of these.
Return	None		

mScrollingView

Purpose	This data member is a pointer to the ScrollingView.
Access	Protected

Prototype	<code>LView *mScrollingView;</code>
-----------	-------------------------------------

mVerticalBar

Purpose	This data member is a pointer to the standard control that is a vertical scroll bar.
---------	--

Access	Protected
--------	-----------

Prototype	<code>LStdControl *mVerticalBar;</code>
-----------	---

mHorizontalBar

Purpose	This data member is a pointer to the standard control that is a horizontal scroll bar.
---------	--

Access	Protected
--------	-----------

Prototype	<code>LStdControl *mHorizontalBar;</code>
-----------	---

mScrollingViewID

Purpose	This data member is the resource ID of the ScrollingView.
---------	---

Access	Protected
--------	-----------

Prototype	<code>PaneIDT mScrollingViewID;</code>
-----------	--

LSemaphore

Overview	LSemaphore is a PowerPlant class that is used for implementing semaphores. Most of the methods and members are internal to PowerPlant.	
Methods	The methods in this class are:	
	LSemaphore()	~LSemaphore()
	BlockThread()	InitSemaphore()
	Signal()	UnblockAll()
	UnblockThread()	Wait()
	operator =()	
Data Members	The data members in this class are:	
	mExcessSignals0	mThreads
Operation	The only two methods you will typically concern yourself with are Wait() and Signal() .	
Source files	(Threads Classes)	
	LSemaphore.h	
	LSemaphore.cp	
See also	LEventSemaphore	
	LMutexSemaphore	

LSemaphore()

Purpose	The constructor creates the object.
Access	Public
Prototype	<code>LSemaphore() ;</code> <code>LSemaphore(SInt32 initialCount) ;</code>

LSemaphore

Copy Constructor

`LSemaphore(const LSemaphore&);`

Parameters SInt32 containing the initial count to set the semaphore at, or a semaphore to make a copy of.

~LSemaphore()

Purpose The destructor destroys the LSemaphore object.

Access Virtual, Public

Prototype `~LSemaphore();`

BlockThread()

Purpose Block the current thread for the given number of milliseconds, refer to [Wait\(\)](#).

NOTE: The current thread must be in a 1-level-deep critical section upon entering this function!

Access Protected

Prototype `ExceptionCode BlockThread(SInt32 milliseconds);`

Parameters SInt32 containing the number of msec to block for.

Return An exception code

InitSemaphore()

Purpose Initializes the fields of a semaphore.

Access Private

Prototype `void InitSemaphore(SInt32 initialCount);`

Parameters	The initial count to set the semaphore to.
Return	None

Signal()

Purpose	Release a semaphore. If any threads are waiting on the semaphore, one of them is unblocked.
Access	Virtual, Public
Prototype	<code>virtual void Signal();</code>
Parameters	None
Return	None

UnblockAll()

Purpose	Unblock all of the threads waiting on this semaphore, passing them the given error code.
Access	Protected
Prototype	<code>void UnblockAll(ExceptionCode error);</code>
Parameters	The exception code
Return	None

UnblockThread()

Purpose	Unblock the thread pointed to by <code>qEl</code> , passing it the given error code.
---------	--

NOTE: The current thread must be in a critical section upon entering this method!

LSemaphore

Access	Protected		
Prototype	<code>LThread* UnblockThread(QElemPtr qEl, ExceptionCode error);</code>		
Parameters	The parameters for this method are:		
	QElemPtr	qEl	The queue element.
	ExceptionCode	error	The exception code.
Return	A pointer to the thread		

Wait()

Purpose	<p>Wait for a semaphore to become available. An optional argument specifies how long the caller is willing to wait for the semaphore to become available. Possible values are:</p> <ul style="list-style-type: none">• semaphore_WaitForeverWait indefinitely (default).• semaphore_NoWaitDo not wait. If the semaphore is unavailable, return <code>errSemaphoreTimedOut</code>.• other (> 0)Wait time in milliseconds <p>If the time interval expires before the semaphore becomes available, this function returns <code>errSemaphoreTimedOut</code>.</p>		
Access	Virtual, Public		
Prototype	<code>ExceptionCode Wait(SInt32 milliseconds);</code>		
Parameters	The time to wait in msec.		
Return	An exception code.		

operator =()

Purpose	The assignment operator
Access	Private

Prototype	<code>LSemaphore&operator = (const LSemaphore&);</code>
Parameters	A reference to a semaphore
Return	A reference to a semaphore

mExcessSignals0

Purpose	Excess signal / thread count
Access	Protected
Prototype	<code>SInt32 mExcessSignals;</code>

mThreads

Purpose	List of waiting threads
Access	Protected
Prototype	<code>QHdr mThreads;</code>

LSendQueue

Overview LSendQueue is a PowerPlant class that queues data for asynchronous style data sending for MacTCP.

Methods The methods in this class are:

LSendQueue()	~LSendQueue()
Append()	GetBlockingDataSize()
GetMaxPendingRelease()	Int_InternalSend()
Int_SendComplete()	InternalClearReleaseQueue()
IsBusy()	KillQueue()
NotifyRelease()	Run()
SetBlockingDataSize()	SetMaxPendingRelease()
WaitingDataSize()	

Data Members The data members in this class are:

mWaitingQueue	mPendingQueue
mReleaseQueue	mContinue
mDeleteData	mEndpointDead
mBusy	mReleaseWaiting
mEndpoint	mThread
mMaxPendingRelease	mBlockingDataSize

Operation This class is used primarily for Open Transport-style “acked” data transfers which are not native to MacTCP.

Source files (Networking Classes)

LSendQueue.h

LSendQueue.cp

See also [LMacTCPTCPSendQueue](#)

[LMacTCPUDPSendQueue](#)

LSendQueue

LSendQueue()

Purpose	The constructor creates the object. This constructor inherits from LThread's constructor.		
Access	Public		
Prototype	<code>LSendQueue (LEndpoint * inEndpoint);</code>		
Parameters	This constructor has the following parameter:		
	LEndpoint*	inEndpoint	This is the endpoint.

~LSendQueue()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LSendQueue();</code>

Append()

Purpose	Perform an append on the queue.		
Access	Virtual, Public		
Prototype	<code>virtual void Append(LInterruptSafeListMember* inItem);</code>		
Parameters	This method has the following parameter:		
	LInterruptSafeListMember*	inItem	This is the endpoint.
Return	None		

GetBlockingDataSize()

Purpose	This method returns the value of the mBlockingDataSize data member.
Access	Virtual, Public
Prototype	<code>virtual UInt32 GetBlockingDataSize() const;</code>
Parameters	None
Return	The size of the blocking data.

GetMaxPendingRelease()

Purpose	This method returns the value of the mMaxPendingRelease data member.
Access	Virtual, Public
Prototype	<code>virtual UInt32 GetMaxPendingRelease() const;</code>
Parameters	None
Return	The value of mMaxPendingRelease .

Int_InternalSend()

Purpose	This method is pure virtual, and you must supply an implementation for it.
Access	Pure virtual, Protected
Prototype	<code>virtual void Int_InternalSend() = 0;</code>
Parameters	None
Return	None

Int_SendComplete()

Purpose	This routine <i>*might*</i> be called at interrupt time.
Access	Virtual, Protected
Prototype	<code>virtual void Int_SendComplete();</code>
Parameters	None
Return	None

InternalClearReleaseQueue()

Purpose	Clear the release queue.
Access	virtual Virtual, Protected
Prototype	<code>void InternalClearReleaseQueue();</code>
Parameters	None
Return	None

IsBusy()

Purpose	Returns the value of the mBusy data member.
Access	Virtual, Public
Prototype	<code>virtual Boolean IsBusy();</code>
Parameters	None
Return	The value of mBusy .

KillQueue()

Purpose	Clear the queue and perform other necessary cleanup.
Access	Virtual, Public
Prototype	<code>virtual void KillQueue();</code>
Parameters	None
Return	None

NotifyRelease()

Purpose	This is a pure virtual method that you must provide code for.		
Access	Pure virtual, Protected		
Prototype	<code>virtual void NotifyRelease(LSendData* inData) = 0;</code>		
Parameters	This method has the following parameter:		
	LSendData*	inData	This is the data to send.
Return	None		

Run()

Purpose	Run only handles releasing data and the ACK'ing notifications. The data is sent via a chained notifier routine (started via Append()).		
Access	Virtual, Public		
Prototype	<code>virtual void* Run();</code>		
Parameters	None		
Return	Always returns nil.		

SetBlockingDataSize()

Purpose	Sets the value of mBlockingDataSize .		
Access	Virtual, Public		
Prototype	<code>virtual void SetBlockingDataSize(UInt32 inReleaseSize);</code>		
Parameters	This method has the following parameter:		
	UInt32	inReleaseSize	The data buffer size.
Return	None		

SetMaxPendingRelease()

Purpose	Sets the value of mMaxPendingRelease .		
Access	Virtual, Public		
Prototype	<code>virtual void SetMaxPendingRelease(UInt32 inReleaseSize);</code>		
Parameters	This method has the following parameter:		
	UInt32	inReleaseSize	The data buffer size.
Return	None		

WaitingDataSize()

Purpose	Returns the size of the pending data in the queue.		
Access	Virtual, Public		
Prototype	<code>virtual UInt32 WaitingDataSize();</code>		
Parameters	None		

Return	The data size.
--------	----------------

mWaitingQueue

Purpose	The waiting queue.
---------	--------------------

Access	Protected
--------	-----------

Prototype	<code>LInterruptSafeList mWaitingQueue;</code>
-----------	--

mPendingQueue

Purpose	The pending queue.
---------	--------------------

Access	Protected
--------	-----------

Prototype	<code>LInterruptSafeList mPendingQueue;</code>
-----------	--

mReleaseQueue

Purpose	The release queue.
---------	--------------------

Access	Protected
--------	-----------

Prototype	<code>LInterruptSafeList mReleaseQueue;</code>
-----------	--

mContinue

Purpose	Whether to continue.
---------	----------------------

Access	Protected
--------	-----------

Prototype	<code>Boolean mContinue;</code>
-----------	---------------------------------

LSendQueue

mDeleteData

Purpose	Whether to delete the data or not.
Access	Protected
Prototype	<code>Boolean mDeleteData;</code>

mEndpointDead

Purpose	Whether the endpoint is dead or not.
Access	Protected
Prototype	<code>Boolean mEndpointDead;</code>

mBusy

Purpose	Whether busy or not.
Access	Protected
Prototype	<code>Boolean mBusy;</code>

mReleaseWaiting

Purpose	Whether a release is waiting.
Access	Protected
Prototype	<code>UInt32 mReleaseWaiting;</code>

mEndpoint

Purpose	The endpoint.
Access	Protected
Prototype	<code>LEndpoint * mEndpoint;</code>

mThread

Purpose	The thread.
Access	Protected
Prototype	<code>LThread * mThread;</code>

mMaxPendingRelease

Purpose	The size of the maximum pending release buffer.
Access	Protected
Prototype	<code>UInt32 mMaxPendingRelease;</code>

mBlockingDataSize

Purpose	The size of the blocking data buffer.
Access	Protected
Prototype	<code>UInt32 mBlockingDataSize;</code>

LSharable

Overview	LSharable is a mix-in PowerPlant class that is used for reference counted objects.
Methods	<div>The methods in this class are:</div> <div><div>LSharable()</div><div>AddUser()</div><div>NoMoreUsers()</div></div> <div><div>~LSharable()</div><div>GetUseCount()</div><div>RemoveUser()</div></div>
Data Members	<div>The data members in this class are:</div> <div>mUseCount</div>
Source files	<div>(Feature Classes)</div> <div>LSharable.h</div> <div>LSharable.cp</div>
Ancestors	LModelObject

LSharable()

Purpose	The constructor creates objects.
Access	Public
Prototype	<code>LSharable() ;</code>

~LSharable()

Purpose	The destructor destroys the object.
Access	Virtual, Protected
Prototype	<code>virtual ~LSharable() ;</code>

AddUser()

Purpose	Add a user to an object.
Access	Virtual, Public
Prototype	<code>virtual void AddUser(void* inUser);</code>
Parameters	A pointer to the user.
Return	None

GetUseCount()

Purpose	Retrieve the use count of the object.
Access	Virtual, Public
Prototype	<code>virtual SInt32 GetUseCount();</code>
Parameters	None
Return	The number of users.

NoMoreUsers()

Purpose	Internal function called when use count drops to zero.
Access	Virtual, Protected
Prototype	<code>virtual void NoMoreUsers();</code>
Parameters	None
Return	None

RemoveUser()

Purpose	Specify that a user is no longer sharing a Sharable object. This method ignores the user and just decrements the use count. Override if you wish to keep track users. NOTE: Do not throw an exception from this method.
Access	Virtual, Public
Prototype	<code>virtual void RemoveUser (void* inUser);</code>
Parameters	A pointer to the user.
Return	None

mUseCount

Purpose	The user count.
Access	Protected
Prototype	<code>SInt32mUseCount;</code>

LSharableModel

Overview	LSharableModel is a mix-in PowerPlant class that is used for reference counting LModelObjects.		
Methods	The methods in this class are:		
	LSharableModel()	~LSharableModel()	
	AddUser()	Finalize()	
	SuperDeleted()		
Data Members	There are no data members in this class.		
Source files	(AppleEvent Classes)		
	LSharableModel.h		
	LSharableModel.cp		
Ancestors	LModelObject		
	LSharable		

LSharableModel()

Purpose	The constructor creates objects from the passed-in parameters.		
Access	Public		
Prototype	<pre>LSharableModel () ; LSharableModel (LModelObject*inSuperModel, DescTypeinKind = typeNull) ;</pre>		
Parameters	The parameters for these constructors are:		
	LModelObject*	inSuperModel	A pointer to the Super object.
	DescType	inKind	The kind, which defaults to typeNull.

LSharableModel

~LSharableModel()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LSharableModel();</code>

AddUser()

Purpose	Add a user to the object. This method calls into the base class method AddUser() in LSharable also.		
Access	Virtual, Public		
Prototype	<code>virtual void AddUser(void *inUser);</code>		
Parameters	The parameter for this method is:		
	<code>void*</code>	<code>inUser</code>	A pointer to the user object.
Return	None		

Finalize()

Purpose	This calls the <code>Finalize()</code> method in the base class.
Access	Virtual, Public
Prototype	<code>virtual void Finalize();</code>
Parameters	None
Return	None

SuperDeleted()

Purpose	This is called when the Super Object is deleted.
Access	Virtual, Public
Prototype	<code>virtual void SuperDeleted(void);</code>
Parameters	None
Return	None

LSharedMemoryPool

Overview	LSharedMemoryPool is a PowerPlant class that implements static shared memory pools that are self-managing and create/delete as needed.		
Methods	The methods in this class are: LSharedMemoryPool() AddPoolUser() RemovePoolUser() ~LSharedMemoryPool() GetSharedPool()		
Data Members	The data members in this class are: sSharedMemoryPool		
Operation	A .		
Source files	(Networking Classes) LSharedMemoryPool.h LSharedMemoryPool.cp		
See also	LReentrantMemoryPool LSharable		

LSharedMemoryPool()

Purpose	The constructor creates the object. This constructor inherits from the LReentrantMemoryPool constructor.		
Access	Public		
Prototype	LSharedMemoryPool (UInt32 inPoolSize);		
Parameters	This constructor has the following parameter: UInt32 inPoolSize This is the size of the pool.		

LSharedMemoryPool

~LSharedMemoryPool()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LSharedMemoryPool();</code>

AddPoolUser()

Purpose	Adds a user to the pool. Can NOT be called at interrupt time.		
Access	Public		
Prototype	<code>void AddPoolUser(void*inUser, UInt32inPoolSize);</code>		
Parameters	This method has the following parameters:		
	<code>void*</code>	<code>inUser</code>	The user pointer.
	<code>UInt32</code>	<code>inPoolSize</code>	This is the size of the pool.
Return	None		

GetSharedPool()

Purpose	Returns pointer to static shared memory pool. Creates pool if necessary.		
Access	Static, Public		
Prototype	<code>static LSharedMemoryPool * GetSharedPool();</code>		
Parameters	None		
Return	The pointer to the shared memory pool.		

RemovePoolUser()

Purpose	Removes a user from the pool.		
Access	Public		
Prototype	<code>void RemovePoolUser(void*inUser);</code>		
Parameters	This method has the following parameters:		
	<code>void*</code>	<code>inUser</code>	The user pointer.
Return	None		

sSharedMemoryPool

Purpose	The shared memory poolk pointer.
Access	Static, Private
Prototype	<code>LSharedMemoryPool *sSharedMemoryPool;</code>

LSharedQueue

Overview	LSharedQueue is a PowerPlant class that combines a queue with a semaphore.	
Methods	The methods in this class are: LSharedQueue() DoForEach() NextPut() ~LSharedQueue() Next() Remove()	
Data Members	The data members in this class are: mAvailable	
Operation	By combining a queue with a semaphore, you can protect the data in the queue. This allows two or more threads to share a common data queue.	
Source files	(_Advanced Classes:Threads Classes) LSharedQueue.h LSharedQueue.cp	
See also	LMutexSemaphore LSemaphore LQueue	

LSharedQueue()

Purpose	The constructor creates the object.
Access	Public
Prototype	<code>LSharedQueue () ;</code>

LSharedQueue

~LSharedQueue()

Purpose	The destructor destroys the LSharedQueue object.
Access	Virtual, Public
Prototype	<code>virtual ~LSharedQueue();</code>

DoForEach()

Purpose	Execute a user-supplied function for each element in the queue.		
Access	Virtual, Public		
Prototype	<code>virtual void DoForEach(LQueueIterator proc, void* arg);</code>		
Parameters	The parameters for this method are:		
	LQueueIterator	proc	The queue iterator.
	void*	arg	The function to perform.
Return	None		

Next()

Purpose	Removes and returns the first element in the queue. milliseconds is an optional argument which specifies how long the caller is willing to wait for an element. Possible values are: <ul style="list-style-type: none">• <code>sharedQueue_WaitForever</code> Wait indefinitely (default).• <code>sharedQueue_NoWait</code> Do not wait. If the queue is empty, return NULL.• other (> 0) Wait time in milliseconds		
Access	Virtual, Public		

Prototype	<code>virtual LLink* Next(SInt32 milliseconds = sharedQueue_WaitForever);</code>
Parameters	The number of msec to wait.
Return	A pointer to an LLink object.

NextPut()

Purpose	Adds the given element to the end of the queue.
Access	Virtual, Public
Prototype	<code>virtual void NextPut(LLink* inLinkP);</code>
Parameters	A pointer to the LLink to put.

Remove()

Purpose	Remove an arbitrary element from the queue. This function traverses the entire queue, looking for the given queue element. If the element is found, it is removed from the queue.
Access	Virtual, Public
Prototype	<code>Boolean Remove(LLink* inLinkP);</code>
Parameters	None
Return	Returns a Boolean indicating if the element was found.

mAvailable

Purpose	Indicates whether the data is available or not.
Access	Protected
Prototype	<code>LSemaphore mAvailable;</code>

LSingleDoc

Overview	LSingleDoc is a PowerPlant class that is used as a simple extension of LDocument . You must derive your own methods from this object to provide functionality.	
Methods	The methods in this class are: LSingleDoc() AllowSubRemoval() GetFile() MakeCurrent() ~LSingleDoc() GetDescriptor() GetWindow() UsesFileSpec()	
Data Members	The data members in this class are: mWindow mFile	
Source files	(Commander Classes) LSingleDoc.h LSingleDoc.cp	
See also	LCommander LDocument	

LSingleDoc()

Purpose	The constructor creates the object.
Access	Public
Prototype	<code>LSingleDoc () ;</code> <code>LSingleDoc (LCommander*inSuper) ;</code>
Parameters	An LCommander super object.

LSingleDoc

~LSingleDoc()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LSingleDoc();</code>

AllowSubRemoval()

Purpose	This is an override of AllowSubRemoval() in LCommander .
---------	--

GetDescriptor()

Purpose	This is an override of GetDescriptor() in LDocument .
---------	---

GetFile()

Purpose	This returns the file pointer held in mFile .
Access	Public
Prototype	<code>LFile*GetFile() const;</code>
Parameters	None
Return	The file pointer.

GetWindow()

Purpose	This method returns the window pointer held in mWindow .
---------	--

Access	Public
Prototype	<code>LWindow*GetWindow() const;</code>
Parameters	None
Return	None

MakeCurrent()

Purpose	Make this Document the current one by selecting its Window.
Access	Virtual, Public
Prototype	<code>virtual void MakeCurrent();</code>
Parameters	None
Return	None

UsesFileSpec()

Purpose	Returns whether the Document's File has the given FSSpec.
Access	Virtual, Public
Prototype	<code>virtual Boolean UsesFileSpec(const FSSpec&inFileSpec) const;</code>
Parameters	A FSSpec for the file.
Return	Return true if it has the FSSpec.

mWindow

Purpose	The window pointer.
Access	Protected

LSingleDoc

Prototype LWindow*mWindow;

mFile

Purpose The file pointer storage.

Access Protected

Prototype LFile*mFile;

LSIOUXAttachment

Overview	LSIOUXAttachment is a PowerPlant class that is used for using the SIOUX facilities within PowerPlant programs.
Methods	The methods in this class are: LSIOUXAttachment() ExecuteSelf()
Data Members	There are no data members in this class.
Operation	To learn more about SIOUX and how to use it, refer to the <i>MSL C Reference</i> on the CodeWarrior documentation CD.
Source files	(Support Classes) LSIOUXAttachment.h LSIOUXAttachment.cp
See also	LAttachment

LSIOUXAttachment()

Purpose	The constructor configures SIOUX to run within a PowerPlant program.
Access	Public
Prototype	LSIOUXAttachment () ;
Parameters	None
Remarks	If you want to change the default SIOUX values, set the fields of the global SIOUXSettings struct in your code before creating the SIOUX Window (normally the before the first call to printf or cout).

ExecuteSelf()

Purpose	Send the Event to SIOUX. If SIOUX doesn't handle the event, SIOUXHandleOneEvent() returns 0, so mExecuteHost is true, meaning that the event will be dispatched to PowerPlant. If SIOUX does handle the event, SIOUXHandleOneEvent() returns a non-zero value, so mExecuteHost is false, meaning that the event will not be passed to PowerPlant.	
Access	Protected	
Prototype	virtual void ExecuteSelf(MessageT inMessage, void *ioParam);	
Parameters	The parameters for this method are:	
	MessageT inMessage	This parameter is currently unused for this method.
	void* ioParam	A pointer to a structure that contains Rect information for drawing.
Return	None	

LStdButton

Overview	LStdButton is a PowerPlant class that is used for managing Mac OS push buttons.
Methods	The methods in this class are: LStdButton() HotSpotResult()
Data Members	There are no data members in this class.
Operation	Using this class, you specify the text to display in the button, the text traits resource ID to use for the text display, and the message to send when the button is clicked.
Source files	(Pane Classes) LStdControl.h LStdControl.cp
Ancestors	The ancestors this class derives from are: LAttachable LControl LStdControl LBroadcaster LPane

LStdButton()

Purpose	The constructors create objects from the passed-in parameters.
Access	Public
Prototype	<pre>LStdButton(); LStdButton(const LStdButton &inOriginal); LStdButton(const SPaneInfo &inPaneInfo, MessageT inValueMessage, ResIDT inTextTraitsID, Str255 inTitle); LStdButton(const SPaneInfo &inPaneInfo, MessageT inValueMessage,</pre>

LStdButton

```
ResIDT inTextTraitsID,  
ControlHandle inMacControlH );  
LStdButton( LStream *inStream );
```

Parameters

These constructors have the following parameters:

const LStdButton&	inOriginal	A reference to the LStdButton object you want to copy.
const SPaneInfo&	inPaneInfo	A reference to the SPaneInfo object that is the super view.
MessageT	inValueMessage	The message sent when the button is depressed.
ResIDT	inTextTraitsID	The text traits resource ID for the button text.
Str255	inTitle	The text string to put on the button.
ControlHandle	inMacControlH	The handle for the Mac OS control.
LStream*	inStream	A pointer to a stream object that contains the information to create the LStdButton object.

HotSpotResult()

Purpose	This method responds to a click in the button by broadcasting the value of the button. This method is an override of HotSpotResult() in the LControl class.
---------	---

LStdCheckBox

Overview	LStdCheckBox is a PowerPlant class that is used for managing Mac OS check box controls.
Methods	The methods in this class are: LStdCheckBox() HotSpotResult()
Data Members	There are no data members in this class.
Operation	If the user clicks the control, the value toggles between one and zero.
Source files	(Pane Classes) LStdControl.h LStdControl.cp
Ancestors	The ancestors this class derives from are: LAttachable LControl LStdControl LBroadcaster LPane

LStdCheckBox()

Purpose	The constructors create objects from the passed-in parameters.
Access	Public
Prototype	<pre>LStdCheckBox(); LStdCheckBox(const LStdCheckBox &inOriginal); LStdCheckBox(const SPaneInfo &inPaneInfo, MessageT inValueMessage, SInt32 inValue, ResIDT inTextTraitsID, Str255 inTitle); LStdCheckBox(const SPaneInfo &inPaneInfo, MessageT inValueMessage, SInt32 inValue,</pre>

LStdCheckBox

```
ResIDT inTextTraitsID,  
ControlHandle inMacControlH );  
LStdCheckBox( LStream *inStream );
```

Parameters	These constructors have the following parameters:	
const LStdCheckBox&	inOriginal	A reference to the LStdCheckBox object you want to copy.
const SPaneInfo&	inPaneInfo	A reference to the SPaneInfo object that is the super view.
MessageT	inValueMessage	The message sent when the check box is clicked.
SInt32	inValue	
ResIDT	inTextTraitsID	The text traits resource ID for the button text.
Str255	inTitle	The text string to put on the button.
ControlHandle	inMacControlH	The handle for the Mac OS control.
LStream*	inStream	A pointer to a stream object that contains the information to create the LStdCheckBox object.

HotSpotResult()

Purpose	This method responds to a click in the check box by toggling the value of the control between checked (value = 0) and unchecked (value = 1). This method is an override of HotSpotResult() in the LControl class.
---------	---

LStdControl

Overview LStdControl is a PowerPlant class that encapsulates the standard Mac OS control behavior.

Methods The methods in this class are:

<u>LStdControl()</u>	<u>~LStdControl()</u>
<u>AlignControlRect()</u>	<u>CalcBigValue()</u>
<u>CalcSmallValue()</u>	<u>CreateFromCNTL()</u>
<u>DisableSelf()</u>	<u>DrawSelf()</u>
<u>EnableSelf()</u>	<u>FindHotSpot()</u>
<u>FocusDraw()</u>	<u>GetDescriptor()</u>
<u>GetMacControl()</u>	<u>GetTrackingControl()</u>
<u>HideSelf()</u>	<u>HotSpotAction()</u>
<u>HotSpotResult()</u>	<u>InitStdControl()</u>
<u>MoveBy()</u>	<u>PointInHotSpot()</u>
<u>ResizeFrameBy()</u>	<u>SetActionProc()</u>
<u>SetDescriptor()</u>	<u>SetMaxValue()</u>
<u>SetMinValue()</u>	<u>SetStdMinAndMax()</u>
<u>SetThumbFunc()</u>	<u>SetValue()</u>
<u>ShowSelf()</u>	<u>TrackHotSpot()</u>
<u>ValueIsInStdRange()</u>	

Data Members The data members in this class are:

<u>mMacControlH</u>	<u>mThumbFunc</u>
<u>mControlKind</u>	<u>mTextTraitsID</u>
<u>mUsingBigValues</u>	<u>sTrackingControl</u>

Operation The specific design of this class supports scroll bar controls and controls that use custom control definitions (CDEFs). This class also

LStdControl

forms the basis for the standard button, check box, pop-up menu, and radio button classes.

For detailed information on Mac OS controls, refer to *Inside Mac: MacintoshToolbox Essentials*, published by Addison-Wesley.

Source files (Pane Classes)

LStdControl.h

LStdControl.cp

Ancestors The ancestors for this class are:

[LAttachable](#)

[LControl](#)

[LBroadcaster](#)

[LPane](#)

LStdControl()

Purpose The constructors create objects from the passed-in parameters. Many of these constructors call [LControl](#) base class constructors to force a portion of the initialization chores.

Access Public

Prototype

```
LStdControl();  
LStdControl( SInt16 inControlKind );  
LStdControl( const SPaneInfo &inPaneInfo,  
MessageT inValueMessage,  
SInt32 inValue,  
SInt32 inMinValue,  
SInt32 inMaxValue,  
SInt16 inControlKind,  
ResIDT inTextTraitsID,  
ConstStringPtr inTitle,  
SInt32 inMacRefCon );  
LStdControl( const SPaneInfo &inPaneInfo,  
MessageT inValueMessage,  
SInt32 inValue,  
SInt32 inMinValue,  
SInt32 inMaxValue,
```

```

SInt16 inControlKind,
ResIDT inTextTraitsID,
ControlHandle inMacControlH );
LStdControl( LStream *inStream );
LStdControl( LStream *inStream, bool inDummy );

```

Parameters These constructors have the following parameters:

SInt16	inControlKind	The value to set for mControlKind .
const SPaneInfo&	inPaneInfo	A reference to the SPaneInfo object that is the super view.
MessageT	inValueMessage	The message sent when the control is interacted with.
SInt32	inValue	The value to set for the control.
SInt32	inMinValue	The minimum value to allow for the control.
SInt32	inMaxValue	The maximum value to allow for the control.
ResIDT	inTextTraitsID	The resource ID that contains the text traits for control text.
ConstStringPtr	inTitle	The title to use for the control.
SInt32	inMacRefCon	The value to put in the Mac OS refcon storage space.
ControlHandle	inMacControlH	The Mac OS handle to the control.
LStream*	inStream	A pointer to a stream object that contains the information to create the LControl object.
bool	inDummy	This parameter is unused for this constructor.

Remarks • LStdControl(SInt16 inControlKind)—Construct a StdControl for a particular kind of Mac OS Toolbox Control. On entry, the current Port must be the Window into which to install the Control.

LStdControl

~LStdControl()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LStdControl();</code>

AlignControlRect()

Purpose	Set the rectangle inside the Mac OS Toolbox ControlRecord to the Frame rectangle of the standard control.
Access	Protected
Prototype	<code>void AlignControlRect();</code>
Parameters	None
Return	None
Remarks	<ul style="list-style-type: none">• At (almost) all times, the <code>ctrlRect</code> field of the Mac OS Toolbox ControlRecord must be the same as the Frame rectangle of the StdControl object. We store directly to the ControlRecord rather than using the SizeControl and MoveControl traps because we want to bypass the automatic drawing (and the need to set the port and coordinate system) performed by those traps.• The <code>ctrlRect</code> is not the same as the Frame rectangle if the latter is outside the 16-bit QuickDraw coordinate space.

CalcBigValue()

Purpose	Map from Small to Big value when using 32 bit values. Call this function only when mUsingBigValues is true.
Access	Protected
Prototype	<code>SInt32 CalcBigValue(SInt16 inSmallValue);</code>

Parameters	This method has the following parameter:			
	<table><tr><td>SInt16</td><td>inSmallValue</td><td>The Small value to be converted.</td></tr></table>	SInt16	inSmallValue	The Small value to be converted.
SInt16	inSmallValue	The Small value to be converted.		
Return	SInt32 containing the Big value.			
Remarks	The control class uses Big (32 bit) values, but the Mac Control Manager only supports Small (16 bit) values. See the discussion for the SetStdMinAndMax() method.			

CalcSmallValue()

Purpose	Map from Big to Small value when using 32 bit values. Call this function only when mUsingBigValues is true.			
Access	Protected			
Prototype	<code>SInt16 CalcSmallValue(SInt32 inBigValue);</code>			
Parameters	<p>This method has the following parameter:</p> <table><tr><td><code>SInt16</code></td><td><code>inBigValue</code></td><td>The Big value to be converted.</td></tr></table>	<code>SInt16</code>	<code>inBigValue</code>	The Big value to be converted.
<code>SInt16</code>	<code>inBigValue</code>	The Big value to be converted.		
Return	<code>SInt16</code> containing the Small value.			
Remarks	The control class uses Big (32 bit) values, but the Mac Control Manager only supports Small (16 bit) values. Refer to the discussion for the SetStdMinAndMax() method.			

CreateFromCNTL()

Purpose	Create a StdControl from a 'CNTL' resource.
Access	Static, Public
Prototype	static LStdControl* CreateFromCNTL(ResIDT inCNTLid,

LStdControl

```
MessageT inValueMessage,  
ResIDT inTextTraitsID,  
LView *inSuperView);
```

Parameters This method has the following parameters:

ResIDT	inCNTLid	The resource ID of the CTNL resource.
MessageT	inValueMessage	The message that the control should send when it is changed.
ResIDT	inTextTraitsID	The resource ID of the text traits to use for drawing text.
LView*	inSuperView	The pointer to the SuperView that the control will belong to.

Return None

DisableSelf()

Purpose Disable a standard control.

Access Virtual, Protected

Prototype `virtual void DisableSelf();`

Parameters None

Return None

DrawSelf()

Purpose This method is an override of the base class method [DrawSelf\(\)](#) in [LPane](#). It draws the standard control.

EnableSelf()

Purpose	Enable a StdControl
Access	Virtual, Protected
Prototype	<code>virtual void EnableSelf();</code>
Parameters	None
Return	None

FindHotSpot()

Purpose	This method is an override of the base class method FindHotSpot() in LControl . It determines which hot spot, if any, contains the specified point.
---------	---

FocusDraw()

Purpose	This method is an override of the base class method FocusDraw() in LPane . It prepares for drawing in the Pane, and signals if the Pane has no SuperView.
---------	---

GetDescriptor()

Purpose	This method is an override of the base class method GetDescriptor() in LPane . It returns the Descriptor, which is the Title, of a standard control.
---------	--

GetMacControl()

Purpose	This method returns the value of mMacControlH .
Access	Inline, Public
Prototype	<code>ControlHandle GetMacControl() const;</code>
Parameters	None
Return	The ControlHandle held in mMacControlH .

GetTrackingControl()

Purpose	This method returns the value of sTrackingControl .
Access	Static, Inline, Public
Prototype	<code>static LStdControl* GetTrackingControl();</code>
Parameters	None
Return	A pointer to an LStdControl, containing the value of sTrackingControl .

HideSelf()

Purpose	This method is an override of the base class method GetDescriptor() in LPane . It hides a standard control.
---------	---

HotSpotAction()

Purpose	This method is an override of the base class method HotSpotAction() in LControl . It makes mouse tracking easier.
---------	---

HotSpotResult()

Purpose This method is an override of the base class method [HotSpotAction\(\)](#) in [LControl](#). It performs the result of clicking and releasing mouse inside a hot spot.

InitStdControl()

Purpose This is a private initializer which creates the Mac OS Toolbox Control.

Access Private

Prototype `void InitStdControl(SInt16 inControlKind,
ResIDT inTextTraitsID,
ConstStringPtr inTitle,
SInt32 inMacRefCon);`

Parameters This method has the following parameters:

SInt16	inControlKind	The value to set for mControlKind .
ResIDT	inTextTraitsID	The resource ID of the resource containing the text traits to use for drawing text.
ConstStringPtr	inTitle	The title to use for the control.
SInt32	inMacRefCon	The value to put in the Mac OS refcon storage space for the control.

Return None

MoveBy()

Purpose This method is an override of the base class method [FocusDraw\(\)](#) in [LPane](#). This method is an override to update the Mac ControlHandle. It moves the location of the Frame by the specified amounts.

PointInHotSpot()

Purpose This method is an override of the base class method [PointInHotSpot\(\)](#) in [LControl](#). It tells whether a particular point is inside a hot spot or not.

ResizeFrameBy()

Purpose This method is an override of the base class method [FocusDraw\(\)](#) in [LPane](#). This method is an override to update the Mac ControlHandle. It changes the Frame size by the specified amounts.

SetActionProc()

Purpose This method sets the control action proc (ControlActionUPP) for the control.

Access Public

Prototype `void SetActionProc(ControlActionUPP inActionProc);`

Parameters This method has the following parameter:

ControlActi onUPP	inActionProc	The procedure pointer to set the control action proc to.
----------------------	--------------	---

Return None

SetDescriptor()

Purpose This method is an override of the base class method [SetDescriptor\(\)](#) in [LPane](#). This method sets the Descriptor, which is the title, of a standard control.

SetMaxValue()

Purpose This method is an override of the base class method [SetMaxValue\(\)](#) in [LControl](#). It sets the maximum value that a control can have.

SetMinValue()

Purpose This method is an override of the base class method [SetMinValue\(\)](#) in [LControl](#). It sets the minimum value that a control can have.

SetStdMinAndMax()

Purpose Sets the minimum and maximum values for the standard Mac OS control associated with this object.

Access Public

Prototype `void SetStdMinAndMax();`

Parameters None

Return None

Remarks	<ul style="list-style-type: none">• The Control class uses Big (32 bit) values, but the Mac Control Manager only supports Small (16 bit) values. So this object class has to map between Big and Small values.• When either the minimum or maximum value is outside 16 bit range (below -32,768 or above 32,767), we set mUsingBigValues to true. In that case, you must call CalcSmallValue() and CalcBigValue() to map between the Small values used by the Mac Control Manager and the Big values used by this class.• The mapping strategy is as follows. When using Big values, we always set the minimum value for the Mac Control to zero. There are two cases for the maximum value, depending on whether the difference between the Big Max and Min is greater than or less than 32,767. If greater, we set the Mac Control maximum to 32,767. If less, we set the Mac Control maximum to that difference. These choices simplify the math for converting between Big and Small values.• The functions CalcSmallValue() and CalcBigValue() use the following equations to convert between Big and Small values. $\begin{array}{l} (1) \text{ BigMax} - \text{BigMin} > 32,767 \\ \text{SmallValue} = \text{BigValue} - \text{BigMin} \\ \text{-----} = \text{-----} \\ 32,767 \quad (\text{BigMax} - \text{BigMin}) \\ (2) \text{ BigMax} - \text{BigMin} \leq 32,767 \\ \text{SmallValue} = \text{BigValue} - \text{BigMin} \end{array}$
---------	---

SetThumbFunc()

Purpose	This method sets the value of the mThumbFunc data member.	
Access	Public	
Prototype	<code>void SetThumbFunc(ThumbActionUPP inThumbFunc);</code>	
Parameters	This method has the following parameter:	
	ThumbActionUPP inThumbFunc	The procedure pointer to set for the control's thumb action proc.
Return	None	

SetValue()

Purpose This method is an override of the base class method [SetValue\(\)](#) in [LPane](#). This method sets the value of a standard control. It overrides the inherited function to map from 32 bit numbers to the 16 bit numbers supported by the Mac OS Control Manager.

ShowSelf()

Purpose This method is an override of the base class method [ShowSelf\(\)](#) in [LPane](#). This method shows the control.

TrackHotSpot()

Purpose This method is an override of the base class method [TrackHotSpot\(\)](#) in [LControl](#). It tracks the mouse while it is down after clicking in a Control hot spot, and tells whether the mouse is released within the hot spot.

ValueIsInStdRange()

Purpose Indicates whether a value for a control is within the range supported by the Mac OS Control Manager.

Access Static, Protected

Prototype `static Boolean ValueIsInStdRange(SInt32 inValue);`

Parameters This method has the following parameter:

SInt32	inValue	The value to check against the control's range.
--------	---------	---

LStdControl

Return	Returns <code>true</code> if the value is in the range, else returns <code>false</code> .
--------	---

mMacControlH

Purpose	This data member stores the handle to the Mac OS control.
Access	Protected
Prototype	<code>ControlHandle mMacControlH;</code>

mThumbFunc

Purpose	This data member stores a pointer to the thumb action procedure for the control.
Access	Protected
Prototype	<code>ThumbActionUPP mThumbFunc;</code>

mControlKind

Purpose	This data member stores the value for the control kind. Refer to the Mac OS system header named <code>Controls.h</code> for more information on these enumerations.
Access	Protected
Prototype	<code>SInt16 mControlKind;</code>

mTextTraitsID

Purpose	This data member stores the value of the resource ID to use for text traits.
---------	--

Access	Protected
Prototype	ResIDT mTextTraitsID;

mUsingBigValues

Purpose	This data member remembers whether we are using Big values for the control or not.
Access	Protected
Prototype	Boolean mUsingBigValues;

sTrackingControl

Purpose	This data member stores the pointer to the current standard control that is being tracked.
Access	Static, Protected
Prototype	static LStdControl* sTrackingControl;

LStdPopupMenu

Overview	LStdPopupMenu is a PowerPlant class that is used for managing Mac OS pop-up menus.
Methods	<p>The methods in this class are:</p> <p>LStdPopupMenu()~LStdPopupMenu()</p> <p>DrawSelf()GetMacMenuH()</p> <p>InitStdPopupMenu()</p>
Data Members	There are no data members in this class.
Operation	LStdPopupMenu encapsulates the details of dealing with Mac OS pop-up menus. To create a new pop-up menu, create the required resources with the Constructor resource editor as described in <i>The PowerPlant Book</i> , then use a constructor to create your pop-up menu.
Source files	<p>(Pane Classes)</p> <p>LStdControl.h</p> <p>LStdControl.cp</p>
Ancestors	<p>The ancestors this class derives from are:</p> <p>LAttachableLControlLStdControl</p> <p>LBroadcasterLPane</p>

LStdPopupMenu()

Purpose	The constructor creates objects from the passed-in parameters.
Access	Public
Prototype	<pre>LStdPopupMenu(const SPaneInfo &inPaneInfo, MessageT inValueMessage, SInt16 inTitleOptions, ResIDT inMENUid, SInt16 inTitleWidth,</pre>

LStdPopupMenu

```
SInt16 inPopupVariation,  
ResIDT inTextTraitsID,  
Str255 inTitle,  
OSType inResTypeMENU,  
SInt16 inInitialMenuItem );  
LStdPopupMenu( const SPaneInfo &inPaneInfo,  
MessageT inValueMessage,  
SInt32 inMaxValue,  
ResIDT inTextTraitsID,  
ControlHandle inMacControlH );  
LStdPopupMenu( LStream *inStream );
```

Parameters These constructors have the following parameters:

const SPaneInfo&	inPaneInfo	A reference to the SPaneInfo object that is the super view.
MessageT	inValueMessage	The message sent when the control is interacted with.
SInt32	inTitleOptions	This is passed to LStdControl() .
ResIDT	inMENUid	The resource ID of the resource containing the menu items.
SInt16	inTitleWidth	This is passed to LStdControl() .
SInt16	inPopupVariation	This is passed to LStdControl() .
ResIDT	inTextTraitsID	The resource ID for the resource containing the text traits to apply to the pop-up menu text strings.
Str255	inTitle	The text string indicating the title of the pop-up menu.
OSType	inResTypeMENU	This is passed to LStdControl() .
SInt16	inInitialMenuItem	The number indicating which menu item should appear in the pop-up menu the first time it is drawn.

ControlHandle	inMacControlH	The handle to the Mac OS control.
LStream*	inStream	Pointer to a stream object that contains the information to create the LControl object.

~LStdPopupMenu()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LStdPopupMenu();</code>

DrawSelf()

Purpose	This method draws the pop-up menu, and is an override of DrawSelf() in LStdControl .
---------	--

GetMacMenuH()

Purpose	Return the Mac OS MenuHandle associated with a the pop-up menu.
Access	Virtual, Public
Prototype	<code>virtual MenuHandle GetMacMenuH();</code>
Parameters	None
Return	MenuHandle containing the Mac OS handle of the menu.

LStdPopupMenu

InitStdPopupMenu()

Purpose	This is a private initializer method used by the internals of the class to set up the pop-up menu.		
Access	Private		
Prototype	<code>void InitStdPopupMenu(SInt16 inInitialMenuItem);</code>		
Parameters	This method has the following parameter:		
	<code>SInt16 inInitialMenuItem</code>	6	The number of the initial menu item to show in the popup menu.
Return	None		

LStdRadioButton

Overview	LStdRadioButton is a PowerPlant class that is used for managing Mac OS radio buttons.
Methods	The methods in this class are: LStdRadioButton() HotSpotResult() SetValue()
Data Members	There are no data members in this class.
Operation	This class is a very simple extension of LStdControl . A click on a radio button sets the value to one. The button then broadcasts a message that it has been clicked. The broadcast message includes a pointer to the button object that was clicked.
Source files	(Pane Classes) LStdControl.h LStdControl.cp
Ancestors	The ancestors this class derives from are: LAttachable LControl LStdControl LBroadcaster LPane
See Also	LRadioGroup

LStdRadioButton()

Purpose	The constructor creates objects from the passed-in parameters.
Access	Public
Prototype	<pre>LStdRadioButton(); LStdRadioButton(const SPaneInfo &inPaneInfo, MessageT inValueMessage, SInt32 inValue,</pre>

LStdRadioButton

```
ResIDT inTextTraitsID,  
Str255 inTitle );  
LStdRadioButton( const SPaneInfo &inPaneInfo,  
MessageT inValueMessage,  
SInt32 inValue,  
ResIDT inTextTraitsID,  
ControlHandle inMacControlH );  
LStdRadioButton(const LStdRadioButton  
&inOriginal);  
LStdRadioButton( LStream *inStream );
```

Parameters These constructors have the following parameters:

const SPaneInfo&	inPaneInfo	A reference to the SPaneInfo object that is the super view.
MessageT	inValueMessage	The message sent when the control is interacted with.
ResIDT	inMENUId	The resource ID of the menu items.
SInt16	inValue	The value to set for the control, must be 1 or 0. One means the control is on, zero means off.
ResIDT	inTextTraitsID	The resource ID of the resource containing the traits to apply to text strings.
Str255	inTitle	The text string to display next to the radio button.
ControlHandle	inMacControlH	This is the Mac OS control handle for the radio button.
LStream*	inStream	A pointer to a stream object that contains the information to create the LControl object.

HotSpotResult()

Purpose Respond to a click in a the radio button. This method is an override of [HotSpotResult\(\)](#) in [LStdControl](#). It calls [SetValue\(\)](#) with the `Button_On` value.

In the Mac interface, clicking on a radio button always turns it on (or leaves it on). The standard way to turn off a radio button is to turn on another one in the same radio group. A radio group will normally be a Listener of a radio button.

SetValue()

Purpose Set the value of the radio button. This method is an override of [SetValue\(\)](#) in [LStdControl](#). It first calls the base class method, then broadcasts a message so that other buttons in the radio group (if present) can be turned off.

LStr255

Overview	LStr255 is a PowerPlant class that is used for Pascal-style strings, having a maximum of 255 characters.														
Methods	<p>The methods in this class are:</p> <table><tr><td>LStr255()</td><td>LStr255(const LStr255&)</td></tr><tr><td>LStr255(const LString&)</td><td>LStr255(const unsigned char*)</td></tr><tr><td>LStr255(unsigned char)</td><td>LStr255(char)</td></tr><tr><td>LStr255(const char*)</td><td>LStr255(const void*, unsigned char)</td></tr><tr><td>LStr255(char**)</td><td>LStr255(short, short)</td></tr><tr><td>LStr255(long)</td><td>LStr255(long double, signed char, short)</td></tr><tr><td>LStr255(unsigned long)</td><td>operator=()</td></tr></table>	LStr255()	LStr255(const LStr255&)	LStr255(const LString&)	LStr255(const unsigned char*)	LStr255(unsigned char)	LStr255(char)	LStr255(const char*)	LStr255(const void*, unsigned char)	LStr255(char**)	LStr255(short, short)	LStr255(long)	LStr255(long double, signed char, short)	LStr255(unsigned long)	operator=()
LStr255()	LStr255(const LStr255&)														
LStr255(const LString&)	LStr255(const unsigned char*)														
LStr255(unsigned char)	LStr255(char)														
LStr255(const char*)	LStr255(const void*, unsigned char)														
LStr255(char**)	LStr255(short, short)														
LStr255(long)	LStr255(long double, signed char, short)														
LStr255(unsigned long)	operator=()														
Data Members	<p>The data members in this class are:</p> <p>mString</p>														
Operation	Use these constructors to create and convert strings into 255-character Pascal strings.														
Source files	<p>(Support Classes)</p> <p>LString.h</p> <p>LString.cp</p>														
Ancestors	LString														

LStr255()

Purpose	This is the default constructor.
Access	Public
Prototype	<code>LStr255 () ;</code>

LStr255

Parameters	None
------------	------

LStr255(const LStr255&)

Purpose	.Copy constructor.		
Access	Public		
Prototype	<code>LStr255(const LStr255& inOriginal);</code>		
Parameters	This constructor takes the following parameter:		
	<code>const LStr255&</code>	<code>inOriginal</code>	The string to copy.

LStr255(const LString&)

Purpose	Copy constructor		
Access	Public		
Prototype	<code>LStr255(const LString& inOriginal);</code>		
Parameters	This constructor takes the following parameter:		
	<code>const LString&</code>	<code>inOriginal</code>	The string to copy.

LStr255(const unsigned char*)

Purpose	Constructor to convert from a pointer to a string.		
Access	Public		
Prototype	<code>LStr255(ConstStringPtr inStringPtr);</code>		
Parameters	This constructor takes the following parameter:		
	<code>ConstStringPtr</code>	<code>inStringPtr</code>	The pointer to convert.

LStr255(unsigned char)

Purpose	Constructor to make a string from a single character.		
Access	Public		
Prototype	<code>LStr255(UInt8 inChar);</code>		
Parameters	This constructor takes the following parameter:		
	<code>UInt8</code>	<code>inChar</code>	The character to convert.

LStr255(char)

Purpose	Constructor to make a string from a single character.		
Access	Public		
Prototype	<code>LStr255(char inChar);</code>		
Parameters	This constructor takes the following parameter:		
	<code>char</code>	<code>inChar</code>	The character to convert.

LStr255(const char*)

Purpose	Constructor to make a string from a C string (null terminated).		
Access	Public		
Prototype	<code>LStr255(const char* inCString);</code>		
Parameters	This constructor takes the following parameter:		
	<code>const char*</code>	<code>inCString</code>	The string to convert.

LStr255(const void*,unsigned char)

Purpose	Constructor to make a string from a pointer and length.		
Access	Public		
Prototype	<code>LStr255(const void* inPtr, UInt8 inLength);</code>		
Parameters	This constructor takes the following parameters:		
	<code>const void*</code>	<code>inPtr</code>	The pointer to convert.
	<code>UInt8</code>	<code>inLength</code>	The length of the string data in the pointer.

LStr255(char)**

Purpose	Constructor to make a string from the data in a handle.		
Access	Public		
Prototype	<code>LStr255(Handle inHandle);</code>		
Parameters	This constructor takes the following parameter:		
	<code>Handle</code>	<code>inHandle</code>	The handle to convert.

LStr255(short,short)

Purpose	Constructor to make a string from a resource.		
Access	Public		
Prototype	<code>LStr255(ResIDT inResID, SInt16 inIndex);</code>		
Parameters	This constructor takes the following parameters:		

ResIDT	inResID	The resource ID containing the string.
SInt16	inIndex	The index into the string resource ID.

LStr255(long)

Purpose	Constructor to make a string from a long integer.	
Access	Public	
Prototype	<code>LStr255(SInt32 inNumber);</code>	
Parameters	This constructor takes the following parameter:	
	<code>SInt32 inNumber</code>	The long integer to create a string from.

LStr255(long double,signed char,short)

Purpose	Assignment to create a string from floating point number.	
Access	Public	
Prototype	<code>LStr255(double_t inNumber, SInt8 inStyle, SInt16 inDigits);</code>	
Parameters	This constructor takes the following parameters:	
	<code>double_t inNumber</code>	The double to create a string from.

SInt8	inStyle	The style to use, must be FLOATDECIMAL or FIXEDDECIMAL.
SInt16	inDigits	<ul style="list-style-type: none">• For FLOATDECIMAL, inDigits is the number of significant digits (should be > 0).• For FIXEDDECIMAL, inDigits is the number of digits to the right of the decimal point.

LStr255(unsigned long)

Purpose	Constructor to create a string from a four character code, which is an unsigned long, the same as ResType and OType .	
Access	Public	
Prototype	LStr255(FourCharCode inCode);	
Parameters	This constructor takes the following parameter:	
	FourCharCod e	The 4-character code to put in the string.

operator=()

Purpose	The assignment operator performs typecasting on strings.	
Access	Public	
Prototype	LStr255& operator=(FourCharCode inCode); LStr255& operator=(SInt32 inNumber); LStr255& operator=(const char* inCString); LStr255& operator=(char inChar); LStr255& operator=(UInt8 inChar); LStr255& operator=(ConstStringPtr inStringPtr); LStr255& operator=(const LString& inString);	

Parameters	This operator takes the following parameters:		
	FourCharCode	inCode	The 4-character code to put in the string.
	SInt32	inNumber	The number to put in the string.
	const char*	inCString	The “C” string to convert.
	char	inChar	The character to make a string from.
	UInt8	inChar	The character to make a string from.
	ConstStringPtr	inStringPtr	The string pointer to make a string from.
	const LString&	inString	The LString to make a string from.
Return	A reference to an LStr255.		

mString

Purpose	String storage.
Access	Private
Prototype	Str255 mString;

LStream

Overview	LStream is a PowerPlant class that is used for managing data streaming in PowerPlant. By abstracting low-level data sourcing and sinking away from objects, you can read and write data without regard to where it came from or is going to.	
Methods	The methods in this class are:	
	LStream()	~LStream()
	AtEnd()	GetBytes()
	GetLength()	GetMarker()
	PeekData()	PutBytes()
	ReadBlock()	ReadCString()
	ReadData()	ReadHandle()
	ReadPString()	ReadPtr()
	SetLength()	SetMarker()
	WriteBlock()	WriteCString()
	WriteData()	WriteHandle()
	WritePString()	WritePtr()
	LStream& operator << (TypeParameter)	LStream& operator >> (TypeParameter)
	LStream& operator =	
Data Members	The data members in this class are:	
	mMarker	mLength
Operation	<i>The PowerPlant Book</i> contains a detailed description of how to work with this class. Refer there for more information.	
Source files	(File & Stream Classes)	
	LStream.h	
	LStream.cp	

LStream

LStream()

Purpose	The constructor creates an object. There are two constructors for LStream. The first is the default constructor. The second is the copy constructor
Access	Public
Prototype	<code>LStream();</code> <code>LStream::LStream(const LStream& inOriginal)</code>
Parameters	None

~LStream()

Purpose	The destructor destroys the LStream object.
Access	Virtual, Public
Prototype	<code>virtual ~LStream();</code>

AtEnd()

Purpose	This method tells if the end of the stream has been reached.
Access	Inline, Public
Prototype	<code>Boolean AtEnd() const;</code>
Parameters	None
Return	Returns true if at the end of the stream.

GetBytes()

Purpose	Read bytes from a Stream to a buffer. Subclasses must override this function to support reading.	
Access	Virtual, Public	
Prototype	<code>ExceptionCode GetBytes(void* outBuffer, SInt32& ioByteCount)</code>	
Parameters	<code>void*</code>	<code>outBuffer</code> Pointer to the stream buffer
	<code>SInt32&</code>	<code>ioByteCount</code> Number of bytes read
Return	Returns an error code and passes back the number of bytes actually read, which may be less than the number requested if an error occurred.	
Remarks	You should not throw an Exception out of this function.	

GetLength()

Purpose	Return the length, in bytes, of the Stream
Access	Public, Virtual
Prototype	<code>SInt32 GetLength() const;</code>
Parameters	None
Return	Size of streams in bytes.

GetMarker()

Purpose	Return the Read/Write Marker position
Access	Public, Virtual

LStream

Prototype	<code>SInt32 GetMarker() const;</code>
Parameters	None
Return	Marker position
Remarks	Position is a byte offset from the start of the Stream.

PeekData()

Purpose	Read data from a Stream to a buffer, without moving the Marker		
Access	Public		
Prototype	<code>SInt32 PeekData(void* outBuffer, SInt32 inByteCount)</code>		
Parameters	<code>void*</code>	<code>outBuffer</code>	Pointer to the stream buffer
	<code>SInt32</code>	<code>inByteCount</code>	Number of bytes read
Return	The number of bytes actually read, which may be less than the number requested if an error occurred.		

PutBytes()

Purpose	Write bytes from a buffer to a Stream		
	Subclasses must override this function to support writing.		
Access	Public, Virtual		
Prototype	<code>ExceptionCode PutBytes(const void* inBuffer, SInt32 &ioByteCount)</code>		

Parameters

<code>const void*</code>	<code>inBuffer</code>	Pointer to the stream buffer
<code>SInt32</code>	<code>&ioByteCount</code>	Number of bytes written

Return Returns an error code and passes back the number of bytes actually written, which may be less than the number requested if an error occurred.

Remarks You should not throw an Exception out of this function.

ReadBlock()

Purpose Read data from a Stream to a buffer.

Access Public

Prototype

```
void ReadBlock(
    void *outBuffer,
    SInt32 inByteCount);
```

Parameters

<code>void</code>	<code>*inBuffer</code>	Pointer to the stream buffer
<code>SInt32</code>	<code>inByteCount</code>	Number of bytes to read

Return None

ReadCString()

Purpose Read a C string from a Stream

Access Public

Prototype `SInt32 ReadCString(char *outString)`

Parameters

<code>char</code>	<code>*outString</code>	C string
-------------------	-------------------------	----------

Return Returns the number of bytes read.

LStream

Remarks	C string is stored as a 4-byte count followed by the characters. The null terminator is not stored and must be added afterwards.
---------	--

ReadData()

Purpose	Read bytes from a Stream to a buffer.		
Access	Public, Virtual		
Prototype	<pre>SInt32 ReadData(void *outBuffer, SInt32 inByteCount);</pre>		
Parameters	void	*outBuffer	Pointer to the stream buffer
	SInt32	inByteCount	Number of bytes to read
Return	Number of bytes actually read		

ReadHandle()

Purpose	Read data from a Stream into a newly created Handle block		
Access	Public		
Prototype	<pre>SInt32 ReadHandle(Handle &outHandle);</pre>		
Parameters	Handle	&outHandle	Address of Handle block
Return	Returns the number of bytes read.		
Remarks	A Handle block is stored in a Stream as a 4-byte count (size of the Handle), followed by the contents of the Handle block.		

ReadPString()

Purpose	Read a Pascal string from a Stream		
Access	Public		
Prototype	<code>SInt32 ReadPString(Str255 outString);</code>		
Parameters	Str255	outString	Pascal string to read
Return	Returns the number of bytes read.		

ReadPtr()

Purpose	Read data from a Stream into a newly created Ptr block		
Access	Public		
Prototype	<code>SInt32 ReadPtr(Ptr &outPtr);</code>		
Parameters	Ptr	&outPtr	Address of point block
Return	Returns the number of bytes read.		
Remarks	A Ptr block is stored in a Stream as a 4-byte count (size of the Ptr), followed by the contents of the Ptr block.		

SetLength()

Purpose	Set the length, in bytes, of the Stream.
Access	Public, Virtual
Prototype	<code>void SetLength(</code>

LStream

	<code>SInt32 inLength;</code>		
Parameters	<code>*</code>		
	<code>SInt32</code>	<code>inLength</code>	New stream length
Return	None		

SetMarker()

Purpose	Place the Read/Write Marker at an offset from a specified position. <code>inFromWhere</code> can be <code>streamFrom_Start</code> , <code>streamFrom_End</code> , or <code>streamFrom_Marker</code>		
Access	Public, Virtual		
Prototype	<code>void SetMarker(SInt32 inOffset, EStreamFrom inFromWhere);</code>		
Parameters	<code>SInt32</code>	<code>inOffset</code>	Offset position in bytes
	<code>EStreamFrom</code>	<code>inFromWhere</code>	Marker position in Stream
Return	None		

WriteBlock()

Purpose	Write data, specified by a pointer and byte count, to a Stream		
Access	Public		
Prototype	<code>void WriteBlock(const void *inBuffer, SInt32 inByteCount);</code>		
Parameters	<code>const void</code>	<code>*inBuffer</code>	Pointer to a stream buffer
	<code>SInt32</code>	<code>inByteCount</code>	Number of bytes to write

Return None

WriteCString()

Purpose Write a C string to a Stream

Access Public

Prototype `SInt32 WriteCString(
 const char *inString);`

Parameters

 const char *inString C string

Return Returns the number of bytes written.

Remarks C string is written as a 4-byte count followed by the characters. The terminator is not written.

WriteData()

Purpose

Access Public

Prototype `SInt32 WriteData(
 const void *inBuffer,
 SInt32 inByteCount);`

Parameters

 const void *inBuffer Pointer to a stream buffer
 SInt32 inByteCount Number of bytes to write

Return Number of bytes actually written.

Remarks Calls `PutBytes()`.

WriteHandle()

Purpose	Write a Toolbox Handle block to a Stream			
Access	Public			
Prototype	<pre>SInt32 WriteHandle(Handle inHandle);</pre>			
Parameters	<table><tr><td>Handle</td><td>inHandle</td><td>Handle block to write</td></tr></table>	Handle	inHandle	Handle block to write
Handle	inHandle	Handle block to write		
Return	Returns the number of bytes written.			
Remarks	A Handle block is written as a 4-byte count (size of the Handle), followed by the contents of the Handle block.			

WritePString()

Purpose	Write a Pascal string to a Stream.		
Access	Public		
Prototype	<pre>SInt32 WritePString(ConstStringPtr inString);</pre>		
Parameters	ConstStringPtr	inString	Pascal string to write
Return	Returns the number of bytes written.		

WritePtr()

Purpose	Write a Toolbox Ptr block to a Stream.
Access	Public
Prototype	<pre>SInt32 WritePtr(Ptr inPtr);</pre>

```
Ptr inPtr);
```

Parameters

Ptr	inPtr	Pointer block to write
-----	-------	------------------------

Return	Returns the number of bytes written.
--------	--------------------------------------

Remarks	A Ptr block is written as a 4-byte count (size of the Ptr), followed by the contents of the Ptr block.
---------	--

LStream& operator << (*TypeParameter*)

<i>Type Parameter</i>	double, float, unsigned long, long, unsigned short, short, char, unsigned char, signed char, char **, const Point&, const Rect&, const char*, const unsigned char*, double&, float&, unsigned long&, long&, unsigned short&, char&, unsigned char&, signed char&, char**&, Point&, Rect&, char*, unsigned char*
-----------------------	---

Purpose	Public
---------	--------

Access	Public
--------	--------

Prototype	LStream& operator << (<i>TypeParameter</i> inNum);
-----------	---

Parameters	Variable
------------	----------

Return	None
--------	------

LStream& operator >> (*TypeParameter*)

<i>Type Parameter</i>	double, float, unsigned long, long, unsigned short, short, char, unsigned char, signed char, char **, const Point&, const Rect&, const char*, const unsigned char*, double&, float&, unsigned long&, long&, unsigned short&, char&, unsigned
-----------------------	--

LStream

	<code>char&, signed char&, char**&, Point&, Rect&, char*, unsigned char*</code>
Purpose	Public
Access	Public
Prototype	<code>LStream& operator >> (TypeParameter inNum);</code>
Parameters	Variable
Return	None

LStream& operator =

Purpose	Assignment constructor.			
Access	Public			
Prototype	<code>LStream& operator = (const LStream& inOriginal);</code>			
Parameters	<table><tr><td><code>const LStream&</code></td><td><code>inOriginal</code></td><td>Address of original LStream object</td></tr></table>	<code>const LStream&</code>	<code>inOriginal</code>	Address of original LStream object
<code>const LStream&</code>	<code>inOriginal</code>	Address of original LStream object		
Return	Returns address of LStream object.			
Remarks	<code>operator =</code> does not create a duplicate of the stream, but assigns a stream the same memory address as the original. If you delete one, you loose both. You have been warned.			

mMarker

Purpose	Marker position offset in bytes.
Access	Protected
Prototype	<code>SInt32 mMarker;</code>

mLength

Purpose	Length of stream in bytes.
Access	Protected
Prototype	SInt32 mLength;

LString

Overview LString is a PowerPlant class that is used for manipulating text strings.

Methods The methods in this class are:

<u>LString()</u>	<u>Append (TypeParameters)</u>
<u>AppendPStr()</u>	<u>Assign(TypeParameters)</u>
<u>BeginsWith(TypeParameters)</u>	<u>CStringLength()</u>
<u>CompareBytes()</u>	<u>CompareIgnoringCase()</u>
<u>CompareTo()</u>	<u>CopyPStr()</u>
<u>EndsWith()</u>	<u>Find()</u>
<u>FindWithin()</u>	<u>FourCharCodeToPStr()</u>
<u>Insert()</u>	<u>Length()</u>
<u>LoadFromPtr()</u>	<u>LoadFromSTRListResource()</u>
<u>LoadFromStringPtr()</u>	<u>PStrToFourCharCode()</u>
<u>Remove()</u>	<u>Replace()</u>
<u>ReverseFind()</u>	<u>SetCompareFunc()</u>
<u>ToolboxCompareText()</u>	<u>operator +=</u>
<u>operator = (TypeParameter)</u>	<u>(TypeParameter)</u>
<u>operator [](unsigned char) const</u>	<u>operator ()(unsigned</u>
<u>operator const unsigned char*()</u>	<u>char,unsigned char) const</u>
<u>const</u>	<u>operator [](unsigned char)</u>
<u>operator long() const</u>	<u>operator long double() const</u>
<u>operator unsigned long() const</u>	<u>operator unsigned char*()</u>

Data Members The data members in this class are:

LString

[mStringPtr](#)

[mCompareFunc](#)

[mMaxBytes](#)

Operation	This class implements string functionality for Pascal-style strings. It serves as a base class for LStr255 and TString.
Source files	(Support Classes) LString.h LString.cp

LString()

Purpose	This is a constructor to create a string from a max length and a string pointer. Subclasses must call this "protected" constructor to specify the maximum string size (including the length byte) and a pointer to the storage for the string
Access	Public
Prototype	<code>LString();</code>
Parameters	None

Append (*TypeParameters*)

Purpose	These overloaded methods append characters to the end of this String.
Access	Public
Prototype	<code>LString& Append(const void* inPtr, UInt8 inLength);</code> <code>LString& Append(SInt32 inNumber);</code> <code>LString& Append(char inChar);</code> <code>LString& Append(UInt8 inChar);</code> <code>LString& Append(ConstStringPtr inStringPtr);</code>

`LString& Append(const LString& inString);`

Parameters	The parameters for these methods are:		
	<code>const void*</code>	<code>inPtr</code>	A pointer to the string to append.
	<code>SInt32</code>	<code>inNumber</code>	The number to append to the string.
	<code>char</code>	<code>inChar</code>	The character to append to the string.
	<code>UInt8</code>	<code>inChar</code>	The character to append to the string.
	<code>ConstStringPtr</code>	<code>inStringPtr</code>	The string to append to the string.
	<code>const LString&</code>	<code>inString</code>	The LString to append to the string.
	<code>UInt8</code>	<code>inLength</code>	The length of the string to append.
Return	A reference to an LString that has the appended string.		

AppendPStr()

Purpose	Append a Pascal string to this String.		
Access	Static, Public		
Prototype	<code>static StringPtr AppendPStr(Str255 ioBaseString, ConstStringPtr inAppendString, SInt16 inDestSize);</code>		
Parameters	The parameters for these methods are:		
	<code>Str255</code>	<code>ioBaseString</code>	A pointer to the string to append to.
	<code>ConstStringPtr</code>	<code>inAppendString</code>	The string to append.
	<code>SInt16</code>	<code>inDestSize</code>	The size of the buffer.
Return	A pointer to the finished String.		

Assign(*TypeParameters*)

Purpose Assign a value to the string.

Access Public

Prototype `LString& Assign(FourCharCode inCode);`
`LString& Assign(double_t inNumber,`
`SInt8 inStyle,SInt16 inDigits);`
`LString& Assign(SInt32 inNumber);`
`LString& Assign(ResIDT inResID,SInt16 inIndex);`
`LString& Assign(Handle inHandle);`
`LString& Assign(const void* inPtr,UInt8 inLength);`
`LString& Assign(const char* inCString);`
`LString& Assign(char inChar);`
`LString& Assign(UInt8 inChar);`
`LString& Assign(ConstStringPtr inStringPtr);`
`LString& Assign(const LString& inString,`
`UInt8 inStartPos,UInt8 inCount);`

Parameters The parameters for this method are:

FourCharCode	inCode	A four characters in a FourCharCode to assign.
double_t	inNumber	The number to assign to the string.
SInt8	inStyle	The style to use, must be FLOATDECIMAL or FIXEDDECIMAL.
SInt16	inDigits	<ul style="list-style-type: none">• For FLOATDECIMAL, inDigits is the number of significant digits (should be > 0).• For FIXEDDECIMAL, inDigits is the number of digits to the right of the decimal point.
SInt32	inNumber	The number to assign to the string.
ResIDT	inResID	The LString to append to the string.
SInt16	inIndex	The length of the string to append.
Handle	inHandle	The handle to the string to assign.
const void*	inPtr	The pointer to the string to assign.
UInt8	inLength	The length of the string.
const char*	inCString	The pointer to the "C" string to assign.

char	inChar	The character to assign to the string.
UInt8	inChar	The character to assign to the string.
ConstStringPtr	inStringPtr	The pointer to the string to assign.
const LString&	inString	The pointer to the string to assign.
UInt8	inStartPos	The starting index in the string to assign from.
UInt8	inCount	The character count to put in the string.

Return A reference to the finished LString.

BeginsWith(*TypeParameters*)

Purpose Indicate whether this String begins with a specified String.

Access Public

Prototype `Boolean BeginsWith(const void* inPtr,
 UInt8 inLength) const;
 Boolean BeginsWith(UInt8 inChar) const;
 Boolean BeginsWith(
 ConstStringPtr inStringPtr) const;
 Boolean BeginsWith(const LString& inString)
 const;`

Parameters The parameters for this method are:

UInt8	inLength	The length of the string.
const void*	inPtr	The pointer to the string to check for.
UInt8	inChar	The character to check for at the beginning of the string.

	<code>ConstStringPtr</code>	<code>inStringPtr</code>	The pointer to the string to check for.
	<code>const LString&</code>	<code>inString</code>	The pointer to the string to check for.
Return	A Boolean value of <code>true</code> if the String starts with the passed String, else <code>false</code> .		

CStringLength()

Purpose	Return the length of a C string, but with an upper limit of 255.		
Access	Static, Public		
Prototype	<code>static UInt8 CStringLength(const char* inCString);</code>		
Parameters	The parameters for this method are:		
	<code>const char*</code>	<code>inCString</code>	The pointer to the "C" string to assign.
Return	The length of the string (number of characters in it).		

CompareBytes()

Purpose	Simple byte-by-byte value comparison of two strings.		
Access	Static, Public		
Prototype	<code>static SInt16 CompareBytes(const void* inLeft, const void* inRight, UInt8 inLeftLength, UInt8 inRightLength);</code>		
Parameters	The parameters for this method are:		
	<code>const void*</code>	<code>inLeft</code>	The first string.
	<code>const void*</code>	<code>inRight</code>	The second string.

	UInt8	inLeftLength	The length of the first string.
	UInt8	inRightLength	The length of the second string.
Return	Return 1 if the Left String is greater, else return -1 if the Right String is greater. Equality will cause a return value of 0.		

CompareIgnoringCase()

Purpose	Case-insensitive string comparison.		
Access	Static, Public		
Prototype	<pre>static SInt16 CompareIgnoringCase(const void* inLeft, const void* inRight, UInt8 inLeftLength, UInt8 inRightLength);</pre>		
Parameters	The parameters for this method are:		
	const void*	inLeft	The first string.
	const void*	inRight	The second string.
	UInt8	inLeftLength	The length of the first string.
	UInt8	inRightLength	The length of the second string.
Return	Return 1 if the Left String is greater, else return -1 if the Right String is greater. Equality will cause a return value of 0.		

CompareTo()

Purpose	Compare two strings using a compare function set in mCompareFunc .		
Access	Protected		
Prototype	<pre>SInt16 CompareTo (const void* inPtr, UInt8 inLength) const;</pre>		

```
SInt16 CompareTo( UInt8 inChar) const;  
SInt16 CompareTo( ConstStringPtr inStringPtr)  
const;  
SInt16 CompareTo( const LString& inString) const;
```

Parameters	The parameters for this method are:		
	const void*	inPtr	The string to compare with.
	UInt8	inLength	The length of the string.
	UInt8	inChar	The character to compare with.
	ConstStringPtr	inStringPtr	The pointer to a string to compare with.
	const LString&	inString	A reference to a string to compare with.
Return	The return values are specified by the compare function you supply.		

CopyPStr()

Purpose	Copy a Pascal String into the String.		
Access	Public		
Prototype	static StringPtr CopyPStr(ConstStringPtr inSourceString, StringPtr outDestString, SInt16 inDestSize);		
Parameters	The parameters for this method are:		
	SInt16	inDestSize	The length of the source buffer.
	StringPtr	outDestString	The destination string.
	ConstStringPtr	inStringPtr	The pointer to a string to copy.
Return	A pointer to the String is returned.		

EndsWith()

Purpose	Return whether this String ends with a specified String.		
Access	Public		
Prototype	<pre>Boolean EndsWith(const void* inPtr, UInt8 inLength) const; Boolean EndsWith(UInt8 inChar) const; Boolean EndsWith(ConstStringPtr inStringPtr) const; Boolean EndsWith(const LString& inString) const;</pre>		
Parameters	The parameters for this method are:		
	UInt8	inLength	The length of the string.
	const void*	inPtr	The pointer to the string to check for.
	UInt8	inChar	The character to check for at the beginning of the string.
	ConstStringPtr	inStringPtr	The pointer to the string to check for.
	const LString&	inString	The pointer to the string to check for.
Return	A Boolean value of true if the String starts with the passed String, else false.		

Find()

Purpose	Find a substring within a String.		
Access	Public		
Prototype	<pre>UInt8 Find(const void* inPtr, UInt8 inLength, UInt8 inStartPos) const; UInt8 Find(UInt8 inChar, UInt8 inStartPos = 1) const; UInt8 Find(ConstStringPtr inStringPtr, UInt8 inStartPos = 1) const;</pre>		

```
UInt8 Find( const LString& inString, UInt8
inStartPos = 1 ) const;
```

Parameters The parameters for this method are:

UInt8	inLength	The length of the string.
const void*	inPtr	The pointer to the string to check for.
UInt8	inStartPos	The starting position in the string to search from.
ConstStringPtr	inStringPtr	The pointer to the string to check for.

Return Return the index of where the substring starts within this String.

FindWithin()

Purpose Return the index of where this String starts within another String.

Access Public

Prototype

```
UInt32 FindWithin( const void* inPtr,
UInt32 inLength, UInt32 inStartPos ) const;
UInt8 FindWithin(UInt8 inChar,
UInt8 inStartPos = 1 ) const;
UInt8 FindWithin( ConstStringPtr inStringPtr,
UInt8
inStartPos = 1 ) const;
UInt8 FindWithin( const LString& inString, UInt8
inStartPos = 1 ) const;
```

Parameters The parameters for this method are:

UInt8	inLength	The length of the string.
const void*	inPtr	The pointer to the string to check for.
UInt8	inStartPos	The starting position in the string to search from.
ConstStringPtr	inStringPtr	The pointer to the string to check for.

Return	Returns 0 if the substring is not in the String. <code>inStartPos</code> is the index within <code>inPtr</code> at which to begin searching. The default value is 1. Value 0 is not valid.
Remarks	To find the next occurrence, set <code>inStartPos</code> to one plus the index returned by the last <code>FindWithin</code> call.

FourCharCodeToPStr()

Purpose	Convert a four character code to a Pascal string and return a pointer to the string.		
Access	Static, Public		
Prototype	<pre>static StringPtr FourCharCodeToPStr(FourCharCode inCode, StringPtr outString);</pre>		
Parameters	The parameters for this method are:		
	FourCharCode	inCode	The four-character value to put in the string.
	StringPtr	outString	The pointer to the string.
Return	Returns a pointer to the string.		
Remarks	<code>outString</code> must point to a buffer (usually a string) that can hold the resulting string, including the length byte. Therefore, the buffer must be at least 5-bytes large. Usually, you will allocate an array of 5 unsigned char.		

Insert()

Purpose			
Access	Public		
Prototype	<pre>LString& Insert(const void* inPtr, UInt8 inLength, UInt8 inAtIndex); LString& Insert(UInt8 inChar, UInt8 inAtIndex); LString& Insert(ConstStringPtr inStringPtr, UInt8 inAtIndex); LString& Insert(const LString& inString, UInt8 inAtIndex);</pre>		

LString

Length()

Parameters	The parameters for this method are:		
	UInt8	inLength	The length of the string.
	const void*	inPtr	The pointer to the string to insert.
	UInt8	inAtIndex	The starting position in the string to insert from.
	UInt8	inChar	The character to insert in the string.
	ConstStringPtr	inStringPtr	The pointer to the string to insert.
	const LString&	inString	A reference to the string to insert.
Return	A reference to the finished string.		

Length()

Purpose	Indicate the String length.
Access	Public
Prototype	UInt8 Length() const;
Parameters	None
Return	Returns the number of characters in the String.

LoadFromPtr()

Purpose	Fill in the String given a pointer and a byte count.		
Access	Protected		
Prototype	void LoadFromPtr(const void* inPtr, UInt8 inLength);		
Parameters	The parameters for this method are:		
	UInt8	inLength	The length of the string.
	const void*	inPtr	The pointer to the string to insert.

LString

Remove()

Parameters	The parameters for this method are:		
	ConstStringPtr	inString	The pointer to the string to use.
	FourCharCode&	outCode	The four-character code extracted from the string.
Return	None		

Remove()

Purpose	Remove a given number of bytes from this String starting at the specified index.		
Access	Public		
Prototype	<pre>LString& Remove(UInt8 inStartPos, UInt8 inCount);</pre>		
Parameters	The parameters for this method are:		
	UInt8	inStartPos	The starting index in the string.
	UInt8	inCount	The character count to remove.
Return	A reference to the String.		

Replace()

Purpose	Replace a portion of this String with a new value.		
Access	Public		
Prototype	<pre>LString& Replace(UInt8 inStartPos, UInt8 inCount, const void* inPtr, UInt8 inLength); LString& Replace(UInt8 inStartPos, UInt8 inCount, UInt8 inChar); LString& Replace(UInt8 inStartPos, UInt8 inCount, ConstStringPtr inStringPtr); LString& Replace(UInt8 inStartPos, UInt8 inCount, const LString& inString);</pre>		
Parameters	The parameters for this method are:		

	UInt8	inStartPos	The starting index in the string.
	UInt8	inCount	The character count to remove.
	const void*	inPtr	The pointer to the string.
	UInt8	inLength	The length of the string.
	const LString&	inString	The reference to the string.
	ConstStringPtr	inStringPtr	The pointer to the string.
	UInt8	inChar	The character to replace.
Return	A reference to the String.		

ReverseFind()

Purpose Find where a substring starts within this String, searching from the end of the String towards the beginning of the String.

Access Public

Prototype

```

UInt8 ReverseFind( const void* inPtr, UInt8
inLength, UInt8 inStartPos ) const;
UInt8 ReverseFind( UInt8 inChar, UInt8 inStartPos
= 255 ) const;
UInt8 ReverseFind( ConstStringPtr inStringPtr,
UInt8 inStartPos = 255 ) const;
UInt8 ReverseFind( const LString& inString, UInt8
inStartPos = 255 ) const;
UInt32 ReverseFindWithin( const void* inPtr,
UInt32 inLength, UInt32 inStartPos ) const;
UInt8 ReverseFindWithin( UInt8 inChar, UInt8
inStartPos = 1 ) const;
UInt8 ReverseFindWithin( ConstStringPtr
inStringPtr, UInt8 inStartPos = 255 ) const;
UInt8 ReverseFindWithin( const LString& inString,
UInt8 inStartPos = 255 ) const;

```

Parameters The parameters for this method are:

LString

SetCompareFunc()

	UInt8	inStartPos	The starting index in the string.
	const void*	inPtr	The pointer to the string.
	UInt8	inLength	The length of the string.
	const LString&	inString	The reference to the string.
	ConstStringPtr	inStringPtr	The pointer to the string.
	UInt8	inChar	The character to find.
Return	The index where the item was found.		

SetCompareFunc()

Purpose	Sets the value of mCompareFunc to point to a function used for comparing strings.		
Access	Public		
Prototype	void SetCompareFunc(CompareFunc inCompareFunc);		
Parameters	The parameters for this method are:		
	CompareFunc	inCompareFunc	The pointer to the compare function.
Return	None		

ToolboxCompareText()

Purpose	This method is a String comparison function that uses the Toolbox CompareText() routine.		
Access	Static, Public		
Prototype	static SInt16 ToolboxCompareText(const void* inLeft, const void* inRight, UInt8 inLeftLength, UInt8 inRightLength);		
Parameters	The parameters for this method are:		

	const void*	inLeft	The first string.
	const void*	inRight	The second string.
	UInt8	inLeftLength	The length of the first string.
	UInt8	inRightLength	The length of the second string.
Return	This is the same as the return value for CompareText () in the Mac OS Toolbox.		

operator += (TypeParameter)

Purpose	This is an operator overload for the += operator. It calls the Append (TypeParameters) method to append a String to another String.
TypeParameter	long, char, unsigned char, const unsigned char*, const LString&
Access	Public
Prototype	LString& operator+= (TypeParameter) ;
Parameters	The specified TypeParameters may be input to this operator.
Return	A reference to the String.

operator = (TypeParameter)

Type Parameter	unsigned long, long, const char*, char, unsigned char, const unsigned char*, const LString&
Purpose	This is an operator overload for the = (assignment) operator. It calls the Assign (TypeParameters) method to assign a String to another String.
TypeParameter	unsigned long, long, const char*, char, unsigned char, const unsigned char*, const LString&
Access	Public
Prototype	LString& operator= (TypeParameter) ;
Parameters	The specified TypeParameters may be input to this operator.

LString

operator()(unsigned char,unsigned char) const

Return A reference to the String.

operator()(unsigned char,unsigned char) const

Purpose Return a LStr255 String object containing the specified substring of this String.

Access Public

Prototype LStr255 operator()(
 UInt8 inStartPos,
 UInt8 inCount) const;

Parameters The parameters for this operator are:

 UInt8 inStartPos The start position within the string.

 UInt8 inCount The length of the string to return.

Return An LStr255

operator [](unsigned char) const

Purpose Return a character at a specified position in the String.

Access Public

Prototype const UInt8& operator[](UInt8 inPosition) const;

Parameters The parameter for this operator is:

 UInt8 inPosition The start position within the string.

Return A reference to the unsigned character.

operator [](unsigned char)

Purpose Access individual characters in a String, in an array-like fashion.

Access Inline, Public

Prototype UInt8& operator[](UInt8 inPosition);

Parameters The parameter for this operator is:

 UInt8 inPosition The start position within the string.

Return A reference to the unsigned character.

operator const unsigned char*() const

Purpose	Retrieve the value of mStringPtr .
Access	Inline, Public
Prototype	<code>operator ConstStringPtr() const;</code>
Parameters	None
Return	A pointer to the String held in mStringPtr .

operator long double() const

Purpose	This operator provides a conversion to a floating point number.
Access	Public
Prototype	<code>operator double_t() const;</code>
Parameters	None
Return	A <code>double_t</code> that represents the String.

operator long() const

Purpose	This operator provides a conversion to a long integer.
Access	Public
Prototype	<code>operator SInt32() const;</code>
Parameters	None
Return	A <code>SInt32</code> that represents the String.

operator unsigned char*()

Purpose	This operator provides a conversion to a <code>char*</code> pointer.
Access	Public
Prototype	<code>operator StringPtr();</code>
Parameters	None
Return	A <code>StringPtr</code> that represents the String.

LString

operator unsigned long() const

operator unsigned long() const

Purpose	This operator provides a conversion to a four character code.
Access	Public
Prototype	<code>operator FourCharCode() const;</code>
Parameters	None
Return	A <code>FourCharCode</code> that represents the String.

mStringPtr

Purpose	The String pointer.
Access	Protected
Prototype	<code>StringPtr mStringPtr;</code>

mCompareFunc

Purpose	The compare function pointer for the String.
Access	Protected
Prototype	<code>CompareFunc mCompareFunc;</code>

mMaxBytes

Purpose	The maximum number of bytes for the String.
Access	Protected
Prototype	<code>UInt16 mMaxBytes;</code>

LSubOverlapView

Overview	LSubOverlapView is a PowerPlant class that is a container view for Panes which might overlap.
Methods	The methods in this class are: LSubOverlapView() FocusDraw() ~LSubOverlapView()
Data Members	There are no data member in this class.
Operation	This class inherits from the chain of classes: LPane, LView and LAttachable. This class helps manage panes that overlap.
Source files	(Pane Classes) LSubOverlapView.h LSubOverlapView.cp
Ancestors	LAttachable LPane LView

LSubOverlapView()

Purpose	The constructors create the objects from the passed-in parameters.		
Access	Public		
Prototype	<pre>LSubOverlapView(); LSubOverlapView(LStream *inStream);</pre>		
Parameters	The parameter for these constructors is:		
	<pre>LStream*</pre>	<pre>inStream</pre>	The stream to read from to create the object.

LSubOverlapView

~LSubOverlapView()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LSubOverlapView();</code>

FocusDraw()

Purpose	Set up coordinates system and clipping region. Panes rely on their superview to focus them. When a Pane requests focus for drawing, a SubOverlapView sets the clipping region to revealed portion of that Pane's Frame minus the Frames of all sibling Panes that are in front of that Pane. This is an override of FocusDraw() in LPane .
---------	--

LTabGroup

Overview	LTabGroup is a PowerPlant class that is used for managing tab key navigation between two or more edit fields in a view. Mac OS Human Interface Guidelines suggest that typing the Tab key should advance the text entry cursor from one field to the next.						
Methods	<p>The methods in this class are:</p> <table><tr><td>LTabGroup()</td><td>~LTabGroup()</td></tr><tr><td>BeTarget()</td><td>GetOnDutySub()</td></tr><tr><td>HandleKeyPress()</td><td>RotateTarget()</td></tr></table>	LTabGroup()	~LTabGroup()	BeTarget()	GetOnDutySub()	HandleKeyPress()	RotateTarget()
LTabGroup()	~LTabGroup()						
BeTarget()	GetOnDutySub()						
HandleKeyPress()	RotateTarget()						
Data Members	There are no data members in this class.						
Operation	<p>A Tab Group switches the Target amongst its SubCommanders in response to Tab and Shift-Tab key presses. A SubCommander is responsible for passing up Tab key presses to its SuperCommander if it wants to be usable with a TabGroup.</p> <p>By default, the first SubCommander of a TabGroup will be the Target when the Window containing the TabGroup is activated. If there is more than one TabGroup in a Window, then the last TabGroup will be the one on duty.</p> <p>To make a particular Commander be the Target when its Window is activated, call SetLatentSub() for that LCommander.</p>						
Source files	<p>(Support Classes)</p> <p><code>LTabGroup.h</code></p> <p><code>LTabGroup.cp</code></p>						
Ancestors	<p>LAttachable</p> <p>LCommander</p>						

LTabGroup

LTabGroup()

Purpose	The constructor creates objects from the passed-in parameters.		
Access	Public		
Prototype	<pre>LTabGroup() ; LTabGroup(LStream* inStream); LTabGroup(LCommander *inStream);</pre>		
Parameters	The parameters for these constructors are:		
	<code>LStream*</code>	<code>inStream</code>	This is unused.
	<code>LCommander *</code>	<code>inSuper</code>	Pointer to the SuperCommander.

~LTabGroup()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<pre>virtual ~LTabGroup();</pre>

BeTarget()

Purpose	TabGroup has become the Target. This is an override of BeTarget() in LCommander .
---------	---

GetOnDutySub()

Purpose	Return the on duty SubCommander of a TabGroup.
Access	Virtual, Protected

Prototype	<code>virtual LCommander* GetOnDutySub();</code>
Parameters	None
Return	A pointer to the LCommander on duty.

HandleKeyPress()

Purpose Tab switches the Target to the next item in the TabGroup. Shift-Tab to the previous item.

All other keystrokes (and Tabs with modifiers other than Shift) get passed up. This is an override of [HandleKeyPress\(\)](#) in [LCommander](#).

RotateTarget()

Purpose	Switch Target to another SubCommander, either the one before or after the current one.		
Access	Virtual, Public		
Prototype	<code>virtual void RotateTarget(Boolean inBackward);</code>		
Parameters	The parameters for these constructors are: <table> <tr> <td><code>Boolean inBackward</code></td><td>This indicates whether to rotate backward or forward in the sequence.</td></tr> </table>	<code>Boolean inBackward</code>	This indicates whether to rotate backward or forward in the sequence.
<code>Boolean inBackward</code>	This indicates whether to rotate backward or forward in the sequence.		
Return	None		

LTable

Overview	LTable is a PowerPlant class that is used for display and management of tabular data: rows and columns of rectangular cells in a two-dimensional grid.	
Methods	The methods in this class are:	
	<u>LTable()</u>	<u>~LTable()</u>
	<u>ActivateSelf()</u>	<u>ClickCell()</u>
	<u>ClickSelf()</u>	<u>DeactivateSelf()</u>
	<u>DrawCell()</u>	<u>DrawSelf()</u>
	<u>EqualCell()</u>	<u>FetchCellDataIndex()</u>
	<u>FetchCellHitBy()</u>	<u>FetchLocalCellFrame()</u>
	<u>GetCellData()</u>	<u>GetSelectedCell()</u>
	<u>GetTableSize()</u>	<u>HiliteCell()</u>
	<u>InitTable()</u>	<u>InsertCols()</u>
	<u>InsertRows()</u>	<u>IsValidCell()</u>
	<u>RemoveCols()</u>	<u>RemoveRows()</u>
	<u>SelectCell()</u>	<u>SetCellData()</u>
	<u>SetCellDataSize()</u>	<u>SetColWidth()</u>
	<u>SetRowHeight()</u>	<u>UnhiliteCell()</u>
Data Members	The data members in this class are:	
	<u>mRows</u>	<u>mCols</u>
	<u>mRowHeight</u>	<u>mColWidth</u>
	<u>mCellData</u>	<u>mSelectedCell</u>
Operation	Refer to <i>The PowerPlant Book</i> for a detailed introduction to this class.	
Source files	(Pane Classes)	
	LTable.h	

LTable

LTable.cp

See also [LAttachable](#)

[LPane](#)

[LView](#)

LTable()

Purpose The constructor creates the object.

Access Public

Prototype

```
LTable();  
LTable(  
    const SPaneInfo&inPaneInfo,  
    const SViewInfo&inViewInfo,  
    SInt32inNumberOfRows,  
    SInt32inNumberOfCols,  
    SInt32inRowHeight,  
    SInt32inColWidth,  
    SInt32inCellDataSize );  
LTable( LStream*inStream );
```

~LTable()

Purpose The destructor destroys the object.

Access Virtual, Public

Prototype

```
virtual ~LTable();
```

ActivateSelf()

Purpose This is an override of [ActivateSelf\(\)](#) in [LPane\(\)](#).

ClickCell()

Purpose	Respond to a click in the cell.		
Access	Virtual, Protected		
Prototype	<pre>virtual void ClickCell(const TableCellT&inCell, const SMouseDownEvent& inMouseDown);</pre>		
Parameters	The parameters for this method are:		
	const TableCellT&	inCell	The cell.
	const SMouseDownEvent&	inMouseDown	The mouse down event info.
Return	None		

ClickSelf()

Purpose	This is an override of ClickSelf() in LPane() .
---------	---

DeactivateSelf()

Purpose	This is an override of DeactivateSelf() in LPane() .
---------	--

DrawCell()

Purpose	Draw the contents of a cell.
Access	Virtual, Protected
Prototype	<pre>virtual void DrawCell(const TableCellT&inCell);</pre>

LTable

Parameters	The parameters for this method are:		
	<code>const TableCellT&</code>	<code>inCell</code>	The cell.
Return	None		

DrawSelf()

Purpose	This is an override of DrawSelf() in LPane() .
---------	--

EqualCell()

Purpose	This method tests 2 cells for equality.		
Access	Public		
Prototype	<pre>Boolean EqualCell(const TableCellT&inCellA, const TableCellT&inCellB) const;</pre>		
Parameters	The parameters for this method are:		
	<code>const TableCellT&</code>	<code>inCellA</code>	The first cell.
	<code>const TableCellT&</code>	<code>inCellB</code>	The other cell.
Return	Returns true if equal, else false.		

FetchCellDataIndex()

Purpose	Retrieves the index of a cell containing the data indicated.		
Access	Protected		
Prototype	<pre>SInt32 FetchCellDataIndex(const TableCellT&inCell);</pre>		
Parameters	The parameters for this method are:		

	<code>const TableCellT& inCell</code>	The cell.
Return	Returns the cell index.	

FetchCellHitBy()

Purpose	Find a cell hit by a certain point.		
Access	Virtual, Protected		
Prototype	<pre>virtual void FetchCellHitBy(const SPoint32&inImagePt, TableCellT&outCell);</pre>		
Parameters	The parameters for this method are:		
	<code>const SPoint32& inImagePt</code>	The point.	
	<code>const TableCellT& outCell</code>	The cell.	
Return	None		

FetchLocalCellFrame()

Purpose	See if a cell intersects a given frame.		
Access	Virtual, Protected		
Prototype	<pre>virtual Boolean FetchLocalCellFrame(const TableCellT&inCell, Rect &outCellFrame);</pre>		
Parameters	The parameters for this method are:		
	<code>Rect&</code>	<code>outCellFrame</code>	The frame rect for the cell.
	<code>const TableCellT& inCell</code>	The cell.	
Return	Return true if there is an intersection, else return false.		

GetCellData()

Purpose	Retrieve the data from a cell.		
Access	Virtual, Public		
Prototype	<pre>virtual void GetCellData(const TableCellT&inCell, void *outData);</pre>		
Parameters	The parameters for this method are:		
	const TableCellT&	inCell	The cell.
	void*	outData	The cell data.
Return	None		

GetSelectedCell()

Purpose	Retrieve the contents of a selected cell.		
Access	Public		
Prototype	<pre>void GetSelectedCell(TableCellT&outCell) const;</pre>		
Parameters	The parameters for this method are:		
	TableCellT&	outCell	The cell.
Return	None		

GetTableSize()

Purpose	Retrieve the size of the table.		
Access	Public		
Prototype	<pre>void GetTableSize(TableIndexT&outRows,</pre>		

```
TableIndexT&outCols ) const;
```

Parameters	The parameters for this method are:		
	TableCellT&	outRows	The rows.
	TableCellT&	outCols	The columns.
Return	None		

HiliteCell()

Purpose	Hilite a given cell.		
Access	Virtual, Protected		
Prototype	<pre>virtual void HiliteCell(const TableCellT& inCell);</pre>		
Parameters	The parameters for this method are:		
	TableCellT&	inCell	The cell.
Return	None		

InitTable()

Purpose			
Access	Private		
Prototype	<pre>void InitTable(SInt32inNumberOfRows, SInt32inNumberOfCols, SInt32inRowHeight, SInt32inColWidth, SInt32inCellDataSize);</pre>		
Parameters	The parameters for this method are:		

LTable

	SInt32	inNumberOfRows	The number of rows.
	SInt32	inNumberOfCols	The number of columns
	SInt32	inRowHeight	The row height.
	SInt32	inColWidth	The column width.
	SInt32	inCellDataSize	The cell data size.
Return	None		

InsertCols()

Purpose	Insert columns after a specified column.		
Access	Virtual, Public		
Prototype	<pre>virtual void InsertCols(SInt32 inHowMany, TableIndexT inAfterCol, void* inCellData);</pre>		
Parameters	The parameters for this method are:		
	SInt32	inHowMany	The number of columns.
	TableIndexT	inAfterCol	The column to insert after.
	void*	inCellData	The cell data.
Return	None		

InsertRows()

Purpose	Insert rows after a specified row.
Access	Virtual, Public

Prototype	<pre>virtual void InsertRows(SInt32 inHowMany, TableIndexT inAfterRow, void* inCellData);</pre>		
Parameters	The parameters for this method are:		
	SInt32	inHowMany	The number of rows.
	TableIndexT	inAfterRow	The row to insert after.
	void*	inCellData	The cell data.
Return	None		

IsValidCell()

Purpose	Indicate whether a cell is valid or not.		
Access	Public		
Prototype	<pre>Boolean IsValidCell(const TableCellT& inCell) const;</pre>		
Parameters	The parameters for this method are:		
	TableCellT&	inCell	The row to insert after.
Return	Returns true if valid, else returns false.		

RemoveCols()

Purpose	Remove columns starting after a given column.		
Access	Virtual, Public		
Prototype	<pre>virtual void RemoveCols(SInt32 inHowMany, TableIndexT inFromCol);</pre>		

LTable

Parameters	The parameters for this method are:		
	SInt32	inHowMany	The number of columns to remove.
	TableIndexT	inFromCol	The starting column.
Return	None		

RemoveRows()

Purpose	Remove rows starting after a given row.		
Access	Virtual, Public		
Prototype	<pre>virtual void RemoveRows(SInt32 inHowMany, TableIndexT inFromRow);</pre>		
Parameters	The parameters for this method are:		
	SInt32	inHowMany	The number of rows to remove.
	TableIndexT	inFromRow	The starting row.
Return	None		

SelectCell()

Purpose	Select a given cell.		
Access	Virtual, Public		
Prototype	<pre>virtual void SelectCell(const TableCellT& inCell);</pre>		
Parameters	The parameters for this method are:		
	const TableCellT&	inCell	The cell.

Return	None
--------	------

SetCellData()

Purpose	Set cell data.		
Access	Virtual, Public		
Prototype	<pre>virtual void SetCellData(const TableCellT&inCell, void *inData);</pre>		
Parameters	The parameters for this method are:		
	<code>const TableCellT&</code>	<code>inCell</code>	The cell.
	<code>void*</code>	<code>inData</code>	The data to set.
Return	None		

SetCellDataSize()

Purpose	Set the cell data size.		
Access	Public		
Prototype	<pre>void SetCellDataSize(SInt32inCellDataSize);</pre>		
Parameters	The parameters for this method are:		
	<code>SInt32</code>	<code>inCellDataSize</code>	The data to set.
Return	None		

SetColWidth()

Purpose	Set the column width.
---------	-----------------------

LTable

Access	Virtual, Public		
Prototype	<pre>virtual void SetColWidth(SInt16 inWidth, TableIndexT inFrom, TableIndexT inTo);</pre>		
Parameters	The parameters for this method are:		
	SInt16	inWidth	The width to set.
	TableIndexT	inFrom	The starting index.
	TableIndexT	inTo	The ending index.
Return	None		

SetRowHeight()

Purpose	Set the row height.		
Access	Virtual, Public		
Prototype	<pre>virtual void SetRowHeight(SInt16 inHeight, TableIndexT inFrom, TableIndexT inTo);</pre>		
Parameters	The parameters for this method are:		
	SInt16	inHeight	The height to set.
	TableIndexT	inFrom	The starting index.
	TableIndexT	inTo	The ending index.
Return	None		

UnhiliteCell()

Purpose	Unhilite a cell.
---------	------------------

Access	Virtual, Protected		
Prototype	<code>virtual void UnhiliteCell(const TableCellT& inCell);</code>		
Parameters	The parameters for this method are:		
	<code>const TableCellT&</code>	<code>inCell</code>	The cell.
Return	None		

mRows

Purpose	The number of rows.
Access	Protected
Prototype	<code>TableIndexTmRows;</code>

mCols

Purpose	The number of columns.
Access	Protected
Prototype	<code>TableIndexTmCols;</code>

mRowHeight

Purpose	The row height.
Access	Protected
Prototype	<code>SInt32mRowHeight;</code>

LTable

mColWidth

Purpose	The column width.
Access	Protected
Prototype	SInt32mColWidth;

mCellData

Purpose	The cell data array.
Access	Protected
Prototype	LArray*mCellData;

mSelectedCell

Purpose	The selected cell.
Access	Protected
Prototype	TableCellTmSelectedCell;

LTableArrayStorage

Overview	LTableArrayStorage is a concrete implementation of LTableStorage to support a table where data is stored in an array.	
Methods	The methods in this class are:	
	LTableArrayStorage()	~LTableArrayStorage()
	FindCellData()	GetCellData()
	GetStorageSize()	InsertCols()
	InsertRows()	RemoveCols()
	RemoveRows()	SetCellData()
Data Members	The data members in this class are:	
	mDataArray	mOwnsArray
Operation	LTableArrayStorage adds no new member functions, but implements every function for LTableStorage .	
Source files	(Table Classes)	
	<code>LTableArrayStorage.h</code>	
	<code>LTableArrayStorage.cp</code>	
See also	LTableStorage	
	LArray	

LTableArrayStorage()

Purpose	The constructor creates the object.
Access	Public
Prototype	For cells with the same size data, this creates an LArray object with items equal to the indicated size. <code>LTableArrayStorage(LTableView*inTableView, UInt32inDataSize);</code>

LTableArrayStorage

This is a user-specified subclass of [LArray](#).

```
LTableArrayStorage(  
    LTableView*inTableView,  
    LArray*inDataArray );
```

Parameters None

~LTableArrayStorage()

Purpose The destructor destroys the object.

Access Virtual, Public

Prototype `virtual ~LTableArrayStorage();`

FindCellData()

Purpose This is an override of [FindCellData\(\)](#) in [LTableStorage](#).

GetCellData()

Purpose This is an override of [GetCellData\(\)](#) in [LTableStorage](#).

GetStorageSize()

Purpose This is an override of [GetStorageSize\(\)](#) in [LTableStorage](#).

InsertCols()

Purpose This is an override of [InsertCols\(\)](#) in [LTableStorage](#).

InsertRows()

Purpose This is an override of [InsertRows\(\)](#) in [LTableStorage](#).

RemoveCols()

Purpose This is an override of [RemoveCols\(\)](#) in [LTableStorage](#).

RemoveRows()

Purpose This is an override of [RemoveRows\(\)](#) in [LTableStorage](#).

SetCellData()

Purpose This is an override of [SetCellData\(\)](#) in [LTableStorage](#).

mdataArray

Purpose The data array.

Access Protected

Prototype LArray *mdataArray;

mOwnsArray

Purpose Determines whether the array is destroyed when the LTableArrayStorage object is destroyed.

LTableArrayStorage

Access	Protected	
Prototype	Boolean	mOwnsArray;

LTableGeometry

Overview	LTableGeometry is an abstract PowerPlant class that specifies the interface for the geometry helper objects. These functions provide behaviors to maintain the location, width, and height of each cell in a table.	
Methods	The methods in this class are:	
	LTableGeometry()	~LTableGeometry()
	GetColHitBy()	GetColWidth()
	GetImageCellBounds()	GetRowHeight()
	GetRowHitBy()	GetTableDimensions()
	InsertCols()	InsertRows()
	RemoveCols()	RemoveRows()
	SetColWidth()	SetRowHeight()
Data Members	The data members in this class are:	
	mTableView	
Operation	All the methods in this class are either pure virtual or empty. These methods form the basis for interacting with a table's geometry in PowerPlant. Most of the time you will not need to concern yourself with these methods. The PowerPlant table classes call these methods to get or set required data. In general, you should call the table methods, not the related geometry methods.	
Source files	(Table Classes)	
	UTableHelpers.h	

LTableGeometry()

Purpose	The constructor creates an object.
Access	Public

LTableGeometry

Prototype `LTableGeometry();`

~LTableGeometry()

Purpose The destructor destroys the object.

Access Virtual, Public

Prototype `virtual ~LTableGeometry();`

GetColHitBy()

Purpose Returns the index of a column that contains a point.

Access Pure Virtual, Public

Prototype `virtual TableIndexTGetColHitBy(
 const SPoint32&inImagePt)
 const = 0;`

Parameters This method has the following parameters:

`SPoint32&` `inImagePt` A reference to the point.

Return Returns the index for the table column containing the point.

GetColWidth()

Purpose Retrieve the column width for a specified column.

Access Pure Virtual, Public

Prototype `virtual UInt16GetColWidth(
 TableIndexTinCol) const = 0;`

Parameters This method has the following parameters:

`TableIndexT&` `inCol` A reference to the column.

Return The column.

GetImageCellBounds()

Purpose Provides bounds of cell in image coordinates.

Access Pure Virtual, Public

Prototype `virtual voidGetImageCellBounds(
 const STableCell&inCell,
 SInt32 &outLeft,
 SInt32 &outTop,
 SInt32 &outRight,
 SInt32 &outBottom) const = 0;`

Parameters This method has the following parameters:

STableCell&	inCell	A reference to the cell.
SInt32	outLeft	The left coordinate.
SInt32	outTop	The top coordinate.
SInt32	outRight	The right coordinate.
SInt32	outBottom	The bottom coordinate.

Return None

GetRowHeight()

Purpose Determine the row height.

Access Pure Virtual, Public

Prototype `virtual UInt16GetRowHeight(
 TableIndexTinRow) const = 0;`

Parameters This method has the following parameters:

TableIndexT&	inRow	A reference to the row.
--------------	-------	-------------------------

LTableGeometry

Return The row height.

GetRowHitBy()

Purpose Returns the index of a row that contains a point.

Access Pure Virtual, Public

Prototype `virtual TableIndexTGetRowHitBy(
 const SPoint32&inImagePt) const = 0;`

Parameters This method has the following parameters:

SPoint32& inImagePt A reference to the point.

Return The row index.

GetTableDimensions()

Purpose Determine the dimensions of the table.

Access Pure Virtual, Public

Prototype `virtual voidGetTableDimensions(
 UInt32&outWidth,
 UInt32&outHeight) const = 0;`

Parameters This method has the following parameters:

UInt32& outWidth A reference to the width.

UInt32& outHeight A reference to the height.

Return None

InsertCols()

Purpose Insert columns after a specified column.

Access	Pure Virtual, Public	
Prototype	<pre>virtual voidInsertCols(UInt32 inHowMany, TableIndexT inAfterCol);</pre>	
Parameters	This method has the following parameters:	
UInt32	inHowMany	The number of columns.
TableIndexT	inAfterCol	The column index to insert after.
Return	None	

InsertRows()

Purpose	Insert rows after a given row.	
Access	Pure Virtual, Public	
Prototype	<pre>virtual voidInsertRows(UInt32 inHowMany, TableIndexT inAfterRow);</pre>	
Parameters	This method has the following parameters:	
UInt32	inHowMany	The number of columns.
TableIndexT	inAfterRow	The row index to insert after.
Return	None	

RemoveCols()

Purpose	Remove columns after a specified starting column.	
Access	Pure Virtual, Public	
Prototype	<pre>virtual voidRemoveCols(UInt32inHowMany, TableIndexTinFromCol);</pre>	

LTableGeometry

Parameters This method has the following parameters:

UInt32	inHowMany	The number of columns.
TableIndexT	inFromCol	The column index to start removal after.
Return	None	

RemoveRows()

Purpose The row to start removal at.

Access Pure Virtual, Public

Prototype `virtual voidRemoveRows(
 UInt32inHowMany,
 TableIndexTinFromRow) ;`

Parameters This method has the following parameters:

UInt32	inHowMany	The number of columns.
TableIndexT	inFromRow	The row index to start removal after.
Return	None	

SetColWidth()

Purpose Set the width of a range of columns.

Access Pure Virtual, Public

Prototype `virtual voidSetColWidth(
 UInt16inWidth,
 TableIndexT inFromCol,
 TableIndexTinToCol) = 0;`

Parameters This method has the following parameters:

UInt16	inWidth	The width of the columns.
--------	---------	---------------------------

TableIndexT	inFromCol	The starting column.
TableIndexT	inToCol	The ending column.
Return	None	

SetRowHeight()

Purpose	Set the height on a range of rows.	
Access	Pure Virtual, Public	
Prototype	<pre>virtual voidSetRowHeight(UInt16inHeight, TableIndexTinFromRow, TableIndexTinToRow) = 0;</pre>	
Parameters	This method has the following parameters:	
UInt16	inWidth	The height of the rows.
TableIndexT	inFromRow	The starting row.
TableIndexT	inToRow	The ending row.
Return	None	

mTableView

Purpose	The pointer to the table that owns this helper object.	
Access	Protected	
Prototype	LTableView*mTableView;	

LTableMonoGeometry

Overview	LTableMonoGeometry is a concrete implementation of LTableGeometry to support a table where all cells are the same size.	
Methods	The methods in this class are: LTableMonoGeometry() GetColHitBy() GetImageCellBounds() GetRowHitBy() SetColWidth() ~LTableMonoGeometry() GetColWidth() GetRowHeight() GetTableDimensions() SetRowHeight()	
Data Members	The data members in this class are: mColWidth mRowHeight	
Operation	This class adds no new member functions, but implements every pure virtual function listed in LTableGeometry .	
Source files	(Table Classes) LTableMonoGeometry.h LTableMonoGeometry.cp	
See also	LTableGeometry	

LTableMonoGeometry()

Purpose	The constructor creates the object.
Access	Public
Prototype	<pre>LTableMonoGeometry(LTableView*inTableView, UInt16inColWidth, UInt16inRowHeight);</pre>
Parameters	This method has the following parameters:

LTableMonoGeometry

LTableView*	inTableView	A pointer to the table.
UInt16	inColWidth	The column width.
UInt16	inRowHeight	The row height.

~LTableMonoGeometry()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LTableMonoGeometry();</code>

GetColHitBy()

Purpose	This is an override of GetColHitBy() in LTableGeometry .
---------	--

GetColWidth()

Purpose	This is an override of GetColWidth() in LTableGeometry .
---------	--

GetImageCellBounds()

Purpose	This is an override of GetImageCellBounds() in LTableGeometry .
---------	---

GetRowHeight()

Purpose	This is an override of GetRowHeight() in LTableGeometry .
---------	---

GetRowHitBy()

Purpose This is an override of [GetRowHitBy\(\)](#) in [LTableGeometry](#).

GetTableDimensions()

Purpose This is an override of [GetTableDimensions\(\)](#) in [LTableGeometry](#).

SetColWidth()

Purpose This is an override of [SetColWidth\(\)](#) in [LTableGeometry](#).

SetRowHeight()

Purpose This is an override of [SetRowHeight\(\)](#) in [LTableGeometry](#).

mColWidth

Purpose The width of columns.

Access Protected

Prototype UInt16 mColWidth;

mRowHeight

Purpose The height of rows.

LTableMonoGeometry

Access	Protected	
Prototype	UInt16	mRowHeight;

LTableMultiGeometry

Overview	LTableMultiGeometry is a concrete implementation of LTableGeometry to support a table where rows and columns may vary in size.	
Methods	The methods in this class are: LTableMultiGeometry() GetColHitBy() GetImageCellBounds() GetRowHitBy() InsertCols() RemoveCols() SetColWidth() ~LTableMultiGeometry() GetColWidth() GetRowHeight() GetTableDimensions() InsertRows() RemoveRows() SetRowHeight()	
Data Members	The data members in this class are: mRowHeights mDefaultRowHeight mColWidths mDefaultColWidth	
Operation	LTableMultiGeometry adds no new methods, but implements every method listed in LTableGeometry . The insert and remove of rows and columns methods maintain the arrays of heights and widths for the table.	
Source files	(Table Classes) LTableMultiGeometry.h LTableMultiGeometry.cp	
See also	LTableGeometry	

LTableMultiGeometry()

Purpose	The constructor creates the object.
---------	-------------------------------------

LTableMultiGeometry

Access Public

Prototype LTableMultiGeometry(LTableView*inTableView,
 UInt16inColWidth,
 UInt16inRowHeight);

Parameters This method has the following parameters:

LTableView*	inTableView	A pointer to the table.
UInt16	inColWidth	The column width.
UInt16	inRowHeight	The row height.

~LTableMultiGeometry()

Purpose The destructor destroys the object.

Access Virtual, Public

Prototype virtual ~LTableMultiGeometry();

GetColHitBy()

Purpose This is an override of [GetColHitBy\(\)](#) in [LTableGeometry](#).

GetColWidth()

Purpose This is an override of [GetColWidth\(\)](#) in [LTableGeometry](#).

GetImageCellBounds()

Purpose This is an override of [GetImageCellBounds\(\)](#) in [LTableGeometry](#).

GetRowHeight()

Purpose This is an override of [GetRowHeight\(\)](#) in [LTableGeometry](#).

GetRowHitBy()

Purpose This is an override of [GetRowHitBy\(\)](#) in [LTableGeometry](#).

GetTableDimensions()

Purpose This is an override of [GetTableDimensions\(\)](#) in [LTableGeometry](#).

InsertCols()

Purpose This is an override of [InsertCols\(\)](#) in [LTableGeometry](#).

InsertRows()

Purpose This is an override of [InsertRows\(\)](#) in [LTableGeometry](#).

RemoveCols()

Purpose This is an override of [RemoveCols\(\)](#) in [LTableGeometry](#).

LTableMultiGeometry

RemoveRows()

Purpose This is an override of [RemoveRows\(\)](#) in [LTableGeometry](#).

SetColWidth()

Purpose This is an override of [SetColWidth\(\)](#) in [LTableGeometry](#).

SetRowHeight()

Purpose This is an override of [SetRowHeight\(\)](#) in [LTableGeometry](#).

mRowHeights

Purpose The height of rows.

Access Protected

Prototype LRunArraymRowHeights;

mColWidths

Purpose The width of columns.

Access Protected

Prototype LRunArraymColWidths;

mDefaultRowHeight

Purpose	The default row height.
Access	Protected
Prototype	<code>UInt16mDefaultRowHeight;</code>

mDefaultColWidth

Purpose	The default column width.
Access	Protected
Prototype	<code>UInt16mDefaultColWidth;</code>

LTableMultiSelector

Overview	LTableMultiSelector is a concrete implementation of LTableSelector to support a table where multiple cells may be selected. This class supports discontinuous selection.	
Methods	The methods in this class are:	
	LTableMultiSelector()	~LTableMultiSelector()
	CellIsSelected()	ClickSelect()
	DragSelect()	GetFirstSelectedCell()
	GetFirstSelectedRow()	InsertCols()
	InsertRows()	RemoveCols()
	RemoveRows()	SelectAllCells()
	SelectCell()	SelectCellBlock()
	UnselectAllCells()	UnselectCell()
Data Members	The data members in this class are:	
	mSelectionRgn	mAnchorCell
Operation	This class only adds one new method, SelectCellBlock() .	
Source files	(Table Classes)	
	<code>LTableMultiSelector.h</code>	
	<code>LTableMultiSelector.cp</code>	
See also	LTableSelector	

LTableMultiSelector()

Purpose	The constructor creates the object.
Access	Public
Prototype	<code>LTableMultiSelector(LTableView*inTableView);</code>

LTableMultiSelector

~LTableMultiSelector()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LTableMultiSelector();</code>

CellsSelected()

Purpose	This is an override of CellsSelected() in LTableSelector .
---------	--

ClickSelect()

Purpose	This is an override of ClickSelect() in LTableSelector .
---------	--

DragSelect()

Purpose	This is an override of DragSelect() in LTableSelector .
---------	---

GetFirstSelectedCell()

Purpose	This is an override of GetFirstSelectedCell() in LTableSelector .
---------	---

GetFirstSelectedRow()

Purpose	This is an override of GetFirstSelectedRow() in LTableSelector .
---------	--

InsertCols()

Purpose This is an override of [InsertCols\(\)](#) in [LTableSelector](#).

InsertRows()

Purpose This is an override of [InsertRows\(\)](#) in [LTableSelector](#).

RemoveCols()

Purpose This is an override of [RemoveCols\(\)](#) in [LTableSelector](#).

RemoveRows()

Purpose This is an override of [RemoveRows\(\)](#) in [LTableSelector](#).

SelectAllCells()

Purpose This is an override of [SelectAllCells\(\)](#) in [LTableSelector](#).

SelectCell()

Purpose This is an override of [SelectCell\(\)](#) in [LTableSelector](#).

SelectCellBlock()

Purpose	Select cells within the specified range, where CellA and CellB are the corners of a rectangular block of cells.	
Access	Virtual, Protected	
Prototype	<pre>virtual void SelectCellBlock(const STableCell&inCellA, const STableCell&inCellB);</pre>	
Parameters	This method has the following parameters:	
STableCell&	inCellA	The first cell.
STableCell&	inCellB	The last cell.
Return	None	

UnselectAllCells()

Purpose	This is an override of UnselectAllCells() in LTableSelector .
---------	---

UnselectCell()

Purpose	An override of UnselectCell() in LTableSelector .
---------	---

mSelectionRgn

Purpose	The selection region.
Access	Protected
Prototype	<code>RgnHandle mSelectionRgn;</code>

mAnchorCell

Purpose	The anchor cell.
Access	Protected
Prototype	<code>STableCellmAnchorCell;</code>

LTableSelector

Overview	LTableSelector is an abstract PowerPlant class that specifies the interface for the selector helper objects. These methods provide behaviors to maintain the selection range in a table.	
Methods	The methods in this class are:	
	LTableSelector()	~LTableSelector()
	CellIsSelected()	ClickSelect()
	DragSelect()	GetFirstSelectedCell()
	GetFirstSelectedRow()	InsertCols()
	InsertRows()	RemoveCols()
	RemoveRows()	SelectAllCells()
	SelectCell()	UnselectAllCells()
	UnselectCell()	
Data Members	The data members in this class are:	
	mTableView	
Operation	All of the methods in this class are pure virtual or empty.	
Source files	(Table Classes)	
	UTableHelpers.h	

LTableSelector()

Purpose	The constructor creates the object.
Access	Public
Prototype	<code>LTableSelector() ;</code>
Parameters	None

LTableSelector

~LTableSelector()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LTableSelector();</code>

CellsSelected()

Purpose	Indicates whether a given cell is selected.
Access	Pure Virtual, Public
Prototype	<code>virtual BooleanCellIsSelected(const STableCell&inCell) const = 0;</code>
Parameters	This method has the following parameters:
<code>STableCell&</code>	<code>inCell</code> A reference to the cell.
Return	Returns true if selected, else returns false.

ClickSelect()

Purpose	Handle a mouse click within the specified cell.
Access	Pure Virtual, Public
Prototype	<code>virtual voidClickSelect(const STableCell&inCell, const SMouseDownEvent&inMouseDown) = 0;</code>
Parameters	This method has the following parameters:
<code>STableCell&</code>	<code>inCell</code> A reference to the cell.
<code>SMouseDownEvent&</code>	<code>inMouseDown</code> The mouse down event info.
Return	None

DragSelect()

Purpose	Handle a drag operation (the user is pressing the mouse) on a cell.	
Access	Pure Virtual, Public	
Prototype	<pre>virtual BooleanDragSelect(const STableCell&inCell, const SMouseDownEvent&inMouseDown) = 0;</pre>	
Parameters	This method has the following parameters:	
STableCell&	inCell	A reference to the cell.
SMouseDownEvent&	inMouseDown	The mouse down event info.
Return	Return true if the mouse never leaves inCell.	

GetFirstSelectedCell()

Purpose	Get the value of the first selected cell.	
Access	Pure Virtual, Public	
Prototype	<pre>virtualSTableCellGetFirstSelectedCell() const = 0;</pre>	
Parameters	None	
Return	The cell.	

GetFirstSelectedRow()

Purpose	Get the first selected row.	
Access	Pure Virtual, Public	
Prototype	<pre>virtualTableIndexTGetFirstSelectedRow() const = 0;</pre>	
Parameters	None	

LTableSelector

Return The cell.

InsertCols()

Purpose This method handles insertion of columns to the table.

Access Virtual, Public

Prototype `virtual void InsertCols(
 UInt32 inHowMany,
 TableIndexT inAfterCol);`

Parameters This method has the following parameters:

UInt32 inHowMany How many columns to insert.

TableIndexT inAfterCol The column index to insert after.

Return None

InsertRows()

Purpose This method handles insertion of rows to the table.

Access Virtual, Public

Prototype `virtual void InsertRows(
 UInt32 inHowMany,
 TableIndexT inAfterRow);`

Parameters This method has the following parameters:

UInt32 inHowMany How many rows to insert.

TableIndexT inAfterRow The row index to insert after.

Return None

RemoveCols()

Purpose	This method handles removal of columns to the table.	
Access	Virtual, Public	
Prototype	<pre>virtual void RemoveCols(UInt32 inHowMany, TableIndexT inFromCol);</pre>	
Parameters	This method has the following parameters:	
UInt32	inHowMany	How many columns to insert.
TableIndexT	inFromCol	The column index to start removal at.
Return	None	

RemoveRows()

Purpose	Remove rows from the table.	
Access	Virtual, Public	
Prototype	<pre>virtual void RemoveRows(UInt32 inHowMany, TableIndexT inFromRow) = 0;</pre>	
Parameters	This method has the following parameters:	
UInt32	inHowMany	How many rows to remove.
TableIndexT	inAfterRow	The row index to start removal at.
Return	None	

SelectAllCells()

Purpose	Selects all cells in the table.
---------	---------------------------------

LTableSelector

Access	Pure Virtual, Public
Prototype	<code>virtual voidSelectAllCells() = 0;</code>
Parameters	None
Return	None

SelectCell()

Purpose	Select a cell.
Access	Pure Virtual, Public
Prototype	<code>virtual voidSelectCell(const STableCell&inCell) = 0;</code>
Parameters	This method has the following parameters:
STableCell&	inCell The cell.
Return	None

UnselectAllCells()

Purpose	Unselect all cells in the table.
Access	Pure Virtual, Public
Prototype	<code>virtual voidUnselectAllCells() = 0;</code>
Parameters	None
Return	None

UnselectCell()

Purpose	Unselect a cell.
---------	------------------

Access	Pure Virtual, Public	
Prototype	<pre>virtual voidUnselectCell(const STableCell&inCell) = 0;</pre>	
Parameters	This method has the following parameters:	
STableCell&	inCell	The cell.
Return	None	

mTableView

Purpose	The table view storage.
Access	Protected
Prototype	<pre>LTableView*mTableView;</pre>

LTableSingleSelector

Overview	LTableSingleSelector is a concrete implementation of LTableSelector for a table that may have one and only one cell selected at a time.	
Methods	The methods in this class are:	
	LTableSingleSelector()	~LTableSingleSelector()
	CellIsSelected()	ClickSelect()
	DragSelect()	GetFirstSelectedCell()
	GetFirstSelectedRow()	InsertCols()
	InsertRows()	RemoveCols()
	RemoveRows()	SelectAllCells()
	SelectCell()	UnselectAllCells()
	UnselectCell()	
Data Members	The data members in this class are:	
	mSelectedCell	
Operation	This class adds no new methods, but implements every method in LTableSelector .	
Source files	(Table Classes)	
	LTableSingleSelector.h	
	LTableSingleSelector.cp	
See also	LTableSelector	

LTableSingleSelector()

Purpose	The constructor creates the object.
Access	Public
Prototype	<code>LTableSingleSelector(LTableView*inTableView);</code>

LTableSingleSelector

~LTableSingleSelector()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LTableSingleSelector();</code>

CellsSelected()

Purpose	This is an override of CellsSelected() in LTableSelector .
---------	--

ClickSelect()

Purpose	This is an override of ClickSelect() in LTableSelector .
---------	--

DragSelect()

Purpose	This is an override of DragSelect() in LTableSelector .
---------	---

GetFirstSelectedCell()

Purpose	This is an override of GetFirstSelectedCell() in LTableSelector .
---------	---

GetFirstSelectedRow()

Purpose	This is an override of GetFirstSelectedRow() in LTableSelector .
---------	--

InsertCols()

Purpose This is an override of [InsertCols\(\)](#) in [LTableSelector](#).

InsertRows()

Purpose This is an override of [InsertRows\(\)](#) in [LTableSelector](#).

RemoveCols()

Purpose This is an override of [RemoveCols\(\)](#) in [LTableSelector](#).

RemoveRows()

Purpose This is an override of [RemoveRows\(\)](#) in [LTableSelector](#).

SelectAllCells()

Purpose This is an override of [SelectAllCells\(\)](#) in [LTableSelector](#).

SelectCell()

Purpose This is an override of [SelectCell\(\)](#) in [LTableSelector](#).

LTableSingleSelector

UnselectAllCells()

Purpose This is an override of [UnselectAllCells\(\)](#) in [LTableSelector](#).

UnselectCell()

Purpose This is an override of [UnselectCell\(\)](#) in [LTableSelector](#).

mSelectedCell

Purpose The selected cell.

Access Protected

Prototype `STableCellmSelectedCell;`

LTableStorage

Overview	LTableStorage is an abstract PowerPlant class that specifies the interface for the data storage helper objects. These functions provide behaviors to maintain the data associated with a table.	
Methods	The methods in this class are:	
	LTableStorage()	~LTableStorage()
	FindCellData()	GetCellData()
	GetStorageSize()	InsertCols()
	InsertRows()	RemoveCols()
	RemoveRows()	SetCellData()
Data Members	The data member in this class is:	
	mTableView	
Operation	All functions in this class are pure virtual, so when you inherit from this class you must provide implementations for every method.	
Source files	(Table Classes)	
	UTableHelpers.h	

LTableStorage()

Purpose	The constructor creates an object.
Access	Public
Prototype	<code>LTableStorage()</code> ;
Parameters	None

LTableStorage

~LTableStorage()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LTableStorage();</code>

FindCellData()

Purpose	Search a cell for the specified data.	
Access	Pure Virtual, Public	
Prototype	<code>virtual BooleanFindCellData(STableCell&outCell, const void*inDataPtr, UInt32 inDataSize) const = 0;</code>	
Parameters	This method has the following parameters:	
STableCell&	outCell	A reference to the cell.
void*	inDataPtr	The buffer pointer.
UInt32	inDataSize	The buffer size.
Return	Return true if the data is found.	

GetCellData()

Purpose	Retrieve data from a cell.	
Access	Pure Virtual, Public	
Prototype	<code>virtual voidGetCellData(const STableCell&inCell, void *outDataPtr, UInt32 &ioDataSize) const = 0;</code>	

Parameters	This method has the following parameters:	
STableCell&	inCell	A reference to the cell.
void*	outDataPtr	The buffer pointer.
UInt32	ioDataSize	The buffer size.
Return	None	

GetStorageSize()

Purpose	Get the storage size.	
Access	Pure Virtual, Public	
Prototype	<pre>virtual voidGetStorageSize(TableIndexT&outRows, TableIndexT&outCols) = 0</pre>	
Parameters	This method has the following parameters:	
TableIndexT&	outRows	The number of rows in storage.
TableIndexT&	outCols	The number of columns in storage.
Return	None	

InsertCols()

Purpose	Insert more columns into storage.	
Access	Pure Virtual, Public	
Prototype	<pre>virtual voidInsertCols(UInt32 inHowMany, TableIndexTinAfterCol, const void*inDataPtr, UInt32 inDataSize) = 0;</pre>	
Parameters	This method has the following parameters:	

LTableStorage

UInt32	inHowMany	The number of columns to insert.
TableIndexT	inAfterCol	The column to insert after.
void*	inDataPtr	The buffer pointer.
UInt32	inDataSize	The buffer size.
Return	None	

InsertRows()

Purpose	Insert more rows into storage.	
Access	Pure Virtual, Public	
Prototype	<pre>virtual void InsertRows(UInt32 inHowMany, TableIndexT inAfterRow, const void* inDataPtr, UInt32 inDataSize) = 0;</pre>	
Parameters	This method has the following parameters:	
UInt32	inHowMany	The number of columns to insert.
TableIndexT	inAfterRow	The row to insert after.
void*	inDataPtr	The buffer pointer.
UInt32	inDataSize	The buffer size.
Return	None	

RemoveCols()

Purpose	Remove columns from storage.	
Access	Pure Virtual, Public	
Prototype	<pre>virtual void RemoveCols(</pre>	

```

        UInt32  inHowMany,
        TableIndexTinFromCol ) = 0;

```

Parameters	This method has the following parameters:	
UInt32	inHowMany	The number of columns to remove.
TableIndexT	inFromCol	The column to start removal from.
Return	None	

RemoveRows()

Purpose	Remove rows from storage.	
Access	Pure Virtual, Public	
Prototype	<pre> virtual voidRemoveRows(UInt32 inHowMany, TableIndexTinFromRow) = 0; </pre>	
Parameters	This method has the following parameters:	
UInt32	inHowMany	The number of rows to remove.
TableIndexT	inFromRow	The row to start removal from.
Return	None	

SetCellData()

Purpose	Put data into a storage cell.	
Access	Pure Virtual, Public	
Prototype	<pre> virtual voidSetCellData(const STableCell&inCell, const void*inDataPtr, UInt32 inDataSize) = 0; </pre>	
Parameters	This method has the following parameters:	

LTableStorage

STableCell&	inCell	The cell.
void *	inDataPtr	The buffer pointer.
UInt32	inDataSize	The buffer size.
Return	None	

mTableView

Purpose	Storage for the pointer to the table.
Access	Public
Prototype	LTableView*mTableView;

LTableView

Overview LTableView is a PowerPlant class that forms the basis for the rest of the table classes. The class data members store the data you need to create and manage a table. The class member functions implement standard table-related behavior.

Methods The methods in this class are:

<u>LTableView()</u>	<u>~LTableView()</u>
<u>ActivateSelf()</u>	<u>AdjustImageSize()</u>
<u>CellIsSelected()</u>	<u>CellToIndex()</u>
<u>ClickCell()</u>	<u>ClickSelect()</u>
<u>ClickSelf()</u>	<u>DeactivateSelf()</u>
<u>DrawCell()</u>	<u>DrawSelf()</u>
<u>FetchIntersectingCells()</u>	<u>FindCellData()</u>
<u>GetCellData()</u>	<u>GetCellHitBy()</u>
<u>GetColWidth()</u>	<u>GetCustomHilite()</u>
<u>GetFirstSelectedCell()</u>	<u>GetHiliteRgn()</u>
<u>GetImageCellBounds()</u>	<u>GetLocalCellRect()</u>
<u>GetNextCell()</u>	<u>GetNextSelectedCell()</u>
<u>GetPreviousCell()</u>	<u>GetPreviousSelectedCell()</u>
<u>GetRowHeight()</u>	<u>GetTableGeometry()</u>
<u>GetTableSelector()</u>	<u>GetTableSize()</u>
<u>GetTableStorage()</u>	<u>HiliteCell()</u>
<u>HiliteCellActively()</u>	<u>HiliteCellInactively()</u>
<u>HiliteSelection()</u>	<u>IndexToCell()</u>
<u>InitTable()</u>	<u>InsertCols()</u>
<u>InsertRows()</u>	<u>IsValidCell()</u>
<u>IsValidCol()</u>	<u>IsValidRow()</u>

<u>PointsAreClose()</u>	<u>RefreshCell()</u>
<u>RefreshCellRange()</u>	<u>RemoveAllCols()</u>
<u>RemoveAllRows()</u>	<u>RemoveCols()</u>
<u>RemoveRows()</u>	<u>ScrollCellIntoFrame()</u>
<u>SelectAllCells()</u>	<u>SelectCell()</u>
<u>SelectionChanged()</u>	<u>SetCellData()</u>
<u>SetColWidth()</u>	<u>SetCustomHilite()</u>
<u>SetDeferAdjustment()</u>	<u>SetRowHeight()</u>
<u>SetTableGeometry()</u>	<u>SetTableSelector()</u>
<u>SetTableStorage()</u>	<u>SetUseDragSelect()</u>
<u>UnselectAllCells()</u>	<u>UnselectCell()</u>

Data Members The data members in this class are:

<u>mRows</u>	<u>mCols</u>
<u>mTableGeometry</u>	<u>mTableSelector</u>
<u>mTableStorage</u>	<u>mUseDragSelect</u>
<u>mCustomHilite</u>	<u>mDeferAdjustment</u>

Operation LTableView inherits from LPane and LView. It is thus closely related to the operations of LView. An understanding of how LPane and LView work is essential to getting the most from this class.

Source files (Table Classes)

LTableView.h

LTableView.cp

See also [LAttachable](#)

[LView](#)

[LPane](#)

LTableView()

Purpose	The constructor creates an object.	
Access	Public	
Prototype	<pre>LTableView(); LTableView(const SPaneInfo&inPaneInfo, const SViewInfo&inViewInfo); LTableView(LStream*inStream);</pre>	
Parameters	This method has the following parameters:	
SPaneInfo&	inPaneInfo	A reference to the pane info.
SViewInfo&	inBefore	A reference to the superview.
LStream*	inStream	A pointer to the stream object to construct from.

~LTableView()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<pre>virtual ~LTableView();</pre>

ActivateSelf()

Purpose	Activate a Table
Access	Virtual, Protected
Prototype	<pre>virtual void ActivateSelf();</pre>
Parameters	None
Return	None

AdjustImageSize()

Purpose	Adjust the Image size of the Table to reflect the number and size of the rows and columns.	
Access	Virtual, Public	
Prototype	<code>virtual void AdjustImageSize(BooleaninRefresh);</code>	
Parameters	This method has the following parameters:	
Boolean	inRefresh	A boolean indicating whether to refresh or not.
Return	None	

CellsSelected()

Purpose	Return whether the specified cell is part of the current selection	
Access	Public	
Prototype	<code>virtual Boolean CellIsSelected(const STableCell&inCell) const;</code>	
Parameters	This method has the following parameters:	
STableCell&	inCell	A reference to the cell.
Return	Boolean indicating true if the cell is part of the current selection	

CellToIndex()

Purpose	Pass back the index number for a specified cell
	The cell does not have to be in the Table, but the index is zero if the table has no columns or inCell.row is zero.
	Cells are ordered by column (across), and then by row (down)

Access	Virtual, Public	
Prototype	virtual void CellToIndex(const STableCell&inCell, TableIndexT&outIndex) const;	
Parameters	This method has the following parameters:	
STableCell&	inCell	A reference to the cell.
TableIndexT&	outIndex	The index.

ClickCell()

Purpose	Handle a mouse click within the specified cell.	
Access	Virtual, Protected	
Prototype	virtual void ClickCell(const STableCell& inCell, const SMouseDownEvent& inMouseDown);	
Parameters	This method has the following parameters:	
STableCell&	inCell	A reference to the cell.
SMouseDownEvent&	inMouseDown	The mouse down event info.

ClickSelect()

Purpose	Adjust selection in response to a click in the specified cell, and return whether or not to process the click as a normal click.	
Access	Virtual, Public	
Prototype	virtual Boolean ClickSelect(const STableCell&inCell, const SMouseDownEvent&inMouseDown);	
Parameters	This method has the following parameters:	

LTableView

STableCell&	inCell	A reference to the cell.
SMouseDownEvent&	inMouseDown	The mouse down event info.
Return	Return true if a normal click, false otherwise.	

ClickSelf()

Purpose	Handle a mouse click within a TableView. This method is an override of ClickSelf() .
---------	--

DeactivateSelf()

Purpose	Deactivate a Table. This method is an override of DeactivateSelf() .
---------	--

DrawCell()

Purpose	Draw the contents of the specified Cell.	
Access	Virtual, Protected	
Prototype	virtual void DrawCell(const STableCell& inCell, const Rect& inLocalRect);	
Parameters	This method has the following parameters:	
STableCell&	inCell	A reference to the cell.
Rect&	inLocalRect	The rectangle coordinates of the cell.
Return	None	

DrawSelf()

Purpose Draw a TableView. This is an override of [DrawSelf\(\)](#) in [LPane\(\)](#).

FetchIntersectingCells()

Purpose Pass back the rectangular range of cells, specified by the top left and bottom right cells, that intersects a given rectangle.

Access Virtual, Public

Prototype

```
virtual void FetchIntersectingCells(  
    const Rect&inLocalRect,  
    STableCell&outTopLeft,  
    STableCell&outBotRight) const;
```

Parameters This method has the following parameters:

STableCell&	outTopLeft	A reference to the top left cell.
STableCell&	outBotRight	A reference to the bottom right cell.
Rect&	inLocalRect	The rectangle coordinates of the cell.

Return None

FindCellData()

Purpose Pass back the cell that contains the specified data.

Access Virtual, Public

Prototype

```
virtual Boolean FindCellData(  
    STableCell&outCell,  
    const void*inDataPtr,  
    UInt32 inDataSize) const;
```

Parameters This method has the following parameters:

LTableView

STableCell&	outCell	A reference to the cell.
void*	inDataPtr	The buffer pointer.
UInt32	inDataSize	The buffer size.
Return	Return false if no match is found.	

GetCellData()

Purpose	Pass back the data for a particular Cell.	
Access	Virtual, Public	
Prototype	<pre>virtual void GetCellData(const STableCell&inCell, void *outDataPtr, UInt32 &ioDataSize) const;</pre>	
Parameters	This method has the following parameters:	
STableCell&	outCell	A reference to the cell.
void*	outDataPtr	The buffer pointer. Points to storage which must be allocated by the caller. It may be nil, in which case only the size of the data is passed back.
UInt32&	ioDataSize	The size of the data buffer (
		If outDataPtr is nil
		<ul style="list-style-type: none">• input:<ignored>• output:size in bytes of cell's data
		If outDataPtr is not nil,
		<ul style="list-style-type: none">• input:maximum bytes of data to retrieve• output:actual bytes of data passed back
Return	None	

GetCellHitBy()

Purpose	Pass back the cell which contains the specified point. If no cell contains the point, return false and outCell.row = 0 if point is above the Table outCell.row = mRows + 1 if point is below the Table outCell.col = 0 if point is to the left of the Table outCell.col = mCols + 1 if point is to the right of the Table For example, if the horizontal coordinate is within Column 2, but the vertical coordinate is above the Table, then, outCell.row = 0 outCell.col = 2	
Access	Virtual, Public	
Prototype	<pre>virtual Boolean GetCellHitBy(const SPoint32&inImagePt, STableCell&outCell) const;</pre>	
Parameters	This method has the following parameters:	
SPoint32&	inImagePt	A reference to the point.
STableCell&	outCell	A reference to the cell.
Return	Return false if no cell contains the point.	

GetColWidth()

Purpose	Return the width of the specified column.
Access	Virtual, Public
Prototype	<pre>virtual UInt16 GetColWidth(</pre>

LTableView

```
TableIndexTinCol ) const;
```

Parameters This method has the following parameters:

TableIndexT inCol The column to interrogate.

Return UInt16 indicating the width.

GetCustomHilite()

Purpose Returns the value of the [mCustomHilite](#) member.

Access Public

Prototype `Boolean GetCustomHilite();`

Parameters None

GetFirstSelectedCell()

Purpose Return the first selected cell, using the LTableSelector helper object.

Access Public, Virtual

Prototype `virtual STableCell GetFirstSelectedCell() const;`

Parameters None

Return The first selected cell.

GetHiliteRgn()

Purpose Pass back a Region containing the frames of all selected cells which are within the visible rectangle of the Table.

Caller must allocate space for the region.

Access Virtual, Protected

Prototype	<code>virtual void GetHiliteRgn(RgnHandle ioHiliteRgn);</code>	
Parameters	This method has the following parameters:	
RgnHandle	ioHiliteRgn	The region handle.
Return	None	

GetImageCellBounds()

Purpose	Pass back the location in Image coords of the specified Cell.	
Access	Virtual, Public	
Prototype	<code>virtual void GetImageCellBounds(const STableCell&inCell, SInt32 &outLeft, SInt32 &outTop, SInt32 &outRight, SInt32 &outBottom) const;</code>	
Parameters	This method has the following parameters:	
STableCell&	inCell	The cell.
SInt32&	outLeft	The left side.
SInt32&	outTop	The top border.
SInt32&	outRight	The right side.
SInt32&	outBottom	The bottom border.
Return	None	

GetLocalCellRect()

Purpose	Pass back the bounding rectangle of the specified Cell and return whether it intersects the Frame of the TableView.
---------	---

LTableView

The bounding rectangle is in Local coordinates so it will always be within QuickDraw space when its within the Frame.

Access Virtual, Public

Prototype

```
virtual Boolean GetLocalCellRect(  
    const STableCell&inCell,  
    Rect      &outCellRect) const;
```

Parameters This method has the following parameters:

STableCell& inCell The cell.

Rect& outCellRect The rect of the cell.

Return If the bounding rectangle is outside the Frame, return false and set the rectangle to (0,0,0,0).

GetNextCell()

Purpose Pass back the Cell after the specified Cell.

Cells are ordered by column (across), and then by row (down).

Row zero is before the first row. The next cell after row zero and any column is Cell (1,1).

Column zero is before column one. The next cell after row "r" and column zero is Cell (r,1).

Access Virtual, Public

Prototype

```
virtual Boolean GetNextCell(  
    STableCell&ioCell) const;
```

Parameters This method has the following parameters:

STableCell& ioCell The cell.

Return Return false if there is no cell after the specified one, and pass back Cell (0,0). Otherwise, return true and pass back the next Cell's indexes.

GetNextSelectedCell()

Purpose	Pass back the selected Cell after the specified Cell. This function uses the same ordering rules as GetNextCell() . Pass in Cell (0,0) to find the first selected Cell.		
Access	Virtual, Public		
Prototype	<pre>virtual Boolean GetNextSelectedCell(STableCell&ioCell) const;</pre>		
Parameters	This method has the following parameters:		
STableCell&	ioCell	The cell.	
Return	Return false if there is no selected Cell after the specified one.		

GetPreviousCell()

Purpose	Pass back the Cell before the specified Cell. Cells are ordered by column (across), and then by row (down). Cell (mRows,mCols) is the one before Cell (0,c), where c is any column. Thus, you can pass in Cell (0,0) to get the last cell. Cell (mRows,mCols) is also the cell before any Cell that is beyond the limits of the table. Column zero is before column one. The cell before row "r" and column zero is Cell (r-1,mCols).		
Access	Virtual, Public		
Prototype	<pre>virtual Boolean GetPreviousCell(STableCell&ioCell) const;</pre>		
Parameters	This method has the following parameters:		
STableCell&	ioCell	The cell.	

LTableView

Return	Return false if there is no cell before the specified one, and pass back Cell (0,0). Otherwise, return true and pass back the previous Cell's indexes.
--------	--

GetPreviousSelectedCell()

Purpose	Pass back the selected Cell before the specified Cell.
---------	--

This function uses the same ordering rules as [GetPreviousCell\(\)](#). Pass in Cell (0,0) to find the last selected Cell.

Access	Virtual, Public
--------	-----------------

Prototype	<pre>virtual Boolean GetPreviousSelectedCell(STableCell&ioCell) const;</pre>
-----------	--

Parameters	This method has the following parameters:
------------	---

STableCell&	ioCell	The cell.
-------------	--------	-----------

Return	Return false if there is no selected cell before the cell indicated, and pass back Cell (0,0). Otherwise, return true and pass back the previous Cell's indexes.
--------	--

GetRowHeight()

Purpose	Return the height of the specified row.
---------	---

Access	Virtual, Public
--------	-----------------

Prototype	<pre>virtual UInt16 GetRowHeight(TableIndexTinRow) const;</pre>
-----------	---

Parameters	This method has the following parameters:
------------	---

TableIndexT	inRow	The table index.
-------------	-------	------------------

Return	A UInt16 indicating the height of the row i display coordinates.
--------	--

GetTableGeometry()

Purpose	Return the value of the mTableGeometry data member.
Access	Public
Prototype	<code>LTableGeometry* GetTableGeometry();</code>
Parameters	None
Return	A pointer to an LTableGeometry object.

GetTableSelector()

Purpose	Return the value of the mTableSelector data member.
Access	Public
Prototype	<code>LTableSelector* GetTableSelector();</code>
Parameters	None
Return	A pointer to an LTableSelector object.

GetTableSize()

Purpose	Pass back the number of rows and columns in a TableView.	
Access	Public	
Prototype	<code>void GetTableSize(TableIndexT&outRows, TableIndexT&outCols) const;</code>	
Parameters	This method has the following parameters:	
TableIndexT&	outRows	The number of rows.
TableIndexT&	outCols	The number of columns.

LTableView

Return None

GetTableStorage()

Purpose Returns the value of the [mTableStorage](#) data member.

Access Public

Prototype `LTableStorage* GetTableStorage();`

Parameters None

Return A pointer to an LTableStorage object.

HiliteCell()

Purpose Draw or undraw hiliting for the specified Cell

Access Virtual, Public

Prototype `virtual void HiliteCell(
 const STableCell&inCell,
 Boolean inHilite);`

Parameters This method has the following parameters:

STableCell&	inCell	The cell.
Boolean	inHilite	Whether to hilite or not.

Return None

HiliteCellActively()

Purpose Draw or undraw active hiliting for a Cell.

Access Virtual, Protected

Prototype	<pre>virtual void HiliteCellActively(const STableCell&inCell, Boolean inHilite);</pre>	
Parameters	This method has the following parameters:	
STableCell&	inCell	The cell.
Boolean	inHilite	Whether to hilite or not.
Return	None	

HiliteCellInactively()

Purpose	Draw or undraw inactive hiliting for a Cell	
Access	Virtual, Protected	
Prototype	<pre>virtual void HiliteCellInactively(const STableCell&inCell, Boolean inHilite);</pre>	
Parameters	This method has the following parameters:	
STableCell&	inCell	The cell.
Boolean	inHilite	Whether to hilite or not.
Return	None	

HiliteSelection()

Purpose	Draw or undraw hiliting for the current selection in either the active or inactive state	
Access	Virtual, Public	
Prototype	<pre>virtual void HiliteSelection(BooleaninActively, BooleaninHilite);</pre>	

LTableView

Parameters	This method has the following parameters:	
Boolean	inActively	Whether actively or not.
Boolean	inHilite	Whether to hilite or not.
Return	None	

IndexToCell()

Purpose	Pass back the cell for a specified index number.	
	Index number does not have to refer to an actual Cell, but Cell is (0,0) if Table has no columns or inIndex is zero.	
Access	Virtual, Public	
Prototype	<pre>virtual void IndexToCell(TableIndexTinIndex, STableCell&outCell) const;</pre>	
Parameters	This method has the following parameters:	
TableIndexT	inIndex	The table index.
STableCell&	outCell	The cell.
Return	None	

InitTable()

Purpose	Private Initializer	
Access	Private	
Prototype	<pre>void InitTable();</pre>	
Parameters	None	
Return	None	

InsertCols()

Purpose	Add columns to a TableView.	
	Use inAfterCol of 0 to insert columns at the beginning All cells in the newly inserted rows have the same data.	
Access	Virtual, Public	
Prototype	<pre>virtual void InsertCols(UInt32 inHowMany, TableIndexT inAfterCol, const void* inDataPtr, UInt32 inDataSize, Boolean inRefresh);</pre>	
Parameters	This method has the following parameters:	
UInt32	inHowMany	How many rows to add.
TableIndexT	inAfterCol	The column to insert after.
void*	inDataPtr	The buffer pointer.
UInt32	inDataSize	The size of the buffer.
Boolean	inRefresh	Whether to refresh or not.
Return	None	

InsertRows()

Purpose	Add rows to a TableView.	
	Use inAfterRow of 0 to insert rows at the beginning All cells in the newly inserted rows have the same data.	
Access	Virtual, Public	
Prototype	<pre>virtual void InsertRows(UInt32 inHowMany, TableIndexT inAfterRow,</pre>	

LTableView

```
const void* inDataPtr,  
UInt32 inDataSize,  
Boolean inRefresh);
```

Parameters	This method has the following parameters:	
UInt32	inHowMany	How many rows to add.
TableIndexT	inAfterRow	The row to insert after.
void*	inDataPtr	The buffer pointer.
UInt32	inDataSize	The size of the buffer.
Boolean	inRefresh	Whether to refresh or not.
Return	None	

IsValidCell()

Purpose	Return whether a TableView includes a specified Cell.	
Access	Public	
Prototype	Boolean IsValidCell(const STableCell&inCell) const;	
Parameters	This method has the following parameters:	
STableCell&	inCell	The cell index to check.
Return	Boolean indicating whether the cell is valid or not.	

IsValidCol()

Purpose	Return whether a TableView includes a specified column.	
Access	Public	
Prototype	Boolean IsValidCol(TableIndexTinCol) const;	

Parameters	This method has the following parameters:	
STableCell&	inCol	The cell column to check.
Return	Boolean indicating whether the cell includes the column or not.	

IsValidRow()

Purpose	Return whether a TableView includes a specified row	
Access	Public	
Prototype	<pre>Boolean IsValidRow(TableIndexTinRow) const;</pre>	
Parameters	This method has the following parameters:	
TableIndexT	inRow	The cell row to check.
Return	Boolean indicating whether the cell includes the row or not.	

PointsAreClose()

Purpose	Indicate whether the two points are close enough to be part of a multi-click. Points are in Local coordinates.	
	Points must meet the standard test (inherited function) as well as be inside the same cell.	
Access	Virtual, Public	
Prototype	<pre>virtual Boolean PointsAreClose(PointinFirstPt, PointinSecondPt) const;</pre>	
Parameters	This method has the following parameters:	
Point	inFirstPt	The first point.
Point	inSecondPt	The second point.

LTableView

Return	Return true if the two points are close enough, else return false.
--------	--

RefreshCell()

Purpose	Invalidate the area occupied by the specified cell so that its contents will be redrawn during the next update event
---------	--

Access	Virtual, Public
--------	-----------------

Prototype	<pre>virtual void RefreshCell(const STableCell&inCell);</pre>
-----------	---

Parameters	This method has the following parameters:
------------	---

STableCell&	inCell	The cell.
-------------	--------	-----------

Return	None
--------	------

RefreshCellRange()

Purpose	Invalidate a rectangular block of cells so that their contents will be redrawn during the next update event.
---------	--

Access	Virtual, Public
--------	-----------------

Prototype	<pre>virtual void RefreshCellRange(const STableCell&inTopLeft, const STableCell&inBotRight);</pre>
-----------	--

Parameters	This method has the following parameters:
------------	---

STableCell&	inTopLeft	The top left cell.
-------------	-----------	--------------------

STableCell&	inBotRight	The bottom right cell.
-------------	------------	------------------------

Return	None
--------	------

RemoveAllCols()

Purpose	Remove all columns in a table.	
Access	Virtual, Public	
Prototype	<pre>virtual void RemoveAllCols(BooleaninRefresh);</pre>	
Parameters	This method has the following parameters:	
Boolean	inRefresh	Whether to refresh or not.
Return	None	

RemoveAllRows()

Purpose	Remove all rows from a table.	
Access	Virtual, Public	
Prototype	<pre>virtual void RemoveAllRows(BooleaninRefresh);</pre>	
Parameters	This method has the following parameters:	
Boolean	inRefresh	Whether to refresh or not.
Return	None	

RemoveCols()

Purpose	Delete columns from a TableView	
Access	Virtual, Public	
Prototype	<pre>virtual void RemoveCols(UInt32inHowMany, TableIndexTinFromCol,</pre>	

UITableView

`BooleaninRefresh);`

Parameters	This method has the following parameters:	
UInt32	inHowMany	How many columns to delete.
TableIndexT	inFromCol	The column to start from.
Boolean	inRefresh	Whether or not to refresh.
Return	None	

RemoveRows()

Purpose	Delete rows from a TableView	
Access	Virtual, Public	
Prototype	<code>virtual void RemoveRows(UInt32inHowMany, TableIndexTinFromRow, BooleaninRefresh);</code>	
Parameters	This method has the following parameters:	
UInt32	inHowMany	How many rows to delete.
TableIndexT	inFromRow	The row to start from.
Boolean	inRefresh	Whether or not to refresh.
Return	None	

ScrollCellIntoFrame()

Purpose	Scroll the TableView as little as possible to move the specified Cell so that it's entirely within the Frame of the TableView. If Cell is wider and/or taller than Frame, align Cell to left/top of Frame.
Access	Virtual, Public

Prototype	<code>virtual void ScrollCellIntoFrame(const STableCell&inCell);</code>	
Parameters	This method has the following parameters:	
STableCell&	inCell	The cell.
Return	None	

SelectAllCells()

Purpose	Select all Cells in a Table
Access	Virtual, Public
Prototype	<code>virtual void SelectAllCells();</code>
Parameters	None
Return	None

SelectCell()

Purpose	Add the specified cell to the current selection.	
Access	Virtual, Public	
Prototype	<code>virtual void SelectCell(const STableCell&inCell);</code>	
Parameters	This method has the following parameters:	
STableCell&	inCell	The cell.
Return	None	

SelectionChanged()

Purpose	Notification that the cells which are selected has changed
Access	Virtual, Public
Prototype	<code>virtual void SelectionChanged();</code>
Parameters	None
Return	None

SetCellData()

Purpose	Specify the data associated with a particular Cell	
Access	Virtual, Public	
Prototype	<code>virtual void SetCellData(const STableCell&inCell, const void*inDataPtr, UInt32 inDataSize);</code>	
Parameters	This method has the following parameters:	
STableCell&	inCell	The cell.
void*	inDataPtr	The buffer pointer.
UInt32	inDataSize	The buffer size.
Return	None	

SetColWidth()

Purpose	Set the width of the specified columns.
Access	Virtual, Public
Prototype	<code>virtual void SetColWidth(</code>

```

        UInt16 inWidth,
        TableIndexT inFromCol,
        TableIndexT inToCol);

```

Parameters	This method has the following parameters:	
UInt16	inWidth	The cell width.
TableIndexT	inFromCol	The starting column.
TableIndexT	inToCol	The ending column.
Return	None	

SetCustomHilite()

Purpose	Set the value of the mCustomHilite data member.	
Access	Public	
Prototype	<code>void SetCustomHilite(Boolean inCustom);</code>	
Parameters	This method has the following parameters:	
Boolean	inCustom	The value to set.
Return	None	

SetDeferAdjustment()

Purpose	Specify whether to defer adjustment of Table to account for a change in the size of the Table (number or size of rows or columns).
	<p>Calling <code>AdjustImageSize()</code> can take a lot of time, especially for large Tables and those with variable row or column sizes. If you are going to make a series of changes to a Table, there's no need to adjust the Image size until after all those changes are made.</p> <p>Therefore, call <code>SetDeferAdjustment(true)</code> before making your changes, then call <code>SetDeferAdjustment(false)</code> afterwards. Or use the <code>StDeferTableAdjustment</code> stack-based class.</p>

LTableView

Access	Public	
Prototype	<code>void SetDeferAdjustment(BooleaninDefer);</code>	
Parameters	This method has the following parameters:	
Boolean	inDefer	Whether to defer or not.
Return	None	

SetRowHeight()

Purpose	Set the height of the specified rows.	
Access	Virtual, Public	
Prototype	<code>virtual void SetRowHeight(UInt16inHeight, TableIndexTinFromRow, TableIndexTinToRow);</code>	
Parameters	This method has the following parameters:	
UInt16	inHeight	The height to use.
TableIndexT	inFromRow	The starting row.
TableIndexT	inToRow	The ending row.
Return	None	

SetTableGeometry()

Purpose	Specify the Geometry for a TableView.	
	The Geometry determines the dimensions of the Cells in the Table	
Access	Virtual, Public	
Prototype	<code>virtual void SetTableGeometry(LTableGeometry*inTableGeometry);</code>	
Parameters	This method has the following parameters:	

LTableGeometry*	inTableGeometry	The table geometry object.
-----------------	-----------------	----------------------------

Return	None
--------	------

SetTableSelector()

Purpose	Specify the Selector for a TableView. The Selector stores and controls which Cells in a Table are selected. This sets the value of mTableSelector .
---------	---

Access	Virtual, Public
--------	-----------------

Prototype	<pre>virtual void SetTableSelector(LTableSelector*inTableSelector);</pre>
-----------	---

Parameters	This method has the following parameters:
------------	---

LTableSelector*	inTableSelector	The table selector pointer.
-----------------	-----------------	-----------------------------

Return	None
--------	------

SetTableStorage()

Purpose	Specify the Storage for a TableView. The Storage holds the data for each Cell in a Table. This sets the value of mTableStorage .
---------	--

Access	Virtual, Public
--------	-----------------

Prototype	<pre>virtual void SetTableStorage(LTableStorage*inTableStorage);</pre>
-----------	--

Parameters	This method has the following parameters:
------------	---

LTableStorage*	inTableStorage	The table storage pointer.
----------------	----------------	----------------------------

Return	None
--------	------

SetUseDragSelect()

Purpose	Indicate whether to use drag select or not. This sets the value of the mUseDragSelect data member.	
Access	Public	
Prototype	<code>void SetUseDragSelect(Boolean inUseIt);</code>	
Parameters	This method has the following parameters:	
Boolean	inUseIt	The value to set.
Return	None	

UnselectAllCells()

Purpose	Unselect all currently selected cells so there is no selection	
Access	Virtual, Public	
Prototype	<code>virtual void UnselectAllCells();</code>	
Parameters	None	
Return	None	

UnselectCell()

Purpose	Remove the specified cell from the current selection	
Access	Virtual, Public	
Prototype	<code>virtual void UnselectCell(const STableCell &inCell);</code>	
Parameters	This method has the following parameters:	
STableCell&	inCell	The cell.

Return	None
--------	------

mRows

Purpose	Data member that contains number of rows.
Access	Protected
Prototype	TableIndexT mRows;

mCols

Purpose	Data member that contains number of columns.
Access	Protected
Prototype	TableIndexT mCols;

mTableGeometry

Purpose	Storage for the table geometry.
Access	Protected
Prototype	LTableGeometry *mTableGeometry;

mTableSelector

Purpose	Storage for the table selector
Access	Protected
Prototype	LTableSelector *mTableSelector;

LTableView

mTableStorage

Purpose	Table storage.
Access	Protected
Prototype	<code>LTableStorage *mTableStorage;</code>

mUseDragSelect

Purpose	Storage for whether or not to use drag select.
Access	Protected
Prototype	<code>Boolean mUseDragSelect;</code>

mCustomHilite

Purpose	Whether to use custom hilite or not.
Access	Protected
Prototype	<code>Boolean mCustomHilite;</code>

mDeferAdjustment

Purpose	Whether to defer adjustments to the table or not.
Access	Protected
Prototype	<code>Boolean mDeferAdjustment;</code>

LTCPEndpoint

Overview `LTCPEndpoint` is a PowerPlant class that is used for implementing Internet TCP endpoints. TCP/IP is the session-oriented protocol of the Internet, and is the layer upon which most of the familiar Internet protocols (HTTP, FTP, SMTP, etc.) are built.

Methods The methods in this class are:

<code>LTCPEndpoint()</code>	<code>~LTCPEndpoint()</code>
<code>AbortiveDisconnect()</code>	<code>AcceptIncoming()</code>
<code>AcceptRemoteDisconnect()</code>	<code>Connect()</code>
<code>Disconnect()</code>	<code>GetAmountUnread()</code>
<code>GetRemoteHostAddress()</code>	<code>Listen()</code>
<code>Receive()</code>	<code>ReceiveChar()</code>
<code>ReceiveData()</code>	<code>ReceiveDataUntilMatch()</code>
<code>ReceiveLine()</code>	<code>RejectIncoming()</code>
<code>Send()</code>	<code>SendCStr()</code>
<code>SendData()</code>	<code>SendDisconnect()</code>
<code>SendHandle()</code>	<code>SendPStr()</code>
<code>SendPtr()</code>	

Data Members There are no data members in this class.

Operation In PowerPlant, there are two subclasses of `LTCPEndpoint`, named `LMacTCPTCPEndpoint` and `LOpenTptTCPEndpoint`. The appropriate class is created automatically when you call [`UNetworkFactory::CreateTCPEndpoint\(\)`](#).

Note that most of these methods are pure virtual, requiring you to implement them if you inherit from this class.

Source files (Networking Classes)

`LTCPEndpoint.h`

`LTCPEndpoint.cp`

LTCPEndpoint

See also [LEndpoint](#)

LTCPEndpoint()

Purpose	The constructor creates the object.
Access	Public
Prototype	<code>LTCPEndpoint();</code>
Parameters	None

~LTCPEndpoint()

Purpose	The destructor destroys the LTCPEndpoint object.
Access	Virtual, Public
Prototype	<code>virtual ~LTCPEndpoint();</code>

AbortiveDisconnect()

Purpose	Called for a disconnect on abort. You must override this and provide functionality if you inherit from this class.
Access	Pure virtual, Public
Prototype	<code>virtual void AbortiveDisconnect() = 0;</code>
Parameters	None
Return	None

AcceptIncoming()

Purpose	Call to accept an incoming connection.	
Access	Pure virtual, Public	
Prototype	<pre>virtual void AcceptIncoming(LTCPEndpoint* inEndpoint) = 0;</pre>	
Parameters	The parameters for this method are:	
	LTCPEndpoint* inEndpoint	The endpoint.
Return	None	

AcceptRemoteDisconnect()

Purpose	Called for accepting a remote disconnect.	
Access	Pure virtual, Public	
Prototype	<pre>virtual void AcceptRemoteDisconnect() = 0;</pre>	
Parameters	None	
Return	None	

Connect()

Purpose	Called to connect.	
Access	Pure virtual, Public	
Prototype	<pre>virtual void Connect(LInternetAddress& inRemoteAddress, UInt32 inTimeoutSeconds = Timeout_None) = 0;</pre>	
Parameters	The parameters for this method are:	

LTCPEndpoint

	LInternetAddress& UInt32	inRemoteAddress inTimeoutSeconds	The address. The timeout value, default is Timeout_None.
Return	None		

Disconnect()

Purpose	Called to disconnect.
Access	Pure virtual, Public
Prototype	<code>virtual void Disconnect() = 0;</code>
Parameters	None
Return	None

GetAmountUnread()

Purpose	Learn how much unread data remains.
Access	Pure virtual, Public
Prototype	<code>virtual UInt32 GetAmountUnread() = 0;</code>
Parameters	None
Return	Returns the amount of unread data.

GetRemoteHostAddress()

Purpose	Retrieve the address of the remote host.
Access	Pure virtual, Public

Prototype	<code>virtual LInternetAddress *GetRemoteHostAddress() = 0;</code>
Parameters	None
Return	A pointer to the LInternetAddress.

Listen()

Purpose	Listen for data.
Access	Pure virtual, Public
Prototype	<code>virtual void Listen() = 0;</code>
Parameters	None
Return	None

Receive()

Purpose	Receive data.		
Access	Pure virtual, Public		
Prototype	virtual void Receive(void* outDataBuffer, UInt32& ioDataSize);		
Parameters	The parameters for this method are:		
	void*	outDataBuffer	The buffer pointer.
	UInt32&	ioDataSize	The data size.
Return	None		

ReceiveChar()

Purpose	Receive a character.	
Access	Virtual, Public	
Prototype	<pre>virtual Boolean ReceiveChar(char& outChar, UInt32 inTimeoutSeconds = Timeout_None);</pre>	
Parameters	The parameters for this method are:	
	char&	outChar
	UInt32	inTimeoutSeconds
		The buffer pointer.
		The timeout, the default is Timeout_None.
Return	Return true if a character is received.	

ReceiveData()

Purpose	Receive data over a connection.	
Access	Pure virtual, Public	
Prototype	<pre>virtual void ReceiveData(void* outDataBuffer, UInt32& ioDataSize, Boolean& outExpedited, UInt32 inTimeoutSeconds = Timeout_None) = 0;</pre>	
Parameters	The parameters for this method are:	
	void*	outDataBuffer
	UInt32	ioDataSize
		The buffer pointer.
		The data size.

	Boolean&	outExpedited	Whether expedited or not.
	UInt32	inTimeoutSeconds	The timeout, the default is Timeout_None.
Return	None		

ReceiveDataUntilMatch()

Purpose	Receive data until a specified character is found.		
Access	Virtual, Public		
Prototype	<pre>virtual Boolean ReceiveDataUntilMatch(void* outDataBuffer, UInt32& ioDataSize, Boolean& outExpedited, UInt32 inTimeoutSeconds, char inMatchChar = 0x0D);</pre>		
Parameters	The parameters for this method are:		
	void*	outDataBuffer	The buffer pointer.
	UInt32	ioDataSize	The data size.
	Boolean&	outExpedited	Whether expedited or not.
	UInt32	inTimeoutSeconds	The timeout, the default is Timeout_None.
	char	inMatchChar	The character to search for, the default is 0x0D.
Return	Returns true if the specified character is found before the timeout, else returns false.		

ReceiveLine()

Purpose	Receive a line of data.		
Access	Virtual, Public		
Prototype	<pre>virtual Boolean ReceiveLine(char * outString, UInt32& ioDataSize, UInt32 inTimeoutSeconds, Boolean inUseLF = false);</pre>		
Parameters	The parameters for this method are:		
	char*	outString	The buffer pointer to fill.
	UInt32&	ioDataSize	The data size.
	UInt32	inTimeoutSeconds	The timeout, the default is Timeout_None.
	Boolean	inUseLF	Whether to use a line feed or not, the default is false. If true then we look for a LF instead of CR to terminate the line.
Return	Returns the value of ReceiveDataUntilMatch() .		

RejectIncoming()

Purpose	Call to reject incoming data.
Access	Pure virtual, Public
Prototype	<pre>void RejectIncoming() = 0;</pre>
Parameters	None
Return	None

Send()

Purpose	Calls SendData() to send data.	
Access	Virtual, Public	
Prototype	<pre>virtual void Send(void* inData, UInt32 inDataSize);</pre>	
Parameters	The parameters for this method are:	
	void*	inData
	The buffer pointer to send.	
	UInt32&	inDataSize
	The data size.	
Return	None	

SendCStr()

Purpose	Utility method for SendData() .	
Access	Virtual, Public	
Prototype	<pre>virtual void SendCStr(char* inString);</pre>	
Parameters	The parameters for this method are:	
	char*	inString
	The C string to send.	
Return	None	

SendData()

Purpose	A routine to actually send the data out.
Access	Pure virtual, Public

LTCPEndpoint

Prototype `virtual void SendData(void* inData,
 UInt32 inDataSize,
 Boolean inExpedited = false,
 UInt32 inTimeoutSeconds = Timeout_None) = 0;`

Parameters The parameters for this method are:

void*	inData	The data to send.
UInt32	inDataSize	The data size.
Boolean	inExpedited	Whether to expedite or not, default is false.
UInt32	inTimeoutSeconds	The timeout, default is Timeout_None.

Return None

SendDisconnect()

Purpose Send a disconnect.

Access Pure virtual, Public

Prototype `virtual void SendDisconnect() = 0;`

Parameters None

Return None

SendHandle()

Purpose Utility method for [SendData\(\)](#).

Access Virtual, Public

Prototype `virtual void SendHandle(Handle inHandle);`

Parameters The parameter for this method is:

	<table><tr><th>Handle</th><th>inHandle</th><th>The handle to send.</th></tr></table>	Handle	inHandle	The handle to send.
Handle	inHandle	The handle to send.		
Return	None			

SendPStr()

Purpose	Utility method for SendData() .		
Access	Virtual, Public		
Prototype	virtual void SendPStr(ConstStringPtr inString);		
Parameters	The parameter for this method is:		
	ConstStringPtr	inString	The Pascal string to send.
Return	None		

SendPtr()

Purpose	Utility method for SendData() .		
Access	Virtual, Public		
Prototype	virtual void SendPtr(Ptr inPtr);		
Parameters	The parameter for this method is:		
	ConstStringPtr	inString	The data to send.
Return	None		

LTEClearAction

Overview	LTEClearAction is a PowerPlant class that is used for handling the clear action for a TextEdit field.
Methods	The methods in this class are: LTEClearAction() RedoSelf() ~LTEClearAction()
Data Members	There are no data members in this class.
Operation	This is a simple class for handling the clearing of the TextEdit field.
Source files	(Action Classes) UTETextAction.h UTETextAction.cp
Ancestors	LTETextAction

LTEClearAction()

Purpose	The constructor creates the object.
Access	Public
Prototype	<code>LTEClearAction();</code>
Parameters	None

~LTEClearAction()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LTEClearAction();</code>

LTClearAction

RedoSelf()

Purpose This is an override of [RedoSelf\(\)](#) in LAction.

LTECutAction

Overview	LTECutAction is a PowerPlant class that is used for handling the cut action for a TextEdit field.
Methods	The methods in this class are: LTECutAction() ~LTECutAction() RedoSelf()
Data Members	There are no data members in this class.
Operation	This is a simple class for handling cuts from the TextEdit field.
Source files	(Action Classes) UTETextAction.h UTETextAction.cp
Ancestors	LTETextAction

LTECutAction()

Purpose	The constructor creates an object from the passed-in parameters.	
Access	Public	
Prototype	LTECutAction(TEHandle inMacTEH, LCommander *inTextCommander, LPane *inTextPane);	
Parameters	The parameters for this constructor are:	
	TEHandle inMacTEH	The Mac OS TextEdit handle.
	LCommander* inTextCommander	The pointer to the text Commander.
	LPane* inTextPane	The pointer to the Pane.

LTECutAction

~LTECutAction()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LTECutAction();</code>

RedoSelf()

Purpose	This is an override of RedoSelf() in LAction.
---------	---

LTEPasteAction

Overview	LTEPasteAction is a PowerPlant class that is used for handling the paste action for a TextEdit field.		
Methods	The methods in this class are:		
	LTEPasteAction()	~LTEPasteAction()	
	RedoSelf()	UndoSelf()	
Data Members	The data members in this class are:		
	mPastedTextH		
Operation	This is a simple class for handling pastes from the TextEdit field.		
Source files	(Action Classes)		
	UTETextAction.h		
	UTETextAction.cp		
Ancestors	LTETextAction		

LTEPasteAction()

Purpose	The constructor creates an object from the passed-in parameters.		
Access	Public		
Prototype	LTEPasteAction(TEHandle inMacTEH, LCommander *inTextCommander, LPane *inTextPane);		
Parameters	The parameters for this constructor are:		
	TEHandle	inMacTEH	The Mac OS TextEdit handle.

LTEPasteAction

LCommander*	inTextCommander	The pointer to the text Commander.
LPane*	inTextPane	The pointer to the Pane.

~LTEPasteAction()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LTEPasteAction();</code>

RedoSelf()

Purpose	This is an override of RedoSelf() in LAction.
---------	---

UndoSelf()

Purpose	This is an override of UndoSelf() in LAction.
---------	---

mPastedTextH

Purpose	The handle to the pasted text.
Access	Protected
Prototype	<code>Handle mPastedTextH;</code>

LTETextAction

Overview	LTETextAction is a PowerPlant class that is used for handling the common editing actions for a TextEdit field.	
Methods	The methods in this class are:	
	LTETextAction()	~LTETextAction()
	CanRedo()	CanUndo()
	Redo()	Undo()
	UndoSelf()	
Data Members	The data members in this class are:	
	mTextCommander	mTextPane
	mMacTEH	mActionCommand
	mDeletedTextH	mDeletedTextLen
	mSelStart	mSelEnd
Operation	This is a simple class for handling operations for the TextEdit field.	
Source files	(Action Classes)	
	UTETextAction.h	
	UTETextAction.cp	
Ancestors	LAction	
	LTETextAction	

LTETextAction()

Purpose	The constructor creates an object from the passed-in parameters.
Access	Public

LTETextAction

Prototype	LTETextAction(SInt16 inDescriptionIndex, CommandT inActionCommand, TEHandle inMacTEH, LCommander *inTextCommander, LPane *inTextPane, Boolean inAlreadyDone);		
Parameters	The parameters for this constructor are:		
	SInt16	inDescriptionIndex	The description index to send to the LAction() constructor.
	CommandT	inActionCommand	The action command.
	TEHandle	inMacTEH	The Mac OS TextEdit handle.
	LCommander *	inTextCommander	The commander.
	LPane*	inTextPane	The pane for the text.
	Boolean	inAlreadyDone	This is passed to the LAction() constructor.

~LTETextAction()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	virtual ~LTETextAction();

CanRedo()

Purpose	This is an override of CanRedo() in LAction.
---------	--

CanUndo()

Purpose This is an override of [CanUndo\(\)](#) in LAction.

Redo()

Purpose This is an override of [Redo\(\)](#) in LAction.

Undo()

Purpose This is an override of [Undo\(\)](#) in LAction.

UndoSelf()

Purpose This is an override of [UndoSelf\(\)](#) in LAction.

mTextCommander

Purpose Pointer storage for the text Commander.

Access Protected

Prototype LCommander *mTextCommander;

mTextPane

Purpose A pointer to the text Pane.

LTETextAction

Access	Protected
Prototype	LPane *mTextPane;

mMacTEH

Purpose	The Mac OS TextEdit handle.
Access	Protected
Prototype	TEHandle mMacTEH;

mActionCommand

Purpose	Storage for the action command.
Access	Protected
Prototype	MessageT mActionCommand;

mDeletedTextH

Purpose	Storage for the handle to deleted text.
Access	Protected
Prototype	Handle mDeletedTextH;

mDeletedTextLen

Purpose	Storage for the length of the deleted text.
Access	Protected
Prototype	SInt32 mDeletedTextLen;

mSelStart

Purpose	The start of the selection.
Access	Protected
Prototype	<code>SInt16 mSelStart;</code>

mSelEnd

Purpose	The end of the selection.
Access	Protected
Prototype	<code>SInt16 mSelEnd;</code>

LTETypingAction

Overview	LTETypingAction is a PowerPlant class that is used for handling common typing actions.	
Methods	The methods in this class are:	
	LTETypingAction()	~LTETypingAction()
	BackwardErase()	ForwardErase()
	InputCharacter()	RedoSelf()
	Reset()	UndoSelf()
Data Members	The data members in this class are:	
	mTypedTextH	mTypingStart
	mTypingEnd	
Operation	This is a simple class for handling typing in the TextEdit field.	
Source files	(Action Classes)	
	UTETextAction.h	
	UTETextAction.cp	
Ancestors	LAction	
	LTETextAction	

LTETypingAction()

Purpose	The constructor creates the object from the passed-in parameters.
Access	Public
Prototype	<code>LTETypingAction(TEHandle inMacTEH, LCommander *inTextCommander, LPane *inTextPane);</code>
Parameters	The parameters for this constructor are:

LTETypingAction

TEHandle	inMacTEH	The Mac OS TextEdit handle.
LCommander*	inTextCommander	The pointer to the text Commander.
LPane*	inTextPane	The pointer to the Pane.

~LTETypingAction()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LTETypingAction();</code>

BackwardErase()

Purpose	Handle the backward delete typing action. Backward delete erases the current selection if one or more characters are selected. If the selection is a single insertion point, then backward delete erases the one character before the insertion point
Access	Virtual, Public
Prototype	<code>virtual void BackwardErase();</code>
Parameters	None
Return	None

ForwardErase()

Purpose	Handle forward delete typing action. Forward delete erases the current selection if one or more characters are selected. If the
---------	---

selection is a single insertion point, then forward delete erases the one character after the insertion point.

Access	Virtual, Public
Prototype	<code>virtual void ForwardErase();</code>
Parameters	None
Return	None

InputCharacter()

Purpose	Handle an input character typing action.
Access	Virtual, Public
Prototype	<code>virtual void InputCharacter(SInt16 inChar);</code>
Parameters	The parameters for this constructor are: <div style="display: flex; justify-content: space-around; margin-top: 5px;"> SInt16 inChar The character that was typed. </div>
Return	None

RedoSelf()

Purpose	Redo a TypingAction by restoring the last typing sequence. This is an override of RedoSelf() in LAction.
---------	--

Reset()

Purpose	Re-initialize the state of the typing action.
Access	Virtual, Public
Prototype	<code>virtual void Reset();</code>

LTETypingAction

Parameters	None
------------	------

Return	None
--------	------

UndoSelf()

Purpose	Undo a typing action by restoring the text and selection that existed before the current typing sequence started. This is an override of UndoSelf() in LAction.
---------	---

mTypedTextH

Purpose	The handle to the typed text.
---------	-------------------------------

Access	Protected
--------	-----------

Prototype	Handle mTypedTextH;
-----------	---------------------

mTypingStart

Purpose	The start of the typing.
---------	--------------------------

Access	Protected
--------	-----------

Prototype	SInt16 mTypingStart;
-----------	----------------------

mTypingEnd

Purpose	The end of the typing.
---------	------------------------

Access	Protected
--------	-----------

Prototype	SInt16 mTypingEnd;
-----------	--------------------

LTextButton

Overview	LTextButton is a PowerPlant class that is used for managing buttons that have text instead of graphics in them. There is no standard Mac OS item that matches this class.	
Methods	The methods in this class are:	
	LTextButton()	~LTextButton()
	DrawSelf()	GetDescriptor()
	HotSpotAction()	HotSpotResult()
	SetDescriptor()	SetValue()
Data Members	The data members in this class are:	
	mText	mTextTraitsID
	mSelectedStyle	
Operation	<p>Clicking an LTextButton toggles the button's state between on and off, like a radio button. The different state is represented visually by a change in the text style. The button style may become underline, bold, italic, outline, shadow, condensed, extended, or any combination of these style options.</p> <p>When the user clicks the button, a message is sent to all the button's Listeners. In a typical scenario, you want some action to occur immediately when the user clicks a text button. You should specify a value message that uniquely identifies the button to any Listener.</p>	
Source files	(Pane Classes)	
	<code>LTextButton.h</code>	
	<code>LTextButton.cp</code>	
Ancestors	The ancestors for this class are:	
	LAttachable	LControl
	LBroadcaster	LPane

LTextButton

LTextButton()

Purpose	The constructors create objects from the passed-in parameters.	
Access	Public	
Prototype	<pre>LTextButton() ; LTextButton(LStream* inStream) ;</pre>	
Parameters	The stream constructor has the following parameter:	
	<pre>LStream* inStream</pre>	A pointer to a stream object that contains the information to create the LTextButton object.

~LTextButton()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<pre>virtual ~LTextButton() ;</pre>

DrawSelf()

Purpose	Draw this text in with appropriate text style for the selected status. This is an override of DrawSelf() in LPane .
---------	---

GetDescriptor()

Purpose	Return the text of the button. This is an override of GetDescriptor() in LPane .
---------	--

HotSpotAction()

Purpose Toggle between highlighted and plain states, depending on mouse location. This is an override of [HotSpotAction\(\)](#) in [LControl](#).

HotSpotResult()

Purpose Update the value of the control and send a message to the Listeners. This is an override of [HotSpotResult\(\)](#) in [LControl](#).

SetDescriptor()

Purpose Reset the title and update region of the button. This is an override of [SetDescriptor\(\)](#) in [LPane](#).

SetValue()

Purpose Reset button value, update region, and send message. This is an override of [SetValue\(\)](#) in [LControl](#).

mText

Purpose This data member contains the text that is drawn in the button.

Access Protected

Prototype LStr255 mText;

LTextButton

mTextTraitsID

Purpose	This data member contains the resource ID of the resource that contains the text traits information.
Access	Protected
Prototype	ResIDT mTextTraitsID;

mSelectedStyle

Purpose	This data member contains the information that determines the style the button text will be drawn with.
Access	Protected
Prototype	SInt16 mSelectedStyle;

LTextView

Overview LTextView is a PowerPlant class that is used for wrapping the TextEdit functionality of the Mac OS Toolbox.

Methods The methods in this class are:

<u>LTextView()</u>	<u>~LTextView()</u>
<u>AdjustCursorSelf()</u>	<u>AdjustImageToText()</u>
<u>AlignTextEditRects()</u>	<u>BeTarget()</u>
<u>CalcTEHeight()</u>	<u>ChangeFontSizeBy()</u>
<u>ClickSelf()</u>	<u>ClickAutoScroll()</u>
<u>DontBeTarget()</u>	<u>DrawSelf()</u>
<u>FindCommandStatus()</u>	<u>FocusDraw()</u>
<u>ForceAutoScroll()</u>	<u>GetAlignment()</u>
<u>GetAttributes()</u>	<u>GetColor()</u>
<u>GetDescriptor()</u>	<u>GetFont()</u>
<u>GetMacTEH()</u>	<u>GetSelection()</u>
<u>GetSize()</u>	<u>GetStyle()</u>
<u>GetTextHandle()</u>	<u>GetTextTraitsID()</u>
<u>GetValue()</u>	<u>HandleKeyPress()</u>
<u>HasAttribute()</u>	<u>HideSelf()</u>
<u>InitTextView()</u>	<u>Insert()</u>
<u>MoveBy()</u>	<u>MyClickLoop()</u>
<u>MyTEClick()</u>	<u>ObeyCommand()</u>
<u>ResizeFrameBy()</u>	<u>RestorePlace()</u>
<u>SavePlace()</u>	<u>SaveStateForUndo()</u>
<u>ScrollImageBy()</u>	<u>SelectAll()</u>
<u>SetAlignment()</u>	<u>SetAttributes()</u>

LTextEditView

	SetClickLoop()	SetColor()
	SetDescriptor()	SetFont()
	SetSize()	SetStyle()
	SetTextHandle()	SetTextPtr()
	SetTextTraitsID()	SpendTime()
	TETooBig()	ToggleAttribute()
	UserChangedText()	
Data Members	The data members in this class are:	
	mClickLoopUPP	mTextEditH
	mTextTraitsID	mTextAttributes
	mTypingAction	
Operation	LTextEditView allows you to specify the font, style, size, color, and justification of the text you create.	
Source files	(Pane Classes)	
	LTextEditView.h	
	LTextEditView.cp	
Ancestors	LCommander	
	LPeriodical	
	LView	

LTextEditView()

Purpose	The constructor creates objects from the passed-in parameters.
Access	Public
Prototype	<code>LTextEditView();</code>


```
LTextView( const SPaneInfo& inPaneInfo,  
const SViewInfo& inViewInfo,  
UInt16 inTextAttributes,  
ResIDT inTextTraitsID );  
LTextView( LStream *inStream );
```

Parameters The parameters for these constructors are:

const SPaneInfo&	inPaneInfo	The Superpane info.
const SViewInfo &	inViewInfo	The Superview info.
UInt16	inTextAttributes	The text attributes.
ResIDT	inTextTraitsID	The text traits.
LStream*	inStream	The stream to read from.

~LTextView()

Purpose The destructor destroys the object.

Access Virtual, Public

Prototype `virtual ~LTextView();`

AdjustCursorSelf()

Purpose TextEdit uses the standard I-Beam cursor. This is an override of [AdjustCursorSelf\(\)](#) in [LPane](#).

AdjustImageToText()

Purpose	This method resizes the image to match the size of the text.
Access	Virtual, Public
Prototype	<code>virtual void AdjustImageToText();</code>
Parameters	None
Return	None

AlignTextEditRects()

Purpose	Align the view and destination rectangles of the Toolbox TextEdit record with the Frame of a TextEdit.
Access	Virtual, Protected
Prototype	<code>virtual void AlignTextEditRects();</code>
Parameters	None
Return	None

BeTarget()

Purpose	TextEdit is becoming the Target.
Access	Virtual, Protected
Prototype	<code>virtual void BeTarget();</code>
Parameters	None
Return	None

CalcTEHeight()

Purpose	Calculates the height of the TextEdit record.
Access	Virtual, Public
Prototype	SInt32 LTextView::CalcTEHeight()
Parameters	none
Return	Height of TextEdit record as 32-bit value.

ChangeFontSizeBy()

Purpose	Changes the font size for a given selection range by a specified value.
Access	Virtual, Public
Prototype	Boolean LTextView::ChangeFontSizeBy(SInt16 inDeltaSize)
Parameters	 SInt16 inDeltaSize Value to change font size by
Return	Returns True if change was successful.
Remarks	Will only work if the font over the selection range is constant.

ClickSelf()

Purpose	Respond to Click inside a TextEdit. This is an override of ClickSelf() in LPane .
---------	---

ClickAutoScroll()

Purpose	Used in the click loop to perform scrolling while clicking.		
Access	Virtual, Protected		
Prototype	<code>void LTextEdit::ClickAutoScroll(const Point &inLocalPoint)</code>		
Parameters	<code>const Point</code>	<code>&inLocalPoint</code>	Point clicked in
Return	none		

DontBeTarget()

Purpose	TextEdit is no longer the Target. Remove TextEdit from the IdleQueue. This is an override of DontBeTarget() in LCommander .
---------	---

DrawSelf()

Purpose	Draw a TextEdit. This is an override of DrawSelf() in LPane .
---------	---

FindCommandStatus()

Purpose	This method passes back the status of a command. This is an override of FindCommandStatus() in LCommander .
---------	---

FocusDraw()

Purpose Prepare for drawing in the TextEdit. This is an override of [FocusDraw\(\)](#) in [LPane](#).

ForceAutoScroll()

Purpose Works with TextEdit's autoscrolling capabilities to keep text and scrollbars synchronized.

GetAlignment()

Purpose Returns the justification setting for the TextEdit record.

GetAttributes()

Purpose Returns the raw attribute flags.

Access Public

Remarks Treated as read only.

GetColor()

Purpose Returns the RGB value of the selected text.

Access Virtual, Public

GetDescriptor()

Purpose	Returns up to the first 255 characters of the TextView as a Pascal string.
Access	Virtual, Public
Prototype	<pre>StringPtr LTextView::GetDescriptor(Str255 outDescriptor) const</pre>
Remarks	Caller must allocate memory for the Str255 variable for storing the string.

GetFont()

Purpose	Determine the font for the selection range. There are two versions of this routine. One passes back the font name, the other passes back the font number.
Access	Virtual, Public
Prototype	<pre>Boolean LTextView::GetFont(SInt16 &outFontNum) Boolean LTextView::GetFont(Str255 &outName)</pre>

GetMacTEH()

Purpose	Retrieve theHandle to the Mac TextEdit Record associated with a TextEdit. Caller may change record fields, and is responsible for redrawing the TextEdit as necessary to reflect any changes. However, caller must not dispose of the TEHandle.
Access	Public
Prototype	<pre>TEHandle GetMacTEH();</pre>

Parameters	None
Return	Handle to the Mac TextEdit Record associated with a TextEdit.

GetSelection()

Purpose	Passes back the AppleEvent Descriptor of the selected text.
---------	---

GetSize()

Purpose	Returns the size of the style record for the text range.
---------	--

GetStyle()

Purpose	Returns the style of the selected text range.
---------	---

GetTextHandle()

Purpose	Retrieve a Handle to the text in the LTextView. The Handle is the actual Handle used by the Toolbox TextEdit record. Treat this Handle as read-only.
Access	Virtual, Public
Prototype	<code>virtual Handle GetTextHandle();</code>
Parameters	None
Return	Return a Handle to the text in the LTextView.

LTextView

GetTextTraitsID()

Purpose	Returns the ID of the current TextTraits record.
Access	Public

GetValue()

Purpose	Return an integer value represented by the contents of the TextView.
Access	Virtual, Public
Prototype	<code>SInt32 LTextView::GetValue() const</code>
Parameters	none
Return	32-bit integer
Remarks	An empty or non-numerical TextView evaluates to zero.

HandleKeyPress()

Purpose	Handle key stroke directed at an TextEdit. This is an override of HandleKeyPress() in LCommander .
---------	--

HasAttribute()

Purpose	Check for the presence of a specific attribute.
Access	Public
Prototype	<code>Boolean HasAttribute(UInt16 inAttribute);</code>
Parameters	The parameter for this method is:

UInt16	inAttribute	The attribute to check for.
--------	-------------	-----------------------------

Return	Boolean indicating whether the attribute is present, false if not.
--------	--

HideSelf()

Purpose	Hide an LTextView. An invisible LTextView can't be OnDuty. This is an override of HideSelf() in LPane .
---------	---

InitTextView()

Purpose	Initialize member variables of a TextEdit to default values.
---------	--

Access	Private
--------	---------

Prototype	<code>void InitTextView(ResIDT inTextTraitsID);</code>
-----------	--

Parameters	The parameter for this method is:
------------	-----------------------------------

ResIDT	inTextTraitsID	The text traits to set.
--------	----------------	-------------------------

Return	None
--------	------

Insert()

Purpose	Replacement for <code>TEInsert()</code> that allows for when text is greater than 32K. Call this routine in place of <code>TEInsert()</code> or <code>TEStyleInsert()</code> . Will optionally recalculate, autoscroll, and refresh the text if desired.
---------	--

Access	Virtual, Public
--------	-----------------

MoveBy()

Purpose	Move the location of the Frame by the specified amounts <code>inHorizDelta</code> and <code>inVertDelta</code> specify, in pixels, how far to move the Frame (within its surrounding Image). Positive horizontal deltas move to the right, negative to the left. Positive vertical deltas move down, negative up. This is an override of MoveBy() in LPane .
---------	--

MyClickLoop()

Purpose	ClickLoop callback used to help autoscrolling while selecting text.
Access	Static, Public

MyTEClick()

Purpose	Handle a click in a TextEdit field. Identical to the standard TEClick funtion, but includes code to handle a problem with clickloop functions in naitive mode.
Access	Virtual, Public

ObeyCommand()

Purpose	Issue a command to a Commander. This is an override of ObeyCommand() in LCommander .
---------	--

ResizeFrameBy()

Purpose Change the Frame size by the specified amounts `inWidthDelta` and `inHeightDelta` specify, in pixels, how much larger to make the Frame. Positive deltas increase the size, negative deltas reduce the size. This is an override of [ResizeFrameBy\(\)](#) in [LPane](#).

RestorePlace()

Purpose Restore TextEdit parameters from a stream.

Access Virtual, Public

Prototype `virtual void RestorePlace(LStream *inPlace);`

Parameters The parameter for this method is:

<code>LStream*</code>	<code>inPlace</code>	The stream to read from.
-----------------------	----------------------	--------------------------

Return None

SavePlace()

Purpose Save TextEdit parameters to a stream.

Access Virtual, Public

Prototype `virtual void SavePlace(LStream *outPlace);`

Parameters The parameter for this method is:

<code>LStream*</code>	<code>outPlace</code>	The stream to write to.
-----------------------	-----------------------	-------------------------

Return None

SaveStateForUndo()

Purpose	Preserve the state so that an Undo operation is possible.
Access	Virtual, Protected
Prototype	<code>virtual STextEditUndoH SaveStateForUndo();</code>
Parameters	None
Return	Returns a handle to the Undo information.

ScrollImageBy()

Purpose	Scroll the Text. This is an override of ScrollImageBy() in LView .
---------	--

SelectAll()

Purpose	Select entire contents of an TextEdit.
Access	Virtual, Public
Prototype	<code>virtual void SelectAll();</code>
Parameters	None
Return	None

SetAlignment()

Purpose	Changes the justification of the TextEdit record.
Access	Virtual, Public

SetAttributes()

Purpose	Allows the setting of attributes. Replaced all existing attributes with the value passed as the argument.
Access	Public
Remarks	If you want to change only a single attribute, use ToggleAttribute() instead.

SetClickLoop()

Purpose	Allows you to specify your own clickloop funtion.		
Access	Virtual, Public		
Prototype	void LTextView::SetClickLoop(void *inClickLoop)		
Parameters			
	void	*inClickLoop	Pointer to a clickloop function
Return	none		
Remarks	Disposes of existing clickloop, if any. Also since the point of the clickloop is for autoscrolling, that attribute is set here.		

SetColor()

Purpose	Sets the color of the text range
Access	Virtual, Public

SetDescriptor()

Purpose	Sets the text to a given string
Access	Virtual, Public
Remarks	Replaces any and all text already in the descriptor.

SetFont()

Purpose	Given a font name or font number, set the font of the selected text accordingly.
Access	Virtual, Public
Prototype	<pre>void LTextView::SetFont(SInt16 inFontNumber) void LTextView::SetFont(ConstStringPtr inFontName)</pre>

SetSize()

Purpose	Sets the font size of the text range.
Access	Virtual, Public

SetStyle()

Purpose	Sets the text style of the selected text.
Access	Virtual, Public

SetTextHandle()

Purpose	Set the text in the LTextView to the contents of the specified Handle. The LTextView copies the data in the Handle, so the caller retains ownership of the Handle (and should dispose of it as needed).	
Access	Virtual, Public	
Prototype	<code>virtual void SetTextHandle(Handle inTextH);</code>	
Parameters	The parameter for this method is:	
	Handle	inTextH The text handle to set to.
Return	None	

SetTextPtr()

Purpose	Set the pointer to the text.	
Access	Virtual, Public	
Prototype	<code>virtual void SetTextPtr(Ptr inTextP, SInt32 inTextLen);</code>	
Parameters	The parameters for this method are:	
	Ptr	inTextP The pointer to the text.
	SInt32	inTextLen The length of the text.
Return	None	

SetTextTraitsID()

Purpose	Specify the resource ID of the TextTraits for an TextEdit. This method updates the line height to fit the text characteristics.				
Access	Virtual, Public				
Prototype	<pre>virtual void SetTextTraitsID(ResIDT inTextTraitsID);</pre>				
Parameters	The parameter for this method is:				
	<table><tr><td>ResIDT</td><td>inTextTraitsID</td><td>The resource ID for the text traits.</td></tr></table>	ResIDT	inTextTraitsID	The resource ID for the text traits.	
ResIDT	inTextTraitsID	The resource ID for the text traits.			
Return	None				

SpendTime()

Purpose	Flash the insertion cursor during idle time. This is an override of SpendTime() in LPeriodical .
---------	--

TETooBig()

ToggleAttribute()

Purpose	Allows the toggling of an attribute setting.
Access	Public
Remarks	If you want to change all attributes, use SetAttributes() instead.

UserChangedText()

Purpose	Text of TextEdit has changed as a result of user action. You should override this to validate the field and/or dynamically update as the user types. This function is not called by <code>SetDescriptor()</code> , which is typically used to programatically change the text.
Access	Virtual, Public
Prototype	<code>virtual void UserChangedText();</code>
Parameters	None
Return	None

mTextEditH

Purpose	Storage for the handle to the TextEdit record.
Access	Protected
Prototype	<code>TEHandle mTextEditH;</code>

mTextTraitsID

Purpose	Storage for the text traits.
Access	Protected
Prototype	<code>ResIDT mTextTraitsID;</code>

LTextEditView

mTextAttributes

Purpose	Storage for the text attributes.
Access	Protected
Prototype	UInt16 mTextAttributes;

LToggleButton

Overview LToggleButton is a PowerPlant class that implements a button. The graphic for the button needs to be stored as a resource of one of these types:

- ICN#
- ICON'
- 'PICT'

This class is very similar to [LButton](#), except that you may specify five separate resource ID numbers, not just two resource ID numbers as an [LButton](#) would have.

Methods The methods in this class are:

[LToggleButton\(\)](#)

[DrawGraphic\(\)](#)

[DrawSelf\(\)](#)

[HotSpotAction\(\)](#)

[HotSpotResult\(\)](#)

[PointIsInFrame\(\)](#)

[SetGraphics\(\)](#)

[SetGraphicsType\(\)](#)

[SetValue\(\)](#)

Data Members The data members in this class are:

[mGraphicsType](#)

[mOnID](#)

[mOnClickID](#)

[mOffID](#)

[mOffClickID](#)

[mTransitionID](#)

Operation You can use LToggleButton to make some simple animations that play when a button is clicked. For example, you could create a drop-down flag button, or a door that opens and closes when the button is clicked. In addition to the “on” and “off” states of the button that [LButton](#) uses, you also specify graphics for:

- A click on a button that is already “on”
- A click on a button that is already “off,” and
- A transition graphic that displays when the button is switching states.

Source files (Pane Classes)

LToggleButton

LToggleButton.h

LToggleButton.cp

Ancestors [LBroadcaster](#)

[LControl](#)

[LPane](#)

See also [LButton](#)

LToggleButton()

Purpose The constructor creates the objects from the passed-in parameters.

Access Public

Prototype `LToggleButton();
LToggleButton(const SPaneInfo &inPaneInfo,
MessageT inClickedMessage,
OSType inGraphicsType,
ResIDT inOnID,
ResIDT inOnClickID,
ResIDT inOffID,
ResIDT inOffClickID,
ResIDT inTransitionID);`

Parameters These constructors have the following parameters:

const LButton&	inOriginal	A reference to the LButton object you want to copy.
const SPaneInfo&	inPaneInfo	A reference to the SPaneInfo object that is the super view.
MessageT	inClickedMessage	The message sent when the button is depressed.
OSType	inGraphicsType	Graphics Type ('ICN#', or 'ICON', or 'PICT')
ResIDT	inOnID	Resource ID for “on” graphic

ResIDT	inOnClickID	Resource ID for the “clicked and on” graphic
ResIDT	inOffID	Resource ID for “off” graphic
ResIDT	inOffClickID	Resource ID for the “clicked and off” graphic
ResIDT	inTransitionID	Resource ID for the transition graphic
LStream*	inStream	A pointer to a stream object that contains the information to create the LButton object.

DrawGraphic()

Purpose	Draw the graphic for a the button. The Pane must already be focused before calling this method.	
Access	Virtual, Protected	
Prototype	virtual void DrawGraphic(ResIDT inGraphicID);	
Parameters	This method has the following parameter:	
	ResIDT	inGraphicID Resource ID for the button graphic to draw.
Return	None	

DrawSelf()

Purpose	This method is an override of the base class method DrawSelf() in LPane . It draws the button.
---------	--

HotSpotAction()

Purpose	This method is an override of the base class method HotSpotAction() in LControl . It causes the buttons to toggle
---------	---

LToggleButton

between two graphics, depending on whether the mouse is inside or outside the button.

HotSpotResult()

Purpose This method is an override of the base class method [HotSpotResult\(\)](#) in [LControl](#). It broadcasts a message to Listeners when the button is clicked.

PointIsInFrame()

Purpose This method is an override of the base class method [PointIsInFrame\(\)](#) in [LPane](#). It gives you information about whether a point lies within a given frame.

SetGraphics()

Purpose Specify the resources IDs for all states of the button.

Access Virtual, Public

Prototype

```
virtual void SetGraphics( ResIDT inOnID,  
ResIDT inOnClickID,  
ResIDT inOffID,  
ResIDT inOffClickID,  
ResIDT inTransitionID );
```

Parameters This method has the following parameters:

ResIDT	inOnID	Resource ID for the “on” graphic.
ResIDT	inOnClickID	Resource ID for the “clicked when on” graphic.

ResIDT	inOffID	Resource ID for the “off” graphic.
ResIDT	inOffClickID	Resource ID for the “clicked when off” graphic.
ResIDT	inTransitionID	Resource ID for the transition graphic.

Return None

SetGraphicsType()

Purpose	This method sets the value of the mGraphicsType data member.
Access	Virtual, Public
Prototype	<code>virtual void SetGraphicsType(OSType inGraphicsType);</code>
Parameters	This method has the following parameter:

OSType	inGraphicsType	The value to set for mGraphicsType .
--------	----------------	--

SetValue()

Purpose	This method is an override of the base class method SetValue() in LControl . It sets the value of the button to on or off.
---------	--

mGraphicsType

Purpose	This method holds the resource type descriptor for the graphic elements of the button. It should be one of the following: <ul style="list-style-type: none">• ICN#
---------	--

LToggleButton

- ICON'
- 'PICT'

Access Protected

Prototype OSType mGraphicsType;

mOnID

Purpose This data member holds the resource ID for the “on” button graphic.

Access Protected

Prototype ResIDT mOnID;

mOnClickID

Purpose This data member holds the resource ID for the “clicked when on” button graphic.

Access Protected

Prototype ResIDT mOnClickID;

mOffID

Purpose This data member holds the resource ID for the “off” button graphic.

Access Protected

Prototype ResIDT mOffID;

mOffClickID

Purpose	This data member holds the resource ID for the “clicked when off” button graphic.
Access	Protected
Prototype	ResIDT mOffClickID;

mTransitionID

Purpose	This data member holds the resource ID for the transition button graphic.
Access	Protected
Prototype	ResIDT mTransitionID;

LUDPEndpoint

Overview	LUDPEndpoint is a PowerPlant class that is used for implementing Internet UDP endpoints. UDP is the sessionless protocol of the Internet, and is used by a handful of protocols such as NTP (network time protocol).
Methods	The methods in this class are: LUDPEndpoint() ReceiveFrom() ~LUDPEndpoint() SendPacketData()
Data Members	There are no data members in this class.
Operation	In PowerPlant, there are two subclasses of LUDPEndpoint, named LMacUDPEndpoint and LOpenTptUDPEndpoint. The appropriate class is created automatically when you call UNetworkFactory::CreateUDPEndpoint() .
Source files	(Networking Classes) LUDPEndpoint.h LUDPEndpoint.cp
See also	LEndpoint LMacTCPUDPEndpoint LOpenTptUDPEndpoint

LUDPEndpoint()

Purpose	The constructor creates the object.
Access	Public
Prototype	<code>LUDPEndpoint()</code> ;
Parameters	None

LUDPEndpoint

~LUDPEndpoint()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>~LUDPEndpoint() ;</code>

ReceiveFrom()

Purpose	Receive UDP datagrams.	
Access	Pure virtual, Public	
Prototype	<pre>void ReceiveFrom(LInternetAddress& outRemoteAddress, void* outDataBuffer, UInt32& ioDataSize, UInt32 inTimeoutTicks = Timeout_None) = 0;</pre>	
Parameters	The parameter for these constructors is:	
	LInternetAddress	The pointer to the thread.
	void*	The buffer.
	UInt32&	The size of the data.
	UInt32	The time to wait. The default is Timeout_None.
Return	None	

SendPacketData()

Purpose	Send UDP datagrams.
---------	---------------------

Access	Pure virtual, Public									
Prototype	<pre>void SendPacketData(LInternetAddress& inRemoteHost, void* inData, UInt32 inDataSize) = 0;</pre>									
Parameters	<p>The parameter for these constructors is:</p> <table><tr><td>LIternetAddres s</td><td>inRemoteHost</td><td>The address.</td></tr><tr><td>void*</td><td>inData</td><td>The buffer.</td></tr><tr><td>UInt32&</td><td>inDataSize</td><td>The size of the data.</td></tr></table>	LIternetAddres s	inRemoteHost	The address.	void*	inData	The buffer.	UInt32&	inDataSize	The size of the data.
LIternetAddres s	inRemoteHost	The address.								
void*	inData	The buffer.								
UInt32&	inDataSize	The size of the data.								
Return	None									

LUndoer

Overview	LUndoer is a PowerPlant class that helps manage the do, undo, and redo operations. It works in conjunction with LAction.
Methods	<p>The methods in this class are:</p> <div><div>LUndoer() ExecuteSelf() PostAction() operator=()</div><div>~LUndoer() FindUndoStatus() ToggleAction()</div></div>
Data Members	<p>The data members in this class are:</p> <div>mAction</div>
Operation	In a typical application having a single-level undo, you should not have to call the LUndoer member functions directly.
Source files	<p>(Action Classes)</p> <div>LUndoer.h LUndoer.cp</div>
Ancestors	LAttachment
See Also	<div>LAction LAttachment LCommander</div>

LUndoer()

Purpose	The default constructor initializes the data member. The copy constructor provides a definition so that the compiler does not supply a default copy constructor for this object.
Access	Public (default constructor) and Private (copy constructor).

LUndoer

Prototype	<pre>LUndoer(); /* default constructor */ LUndoer(const LUndoer &inOriginal); /* pvt copy */</pre>
Parameters	None

~LUndoer()

Purpose	The destructor destroys the object.
Access	Public, Virtual
Prototype	<pre>virtual ~LUndoer();</pre>
Parameters	None

ExecuteSelf()

Purpose	This method intercepts messages for the host. When called with a <code>msg_PostAction</code> message it calls SetExecuteHost() with the new action to perform. When called with a <code>msg_CommandStatus</code> message it enables and sets the text for the Undo menu item using FindUndoStatus() . When called with a <code>cmd_Undo</code> message it toggles betweenUndoing/Redoing the Action using ToggleAction() . If called with any other message, SetExecuteHost() is called with a value of <code>true</code> .	
Access	Virtual, Protected	
Prototype	<pre>virtual void ExecuteSelf(MessageT inMessage, void *ioParam);</pre>	
Parameters	This method has the following parameters:	
	<code>MessageT inMessage</code>	This is the message that is being received.
	<code>void* ioParam</code>	This is a pointer to a parameter that accompanies the message.

Return	None
--------	------

FindUndoStatus()

Purpose	This method will enable/disable and set the text for the Undo menu item. The menu item can be set to the Redo string, Undo string, or Can't Undo string depending on whether mAction is undoable or redoable.				
Access	Virtual, Protected				
Prototype	<code>virtual void FindUndoStatus(SCommandStatus *ioStatus);</code>				
Parameters	This method has the following parameter: <table><tr><td><code>SCommandStatus*</code></td><td><code>ioStatus</code></td><td>This is a pointer to a structure containing status information for the command.</td></tr></table>		<code>SCommandStatus*</code>	<code>ioStatus</code>	This is a pointer to a structure containing status information for the command.
<code>SCommandStatus*</code>	<code>ioStatus</code>	This is a pointer to a structure containing status information for the command.			
Return	None				

PostAction()

Purpose	This method handles a new Action that is posted.				
Access	Virtual, Protected				
Prototype	<code>virtual void PostAction(LAction *inAction);</code>				
Parameters	This method has the following parameter: <table><tr><td><code>LAction*</code></td><td><code>inAction</code></td><td>This is a pointer to LAction object that indicates an Action to be taken.</td></tr></table>		<code>LAction*</code>	<code>inAction</code>	This is a pointer to LAction object that indicates an Action to be taken.
<code>LAction*</code>	<code>inAction</code>	This is a pointer to LAction object that indicates an Action to be taken.			
Return	None				

ToggleAction()

Purpose	Undo/Redo the Action associated with this Undoer, held in the member mAction .
Access	Virtual, Protected
Prototype	<code>virtual void ToggleAction();</code>
Parameters	None
Return	None

operator=()

Purpose	Overloads the assignment operator so that the compiler will not substitute a default.					
Access	Private					
Prototype	LUndoer& operator=(const LUndoer &inOriginal);					
Parameters	This operator takes the following parameter: <table><tr><td>const LUndoer&</td><td>inOriginal</td><td>A reference to an LUndoer object.</td></tr></table>			const LUndoer&	inOriginal	A reference to an LUndoer object.
const LUndoer&	inOriginal	A reference to an LUndoer object.				
Return	Returns a reference to an LUndoer object.					

mAction

Purpose	This data member stores a pointer to the action object that encapsulates the user's most recent action. Because the Undoer ot can send messages to the action object telling it to redo or undo itself.
Access	Protected
Prototype	<code>LAction *mAction;</code>

LVariableArray

Overview	LVariableArray is a PowerPlant class that is similar to LArray , but allows you to store data of differing element sizes.	
Methods	The methods in this class are:	
	LVariableArray()	~LVariableArray()
	AdjustAllocation()	AdjustStorage()
	AssignItemsAt()	GetItemOffset()
	GetItemPtr()	GetItemSize()
	GetOffsetsHandle()	GrabItemRangeSize()
	GrabItemSize()	InternalAdjustAllocation()
	InternalCopyItem()	PokeItem()
	ShiftItems()	Sort()
	StoreNewItems()	
Data Members	The data members in this class are:	
	mItemOffsetsH	mItemsAllocated
Operation	This class overrides several member functions of LArray to implement data storage and retrieval in a situation where array elements vary in size.	
	The public interface and usage is mostly identical to that of LArray.	
Source files	(Array Classes)	
	LVariableArray.h	
	LVariableArray.cp	
Ancestors	LArray	
See Also	LArrayIterator	
	LComparator	
	LLockedArrayIterator	

LVariableArray

LVariableArray()

Purpose	The constructor creates objects from the passed-in parameters.		
Access	Public		
Prototype	<pre>LVariableArray(LComparator *inComparator, Boolean inKeepSorted); LVariableArray(Handle inItemsHandle, ArrayOffsetH inOffsetsHandle, LComparator *inComparator, Boolean inIsSorted, Boolean inKeepSorted);</pre>		
Parameters	The parameters for these constructors are:		
	LComparator*	inComparator	The pointer to the comparator object to use.
	Boolean	inKeepSorted	Whether to keep the array sorted or not.
	Handle	inItemsHandle	A handle to the items to put into the array.
	ArrayOffsetH	inOffsetsHandle	An offset. It is assumed that the number of items for the array is the same as the number of offsets in the handle.
	Boolean	inIsSorted	Whether the elements are sorted or not.

~LVariableArray()

Purpose	The destructor destroys the object.
Access	Public

Prototype `virtual ~LVariableArray();`

AdjustAllocation()

Purpose Adjust the size of the Handle used to store array items. This is an override of [AdjustAllocation\(\)](#) in [LArray](#).

AdjustStorage()

Purpose Called internally when the number of bytes used by Items in the array changes. This is an override of [AdjustStorage\(\)](#) in [LArray](#).

AssignItemsAt()

Purpose Assign the same value to items starting at the specified index. This is an override of [AdjustStorage\(\)](#) in [LArray](#).

GetItemOffset()

Purpose This method returns the offset into the array for a given item.

Access Protected

Prototype `UInt32 GetItemOffset(ArrayIndexT inAtIndex);`

Parameters This method has the following parameter:

ArrayIndexT	inAtIndex	The index to calculate an offset for.
-------------	-----------	---------------------------------------

Return UInt32 containing the offset.

GetItemPtr()

Purpose	Return a pointer to the start of an item's data within the internal storage Handle. This method performs no error checking on the index.			
Access	Virtual, Public			
Prototype	<code>virtual void* GetItemPtr(ArrayIndexT inIndex) const;</code>			
Parameters	<div>This method has the following parameter:<table><tr><td>ArrayIndexT</td><td>inIndex</td><td>The index for the item.</td></tr></table></div>	ArrayIndexT	inIndex	The index for the item.
ArrayIndexT	inIndex	The index for the item.		
Return	The pointer to the array item.			
Remarks	WARNING: Unless you have called Lock() , the returned pointer points to data within a relocatable Handle block.			

GetItemSize()

Purpose	Return the size in bytes of an item in the array.			
Access	Virtual, Public			
Prototype	<pre>virtual UInt32 GetItemSize(ArrayIndexT inIndex) const;</pre>			
Parameters	<div>This method has the following parameter:</div> <table><tr><td>ArrayIndexT</td><td>inIndex</td><td>The index to get an item.</td></tr></table>	ArrayIndexT	inIndex	The index to get an item.
ArrayIndexT	inIndex	The index to get an item.		
Return	UInt32 containing the size in bytes of an item.			

GetOffsetsHandle()

Purpose	Return Handle used to store offsets to the data for array items. This is an accessor method for mItemOffsetsH .
Access	Public
Prototype	<code>Handle GetOffsetsHandle() const;</code>
Parameters	None
Return	The handle used to store offsets (mItemOffsetsH).
Remarks	<ul style="list-style-type: none">• Treat the Handle as read-only. Changing the data in the Handle could invalidate the internal state of the array.• In general, you should only access the offsets Handle in order to copy it (either in memory or by writing it to a file).

GrabItemRangeSize()

Purpose	Return the size in bytes of the items from <code>inStartIndex</code> to <code>inEndIndex</code> , inclusive. This is an override of GrabItemRangeSize() in LArray .
---------	---

GrabItemSize()

Purpose	Return the size in bytes of the specified item. This method is the same as GetItemSize() except it doesn't validate <code>inIndex</code> . This is an override of GrabItemSize() in LArray .
---------	--

InternalAdjustAllocation()

Purpose	Called internally to change the size of the storage used. This is an override of InternalAdjustAllocation() in LArray .
---------	---

LVariableArray

InternalCopyItem()

Purpose Set the item at `inDestIndex` to a copy of the item at `inSourceIndex`. This is an override of [InternalCopyItem\(\)](#) in [LArray](#).

PokeItem()

Purpose Store data for the item at the specified index. This is an override of [PokeItem\(\)](#) in [LArray](#).

ShiftItems()

Purpose Moves items within the Handle used for internal storage. This is an override of [ShiftItems\(\)](#) in [LArray](#).

Sort()

Purpose Sort items in the array. This is an override of [Sort\(\)](#) in [LArray](#).

StoreNewItem()

Purpose Store values within the internal storage Handle. Items all have the same value, and space must already have been allocated for them. This is an override of [StoreNewItem\(\)](#) in [LArray](#).

mItemOffsetsH

Purpose	This stores the handle for the offsets.
Access	Protected
Prototype	<code>ArrayOfsetH mItemOffsetsH;</code>

mItemsAllocated

Purpose	This stores the items allocated information.
Access	Protected
Prototype	<code>UInt32 mItemsAllocated;</code>

LView

Overview LView is a PowerPlant class that is the basis for the visual hierarchy. An LView object is a pane that can contain other panes.

Methods The methods in this class are:

<u>LView()</u>	<u>~LView()</u>
<u>Activate()</u>	<u>AdaptToNewSurroundings()</u>
<u>AdaptToSuperFrameSize()</u>	<u>AddSubPane()</u>
<u>AdjustCursor()</u>	<u>AutoScrollImage()</u>
<u>CalcPortOrigin()</u>	<u>CalcRevealedRect()</u>
<u>Click()</u>	<u>CountPanels()</u>
<u>Deactivate()</u>	<u>DeleteAllSubPanels()</u>
<u>Disable()</u>	<u>DontRefresh()</u>
<u>Draw()</u>	<u>Enable()</u>
<u>EstablishPort()</u>	<u>ExpandSubPane()</u>
<u>FindConstPaneByID()</u>	<u>FindDeepSubPaneContaining()</u>
<u>FindPaneByID()</u>	<u>FindShallowSubPaneContaining()</u>
<u>FinishCreate()</u>	<u>FocusDraw()</u>
<u>FocusDraw()</u>	<u>FocusExposed()</u>
<u>GetDescriptorForPaneID()</u>	<u>GetImageLocation()</u>
<u>GetDescriptorForPaneID()</u>	<u>GetImageLocation()</u>
<u>GetImageSize()</u>	<u>GetInFocusView()</u>
<u>GetLocalUpdateRgn()</u>	<u>GetPortOrigin()</u>
<u>GetRevealedRect()</u>	<u>GetScrollPosition()</u>
<u>GetScrollUnit()</u>	<u>GetSubPanels()</u>
<u>GetValueForPaneID()</u>	<u>Hide()</u>
<u>ImagePointIsInFrame()</u>	<u>ImageRectIntersectsFrame()</u>

<u>ImageToLocalPoint()</u>	<u>InitView()</u>
<u>LocalToImagePoint()</u>	<u>LocalToPortPoint()</u>
<u>MoveBy()</u>	<u>OrientAllSubPanels()</u>
<u>OrientSubPane()</u>	<u>OutOfFocus()</u>
<u>PortToLocalPoint()</u>	<u>PrintPanel()</u>
<u>ReconcileFrameAndImage()</u>	<u>Refresh()</u>
<u>RemoveSubPane()</u>	<u>ResizeFrameBy()</u>
<u>ResizeImageBy()</u>	<u>ResizeImageTo()</u>
<u>RestorePlace()</u>	<u>SavePlace()</u>
<u>ScrollBits()</u>	<u>ScrollImageBy()</u>
<u>ScrollImageTo()</u>	<u>ScrollPinnedImageBy()</u>
<u>ScrollToPanel()</u>	<u>SetDescriptorForPanelID()</u>
<u>SetReconcileOverhang()</u>	<u>SetScrollUnit()</u>
<u>SetValueForPanelID()</u>	<u>Show()</u>
<u>SubImageChanged()</u>	<u>SuperActivate()</u>
<u>SuperDeactivate()</u>	<u>SuperDisable()</u>
<u>SuperEnable()</u>	<u>SuperHide()</u>
<u>SuperPrintPanel()</u>	<u>SuperShow()</u>

Data Members The data members in this class are:

<u>sInFocusView</u>	<u>mImageSize</u>
<u>mImageLocation</u>	<u>mScrollUnit</u>
<u>mPortOrigin</u>	<u>mSubPanels</u>
<u>mRevealedRect</u>	<u>mUpdateRgnH</u>
<u>mReconcileOverhang</u>	

Operation LView objects contain the panes that you draw. In other words, LView is a container object, with the panes being the contents. Objects that derive from LView are usually places where you draw things.

To learn more about views, refer to *The PowerPlant Book*.

Source files (Pane Classes)

LView.h

LView.cp

Ancestors [LAttachable](#)

[LPane](#)

LView()

Purpose The constructors create objects from the passed-in parameters.

Access Public

Prototype

```
LView();
LView( const LView &inOriginal );
LView( const SPaneInfo &inPaneInfo,
const SViewInfo &inViewInfo );
LView( LStream *inStream );
```

Parameters The parameters for the constructors are:

const LView&	inOriginal	The LView object to copy.
const SPaneInfo&	inPaneInfo	The reference to the Superpane.
const SViewInfo&	inViewInfo	The reference to the Superview.
LStream*	inStream	The stream to construct with.

LView

~LView()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~LView();</code>

Activate()

Purpose	Activate a View. This is an override of Activate() in LPane .
---------	---

AdaptToNewSurroundings()

Purpose	Adjust view when its SuperView changes identity or size. This is an override of AdaptToNewSurroundings() in LPane .
---------	---

AdaptToSuperFrameSize()

Purpose	Adjust state of View when size of SuperView's Frame changes by the specified amounts. This is an override of AdaptToSuperFrameSize() in LPane .
---------	---

AddSubPane()

Purpose	You should not normally call this function. Call LPane::PutInside() to associate a Pane with a SuperView.
Access	Virtual, Public
Prototype	<code>virtual void AddSubPane(LPane *inSub);</code>

Parameters	The parameter for this method is:		
	LPane*	inSub	The pointer to the sub pane to add.
Return	None		

AdjustCursor()

Purpose	Wrapper function for setting the cursor shape. View needs to determine which subpane, if any, should set the cursor shape. This is an override of AdjustCursor() in LPane .
---------	---

AutoScrollImage()

Purpose	Scroll the image if the specified point is outside the View's Frame. Call this function while tracking the mouse to scroll a View in the direction of the mouse location.
Access	Virtual, Public
Prototype	<code>virtual Boolean AutoScrollImage(Point inLocalPt);</code>
Parameters	The parameter for this method is:
	Point inLocalPt The pointer to the sub pane to add.
Return	Returns true if the View actually scrolled, else false.

CalcPortOrigin()

Purpose	Calculate the coordinate origin for the Port needed to set up the local coordinates of a View.
---------	--

Access	Virtual, Public
Prototype	<code>virtual void CalcPortOrigin();</code>
Parameters	None
Return	None
Remarks	The Port origin must be in 16-bit space. In fact, the limitation is more restrictive because the entire Port must be in 16-bit space. The origin is the top left corner. To make sure that the bottom right corner is in 16-bit space, we force the origin to be less than 2^{14} (16,384), which allows Port dimensions of a maximum of 16,384 pixels. At 72 dpi, this is about 227 inches or 19 feet (much more screen or printer area than you can get with current technology).

This means that Local and Image coordinates will be the same when the Image size is less than 16,384 pixels. For Images greater than this, you can't use absolute coordinates for drawing. You need to offset the coordinates using [ImageToLocalPoint\(\)](#) and [LocalToImagePoint\(\)](#).

The true coordinates offset is the distance between the top left corners of the Image and the Port. If this offset is greater than 2^{14} , we use an effective offset that is the true offset modulo 2^{14} :

$$\text{effective_offset} = \text{true_offset} \text{ modulo } 2^{14}$$

Using this effective offset maintains the bit-wise alignment of the Port with respect to base-2 byte boundaries. This is important for drawing Toolbox Patterns and PixPats, as well as for `CopyBits()` calls.

CalcRevealedRect()

Purpose	Calculate the portion of the Frame which is revealed through the frames of all SuperViews. mRevealedRect is in Port coordinates.
Access	Public
Prototype	<code>void CalcRevealedRect();</code>
Parameters	None

Return None

Click()

Purpose Handle a click inside a View. This is an override of [Click\(\)](#) in [LPane](#).

CountPanels()

Purpose Return the number of horizontal and vertical Panels. A Panel is a "frameful" of a View's Image. This is an override of [CountPanels\(\)](#) in [LPane](#).

Deactivate()

Purpose Deactivate a View. This is an override of [Deactivate\(\)](#) in [LPane](#).

DeleteAllSubPanels()

Purpose Deletes all subpanes of a View.

Access Public

Prototype `void DeleteAllSubPanels () ;`

Parameters None

Return None

LView

Disable()

Purpose Disable a View. This is an override of [Disable\(\)](#) in [LPane](#).

DontRefresh()

Purpose Validate the area occupied by a View. This removes the View area from the update region so that the View won't be redrawn during the next Update event.

This is an override of [DontRefresh\(\)](#) in [LPane](#).

Draw()

Purpose Draw a View and all its subpanes. This is an override of [Draw\(\)](#) in [LPane](#).

Enable()

Purpose Enable a View. This is an override of [Enable\(\)](#) in [LPane](#).

EstablishPort()

Purpose Set the GrafPort for a View.

Access Virtual, Public

Prototype `virtual Boolean EstablishPort();`

Parameters None

Return Return `true` if the port was set successfully, else `false`.

ExpandSubPane()

Purpose	Adjust the size and location of a SubPane to fit within the interior of a View in the horizontal and/or vertical directions.	
Access	Virtual, Public	
Prototype	<pre>virtual void ExpandSubPane(LPane *inSub, Boolean inExpandHoriz, Boolean inExpandVert);</pre>	
Parameters	The parameters for this method are:	
	<code>LPane* inSub</code>	The pointer to the sub pane to add.
	<code>Boolean inExpandHoriz</code>	Tell whether to expand horizontally.
	<code>Boolean inExpandVert</code>	Tell whether to expand vertically.
Return	None	

FindConstPaneByID()

Purpose	Find the Pane of a View which has the specified ID. This is an override of FindConstPaneByID() in LPane .
---------	---

FindDeepSubPaneContaining()

Purpose	Return the most deeply nested visible SubPane which contains the specified point, which is in Port coordinates. This is an override of FindDeepSubPaneContaining() in LPane .
---------	---

FindPaneByID()

Purpose Find the Pane of a View which has the specified ID. This is an override of [FindPaneByID\(\)](#) in [LPane](#).

FindShallowSubPaneContaining()

Purpose Return the immediate visible SubPane which contains the specified point, which is in Port coordinates. This is an override of [FindShallowSubPaneContaining\(\)](#) in [LPane](#).

FindSubPaneHitBy()

Purpose Find the SubPane of this View that is hit by the specified point. This is an override of [FindSubPaneHitBy\(\)](#) in [LPane](#).

FinishCreate()

Purpose Wrapper function for FinishCreateSelf. This is an override of [FinishCreate\(\)](#) in [LPane](#).

You will rarely want to override this function.

FocusDraw()

Purpose Prepare for drawing in the View by setting the Port and clipping area. This is an override of [FocusDraw\(\)](#) in [LPane](#).

FocusExposed()

Purpose Prepare for drawing in the View by setting the Port and clipping area. This is an override of [FocusExposed\(\)](#) in [LPane](#).

GetDescriptorForPaneID()

Purpose Get the Descriptor of the Pane of a View which has the specified ID. The Pane may be the View itself, or a subpane of the View.

Access Virtual, Public

Prototype

```
virtual StringPtr GetDescriptorForPaneID(
    PaneIDT inPaneID,
    Str255 outDescriptor) const;
```

Parameters The parameters for this method are:

PaneIDT	inPaneID	The resource ID for the pane.
Str255	outDescriptor	The string that we want, the descriptor.

Return Returns a pointer that is the same as outDescriptor.

GetImageLocation()

Purpose Determine the location of the image within a View.

Access Public

Prototype

```
void GetImageLocation(SPoint32 &outLocation)
const;
```

Parameters The parameter for this method is:

LView

	<div><div>SPoint32&</div><div>outLocation</div><div>The location of the image within the view.</div></div>
Return	None

GetImageSize()

Purpose	Pass back the dimensions of a View's image.		
Access	Virtual, Public		
Prototype	<pre>virtual void GetImageSize(SDimension32 &outSize) const;</pre>		
Parameters	The parameter for this method is: <table><tr><td><div><div>SDimension32</div><div>&</div><div>outSize</div></div></td><td>The dimensions of the image within the view.</td></tr></table>	<div><div>SDimension32</div><div>&</div><div>outSize</div></div>	The dimensions of the image within the view.
<div><div>SDimension32</div><div>&</div><div>outSize</div></div>	The dimensions of the image within the view.		
Return	None		

GetInFocusView()

Purpose	Retrieve a pointer to the view that has the focus. This is an accessor method for the sInFocusView data member.
Access	Static, Public
Prototype	<pre>static LView* GetInFocusView();</pre>
Parameters	None
Return	The pointer to the LView that has the focus.

GetLocalUpdateRgn()

Purpose Return the region, in local coordinates, to which update drawing is clipped. This is an override of [GetLocalUpdateRgn\(\)](#) in [LPane](#).

GetPortOrigin()

Purpose Pass back the coordinate origin for the Port needed to set up the local coordinates of a View.

Access Virtual, Public

Prototype `virtual void GetPortOrigin(Point &outOrigin)
const;`

Parameters The parameter for this method is:

<code>Point&</code>	<code>outOrigin</code>	The coordinate origin passed back to the caller.
-------------------------	------------------------	--

Return None

GetRevealedRect()

Purpose Retrieve the Rect from [mRevealedRect](#).

Access Public

Prototype `void GetRevealedRect(Rect &outRect) const;`

Parameters The parameter for this method is:

<code>Rect&</code>	<code>outRect</code>	The Rect in mRevealedRect is passed back to the caller.
------------------------	----------------------	---

Return None

GetScrollPosition()

Purpose	Pass back the location of a View's frame within its image.	
Access	Virtual, Public	
Prototype	<code>virtual void GetScrollPosition(SPoint32 &outScrollPosition) const;</code>	
Parameters	The parameter for this method is:	
	<code>SPoint32& outScrollPosition</code>	The location of the View's frame.
Return	None	

GetScrollUnit()

Purpose	This method retrieves the value of mScrollUnit .	
Access	Public	
Prototype	<code>void GetScrollUnit(SPoint32 &outScrollUnit) const;</code>	
Parameters	The parameter for this method is:	
	<code>SPoint32& outScrollUnit</code>	The value of mScrollUnit .
Return	None	

GetSubPanels()

Purpose	This method returns the value of mSubPanels .
Access	Public

Prototype	<code>TArray<LPane*>& GetSubPanes () ;</code>
Parameters	None
Return	A reference to an array that contains mSubPanes .

GetValueForPaneID()

Purpose	Get the Value of the Pane of a View which has the specified ID. The Pane may be the View itself, or a subpane of the View.			
Access	Virtual, Public			
Prototype	<code>virtual SInt32 GetValueForPaneID(PaneIDT inPaneID) const;</code>			
Parameters	<div>The parameter for this method is:</div> <table><tr><td>PaneIDT</td><td>inPaneID</td><td>The pane ID to query the value of.</td></tr></table>	PaneIDT	inPaneID	The pane ID to query the value of.
PaneIDT	inPaneID	The pane ID to query the value of.		
Return	Returns the value in an <code>SInt32</code> .			

Hide()

Purpose	Make a View invisible. This is an override of Hide() in LPane .
---------	---

ImagePointIsInFrame()

Purpose	Return whether a Point specified in Image Coords is within the Frame of a View.
Access	Public
Prototype	<code>Boolean ImagePointIsInFrame(SInt32 inHorizImage, SInt32 inVertImage) const;</code>

LView

Parameters	The parameters for this method are:		
	SInt32	inHorizImage	The horizontal coordinate.
	SInt32	inVertImage	The vertical coordinate.
Return	Boolean indicating whether the point is in the frame (<code>true</code>) or not (<code>false</code>).		

ImageRectIntersectsFrame()

Purpose	Determine whether a rectangle specified in image coordinates intersects the frame of a View.		
Access	Public		
Prototype	<pre>Boolean ImageRectIntersectsFrame(SInt32 inLeftImage, SInt32 inTopImage, SInt32 inRightImage, SInt32 inBottomImage) const;</pre>		
Parameters	The parameters for this method are:		
	SInt32	inLeftImage	The left coordinate.
	SInt32	inTopImage	The top coordinate.
	SInt32	inRightImage	The right coordinate.
	SInt32	inBottomImage	The bottom coordinate.
Return	Boolean indicating whether the rectangle is in the frame (<code>true</code>) or not (<code>false</code>).		

ImageToLocalPoint()

Purpose	Convert point from Image (32-bit) to Local (16-bit) coordinates.
---------	--

Image and Local coordinates are different only when the Image size is greater than 16K (15-bit).

Access	Public						
Prototype	<pre>void ImageToLocalPoint(const SPoint32 &inImagePt, Point &outLocalPt) const;</pre>						
Parameters	<div>The parameters for this method are:</div> <table><tr><td>const SPoint32&</td><td>inImagePt</td><td>The point to convert.</td></tr><tr><td>Point&</td><td>outLocalPt</td><td>The converted point.</td></tr></table>	const SPoint32&	inImagePt	The point to convert.	Point&	outLocalPt	The converted point.
const SPoint32&	inImagePt	The point to convert.					
Point&	outLocalPt	The converted point.					
Return	None						

InitView()

Purpose	Initializer for the class.		
Access	Protected		
Prototype	void InitView(const SViewInfo &inViewInfo);		
Parameters	The parameter for this method is:		
	const SViewInfo &	inViewInfo	The reference to the Superview to initialize.
Return	None		

LocalToImagePoint()

Purpose Convert point from Local (16-bit) to Image (32-bit) coordinates.

Image and Local coordinates are different only when the Image size is greater than 16K (15-bit)

LView

Access	Public	
Prototype	<pre>void LocalToImagePoint(const Point &inLocalPt, SPoint32 &outImagePt) const;</pre>	
Parameters	The parameters for this method are:	
	<pre>const Point& inLocalPt</pre>	The point to convert.
	<pre>SPoint32& outImagePt</pre>	The converted point.
Return	None	

LocalToPortPoint()

Purpose	Convert point from Local to Port coordinates. This is an override of LocalToPortPoint() in LPane .
---------	--

MoveBy()

Purpose	Move the location of the Image and Frame by the specified amounts. This is an override of MoveBy() in LPane .
---------	---

OrientAllSubPanels()

Purpose	<p>Adjust the Enabled, Active, and Visible properties of all subpanes.</p> <p>This function is for the convenience of people who have code from CW5 (PowerPlant 1.0.2) and earlier which created Panes directly in the code (not from 'PPob' resources). You really should call FinishCreate() for each Pane, but to quickly get code running, you can call this function after creating several Panes inside one View.</p> <p>Do NOT use this function for new code.</p>
---------	---

Access	Public
Prototype	<code>void OrientAllSubPanels();</code>
Parameters	None
Return	None

OrientSubPane()

Purpose	Adjust the Enabled, Active, and Visible properties of a subpane based on the properties of this View (which must be the subpane's SuperView).		
Access	Virtual, Public		
Prototype	<code>virtual void OrientSubPane(LPane *inSub);</code>		
Parameters	The parameter for this method is:		
	<code>LPane*</code>	<code>inSub</code>	The pointer to the subpane.
Return	None		

OutOfFocus()

Purpose	<p>Notify the View system that a View is no longer in focus.</p> <p>Call when the coordinate system or clipping region of a View changes, passing a pointer to that View. This clears the saved focus if that View was the one in focus.</p> <p>Use nil for inView if you manually change (and don't restore) the current port or clipping region</p>		
Access	Static, Public		
Prototype	<code>static void OutOfFocus(LView *inView);</code>		
Parameters	The parameter for this method is:		

LView

LView*	inView	The pointer to the view that is no longer in focus.
--------	--------	---

Return	None
--------	------

PortToLocalPoint()

Purpose	Convert a point from Port to local coordinates. This is an override of PortToLocalPoint() in LPane .
---------	--

PrintPanel()

Purpose	Try to print a panel of a View. The View is at the top level of the Printout, meaning that it controls pagination. This functions scrolls the View to the specified panel.
---------	--

This is an override of [PrintPanel\(\)](#) in [LPane](#).

ReconcileFrameAndImage()

Purpose	Adjusts the Image so that it fits within the Frame. This function addresses the problem of what to do when you scroll a View to at or near the bottom or right, then make the View's frame larger. This would normally expose some undefined area below or to the right of the image.
---------	---

If [mReconcileOverhang](#) is true, this function scrolls the image so that the bottom right corner is at the bottom right of the frame. However, it never moves the top left corner of the image beyond the top left of the frame. Therefore, the only time undefined area is exposed is when the frame is larger than the Image.

For Views with fixed image sizes, such as drawings where the Image size is the size of a printed page, set [mReconcileOverhang](#) to true. The user does not normally want to see past the bottom or right of such Views.

For Views with variable Image sizes, such as text blocks where the size of the Image depends on the number of lines of text, set [mReconcileOverhang](#) to false. The user may want to see the undefined area in anticipation of the image growing.

Access	Virtual, Protected
Prototype	<code>virtual void ReconcileFrameAndImage(Boolean inRefresh);</code>
Parameters	<div>The parameter for this method is:</div> <div><div><div>Boolean</div><div>inRefresh</div></div><div>The value indicating whether to refresh.</div></div>
Return	None

Refresh()

Purpose	Invalidate the area occupied by a View. This forces an Update event that, when processed, will redraw the View. Refresh does nothing if the View is not visible. This is an override of Refresh() in LPane .
---------	--

RemoveSubPane()

Purpose	You should not normally call this function. Call <code>LPane::PutInside(nil)</code> to remove a SubPane from its SuperView
Access	Virtual, Public
Prototype	<code>virtual void RemoveSubPane(LPane *inSub);</code>
Parameters	The parameter for this method is:

	<div><div>LPane</div><div>inSub</div><div>The value indicating whether to refresh.</div></div> <div>*</div>
Return	None

ResizeFrameBy()

Purpose Change the Frame size by the specified amounts. This is an override of [ResizeFrameBy\(\)](#) in [LPane](#).

ResizeImageBy()

Purpose Change the image size by the specified pixel increments.

Access Public

Prototype virtual void ResizeImageBy(
 SInt32 inWidthDelta,
 SInt32 inHeightDelta,
 Boolean inRefresh);

Parameters The parameters for this method are:

SInt32	inWidthDelta	The value indicating the change in width.
SInt32	inHeightDelta	The value indicating the change in height.
Boolean	inRefresh	Whether to refresh or not.

Return None

ResizeImageTo()

Purpose	Set the image size to the specified pixel dimensions.	
Access	Public	
Prototype	<pre>void ResizeImageTo(SInt32 inWidth, SInt32 inHeight, Boolean inRefresh);</pre>	
Parameters	The parameters for this method are:	
	SInt32 inWidth	The value indicating the change in width.
	SInt32 inHeight	The value indicating the change in height.
	Boolean inRefresh	Whether to refresh or not.
Return	None	

RestorePlace()

Purpose	Read size and location information stored in a Stream by the SavePlace() function. This is an override of RestorePlace() in LPane .
---------	---

SavePlace()

Purpose	Write size and location information to a Stream for later retrieval by the RestorePlace() function. This is an override of SavePlace() in LPane .
---------	---

ScrollBits()

Purpose	Scroll the pixels of a View. This method is called internally by ScrollImageBy() to shift the pixels.		
Access	Virtual, Public		
Prototype	<pre>virtual void ScrollBits(SInt32 inLeftDelta, SInt32 inTopDelta) ;</pre>		
Parameters	The parameters for this method are:		
	SInt32 2	inLeftDelta a	The value indicating the change from the left.
	SInt32 2	inTopDelta	The value indicating the change from the top.
Return	None		

ScrollImageBy()

Purpose	<p>Scroll image by specified horizontal and vertical increments. Scrolling moves the image relative to the frame and port, so that a different portion of the image is visible thru the frame.</p> <p>Positive deltas scroll right and down. Negative deltas scroll left and up.</p> <p>If inRefresh is true, the Port containing the View is updated immediately, rather than refreshed at the next update event. Scrolling usually happens during mouse down tracking, so we want immediate visual feedback.</p>		
Access	Public		
Prototype	<pre>virtual void ScrollImageBy(SInt32 inLeftDelta,</pre>		

```
SInt32 inTopDelta,
Boolean inRefresh );
```

Parameters The parameters for this method are:

SInt32	inLeftDelta	The value indicating the change from the left.
SInt32	inTopDelta	The value indicating the change from the top.
Boolean	inRefresh	Whether to refresh or not.

Return None

ScrollImageTo()

Purpose Scroll image to the specified horizontal and vertical locations.

Scrolling moves the image relative to the frame and port, so that a different portion of the image is visible thru the frame.

When scrolled to (0, 0), the top left of the image coincides with the top left of the frame (home position).

If `inRefresh` is true, the Port containing the View is updated immediately, rather than refreshed at the next update event. Scrolling usually happens during mouse down tracking, so we want immediate visual feedback.

Access Public

Prototype

```
void ScrollImageTo(
SInt32 inLeftLocation,
SInt32 inTopLocation,
Boolean inRefresh );
```

Parameters The parameters for this method are:

LView

	SInt32	inLeftLocation	The value indicating the point on the left.
	SInt32	inTopLocation	The value indicating the on the top.
	Boolean	inRefresh	Whether to refresh or not.
Return	None		

ScrollPinnedImageBy()

Purpose	Scroll image by specified horizontal and vertical increments, but don't scroll beyond an edge of the frame.		
Access	Public		
Prototype	virtual Boolean ScrollPinnedImageBy(SInt32 inLeftDelta, SInt32 inTopDelta, Boolean inRefresh);		
Parameters	The parameters for this method are:		
	SInt32	inLeftDelta	The value indicating the change from the left.
	SInt32	inTopDelta	The value indicating the change from the top.
	Boolean	inRefresh	Whether to refresh or not.
Return	Return true if the View actually scrolls, else false.		

ScrollToPanel()

Purpose	Scroll View Image to the specified panel. This is an override of ScrollToPanel() in LPane .
---------	---

SetDescriptorForPaneID()

Purpose	Set the descriptor of the Pane of a View which has the specified ID. The Pane may be the View itself, or a subpane of the View.							
Access	Virtual, Public							
Prototype	<pre>virtual void SetDescriptorForPaneID(PaneIDT inPaneID, ConstStr255Param inDescriptor);</pre>							
Parameters	The parameters for this method are:							
	<table> <tr> <td>PaneIDT</td><td>inPaneID</td><td>The pane ID.</td></tr> <tr> <td>ConstStr255Param</td><td>inDescriptor</td><td>The string to set for the descriptor.</td></tr> </table>	PaneIDT	inPaneID	The pane ID.	ConstStr255Param	inDescriptor	The string to set for the descriptor.	
PaneIDT	inPaneID	The pane ID.						
ConstStr255Param	inDescriptor	The string to set for the descriptor.						
Return	None							

SetReconcileOverhang()

Purpose	Specify whether to reconcile the Frame and Image when there is overhang. Refer to ReconcileFrameAndImage() to learn more.				
Access	Public				
Prototype	<pre>void SetReconcileOverhang(Boolean inSetting);</pre>				
Parameters	The parameter for this method is:				
	<table> <tr> <td>Boolean</td><td>inSetting</td><td>The value to set for mReconcileOverhang.</td></tr> </table>	Boolean	inSetting	The value to set for mReconcileOverhang .	
Boolean	inSetting	The value to set for mReconcileOverhang .			
Return	None				

SetScrollUnit()

Purpose	Set the mScrollUnit data member.	
Access	Public	
Prototype	<code>void SetScrollUnit(const SPoint32 &inScrollUnit);</code>	
Parameters	The parameter for this method is:	
	<code>const SPoint32& inScrollUnit</code>	The value to set for mScrollUnit .
Return	None	

SetValueForPaneID()

Purpose	Set the value of the pane of a View which has the specified ID. The pane may be the View itself, or a subpane of the View.	
Access	Virtual, Public	
Prototype	<code>virtual void SetValueForPaneID(PaneIDT inPaneID, SInt32 inValue);</code>	
Parameters	The parameters for this method are:	
	<code>PaneIDT inPaneID</code>	The pane ID.
	<code>SInt32 inValue</code>	value to set for the pane of the View.
Return	None	

Show()

Purpose Make a View visible. This is an override of [Show\(\)](#) in [LPane](#).

SubImageChanged()

Purpose This method gets called as notification that the image of some SubView changed size, location, or scroll units. You should override this method to respond to such changes.

Access Virtual, Public

Prototype `virtual void SubImageChanged(LView* inSubView);`

Parameters The parameter for this method is:

LView*	inSubView	The pointer to the SubView.
--------	-----------	-----------------------------

Return None

SuperActivate()

Purpose The SuperView of a View has been activated. This is an override of [SuperActivate\(\)](#) in [LPane](#).

SuperDeactivate()

Purpose The SuperView of a View has been deactivated. This is an override of [SuperDeactivate\(\)](#) in [LPane](#).

SuperDisable()

Purpose The SuperView of a View has been disabled. This is an override of [SuperDisable\(\)](#) in [LPane](#).

SuperEnable()

Purpose The SuperView of a View has been enabled. This is an override of [SuperEnable\(\)](#) in [LPane](#).

SuperHide()

Purpose This is called when the SuperView of a View has been hidden. This is an override of [SuperHide\(\)](#) in [LPane](#).

SuperPrintPanel()

Purpose This is called when the SuperView is printing a panel. This is an override of [SuperPrintPanel\(\)](#) in [LPane](#).

SuperShow()

Purpose The SuperView of a View has become visible. This is an override of [SuperShow\(\)](#) in [LPane](#).

sInFocusView

Purpose	The pointer to the LView that has the focus.
Access	Protected
Prototype	<code>static LView *sInFocusView;</code>

mImageSize

Purpose	The image size.
Access	Protected
Prototype	<code>SDimension32 mImageSize;</code>

mImageLocation

Purpose	The image location.
Access	Protected
Prototype	<code>SPoint32 mImageLocation;</code>

mScrollUnit

Purpose	The scroll unit.
Access	Protected
Prototype	<code>SPoint32 mScrollUnit;</code>

LView

mPortOrigin

Purpose	The origin for the port.
Access	Protected
Prototype	<code>Point mPortOrigin;</code>

mSubPanels

Purpose	A list of the subpanels for this View.
Access	Protected
Prototype	<code>TArray<LPane*> mSubPanels;</code>

mRevealedRect

Purpose	The revealed rectangle.
Access	Protected
Prototype	<code>Rect mRevealedRect;</code>

mUpdateRgnH

Purpose	The handle to the update region.
Access	Protected
Prototype	<code>RgnHandle mUpdateRgnH;</code>

mReconcileOverhang

Purpose	The value indicating whether to reconcile overhang.
Access	Protected
Prototype	Boolean mReconcileOverhang;

LWindow

Overview LWindow is a PowerPlant class that defines window behavior.

Methods The methods in this class are:

<u>LWindow()</u>	<u>~LWindow()</u>
<u>Activate()</u>	<u>ActivateSelf()</u>
<u>ApplyForeAndBackColors()</u>	<u>AttemptClose()</u>
<u>CalcStandardBounds()</u>	<u>CalcStandardBoundsForScreen()</u>
<u>ClearAttribute()</u>	<u>ClickInContent()</u>
<u>ClickInDrag()</u>	<u>ClickInGoAway()</u>
<u>ClickInGrow()</u>	<u>ClickInZoom()</u>
<u>CreateWindow()</u>	<u>Deactivate()</u>
<u>DeactivateSelf()</u>	<u>DoClose()</u>
<u>DoSetBounds()</u>	<u>DoSetPosition()</u>
<u>DoSetZoom()</u>	<u>DrawSelf()</u>
<u>DrawSizeBox()</u>	<u>Enable()</u>
<u>EstablishPort()</u>	<u>ExpandSubPane()</u>
<u>FetchWindowObject()</u>	<u>FindCommandStatus()</u>
<u>FindWindowByID()</u>	<u>GetAERProperty()</u>
<u>GetAELWindowAttribute()</u>	<u>GetDescriptor()</u>
<u>GetGlobalBounds()</u>	<u>GetMacPort()</u>
<u>GetMinMaxSize()</u>	<u>GetStandardSize()</u>
<u>GlobalToPortPoint()</u>	<u>HandleAppleEvent()</u>
<u>HandleClick()</u>	<u>HasAttribute()</u>
<u>HideSelf()</u>	<u>InitWindow()</u>
<u>InvalPortRect()</u>	<u>InvalPortRgn()</u>
<u>MakeMacWindow()</u>	<u>MakeSelfSpecifier()</u>

MoveWindowBy()	MoveWindowTo()
ObeyCommand()	PortToGlobalPoint()
ResizeFrameBy()	ResizeWindowBy()
ResizeWindowTo()	Resume()
Select()	SendAESetBounds()
SendAESetPosition()	SendAESetZoom()
SetAEProperty()	SetAttribute()
SetDescriptor()	SetForeAndBackColors()
SetMinMaxSize()	SetStandardSize()
Show()	ShowSelf()
Suspend()	UpdatePort()
ValidPortRect()	ValidPortRgn()

Data Members The data members in this class are:

mMacWindowP	mMinMaxSize
mStandardSize	mUserBounds
mAttributes	mForeColor
mBackColor	mMoveOnlyUserZoom

Operation LWindow is a complex class that derives from many other classes. Many of the member functions in LWindow are for internal use only, and you shouldn't have to call them directly.

For more information on working with the LWindow class, refer to *The PowerPlant Book*.

Source files (Pane Classes)

LWindow.h

LWindow.cp

Ancestors [LAttachable](#)
[LCommander](#)

 LModelObject

[LPane](#)
[LView](#)

LWindow()

Purpose	The constructor creates objects from the passed-in parameters.
Access	Public
Prototype	<pre> LWindow(); LWindow(const SWindowInfo &inWindowInfo); LWindow(ResIDT inWINDid, UInt32 inAttributes, LCommander *inSuper); LWindow(LStream *inStream); </pre>
Parameters	These constructors have the following parameters:

const SWindowInfo&	inWindowInfo	The SuperView reference.
ResIDT	inWINDid	The resource ID for the WIND resource.
UInt32	inAttributes	The attributes for the window (see Window.h).
LCommander*	inSuper	The pointer to the Supercommander.
LStream*	inStream	The stream to use to create the object.

~LWindow()

Purpose	The destructor destroys the object.
---------	-------------------------------------

LWindow

Access	Public
Prototype	<code>virtual ~LWindow();</code>

Activate()

Purpose	Activate a Window. This is an override of Activate() in LPane .
---------	---

ActivateSelf()

Purpose	This is called when a Window is being activated. This is an override of ActivateSelf() in LPane .
---------	---

ApplyForeAndBackColors()

Purpose	Set the foreground and background colors of the current port. The Window or one of its subpanes must already be focused. This is an override of ApplyForeAndBackColors() in LPane .
---------	---

AttemptClose()

Purpose	Try to close a Window as a result of direct user action.
Access	Virtual, Public
Prototype	<code>virtual void AttemptClose();</code>
Parameters	None
Return	None

CalcStandardBounds()

Purpose	Calculate the bounds of the Window in its standard state and return whether it is in the standard state. The standard state depends on the screen containing the largest area of the Window and the current standard size.	
Access	Virtual, Public	
Prototype	<code>virtual Boolean CalcStandardBounds (Rect &outStdBounds) const;</code>	
Parameters	The parameter for this method is:	
	<code>Rect& outStdBounds</code>	The port rectangle of Window at standard size in global coordinates.
Return	Returns <code>true</code> if in the standard state, else <code>false</code> .	

CalcStandardBoundsForScreen()

Purpose	Calculate the bounds of the Window if it was at a standard (zoomed out) state on a Screen with the specified bounds.	
Access	Virtual, Public	
Prototype	<code>virtual void CalcStandardBoundsForScreen (const Rect &inScreenBounds, Rect &outStdBounds) const;</code>	
Parameters	The parameters for this method are:	
	<code>Rect& outStdBounds</code>	The port rectangle of Window at standard size in global coordinates.
	<code>const Rect& inScreenBounds</code>	Bounding box of screen in global coordinates.
Return	None	

ClearAttribute()

Purpose	Clear the attribute in mAttributes that is passed in to this method.		
Access	Public		
Prototype	<code>void ClearAttribute(EWindAttr inAttribute);</code>		
Parameters	The parameter for this method is: <table><tr><td><code>EWindAttr inAttribute</code></td><td>The window attributes.</td></tr></table>	<code>EWindAttr inAttribute</code>	The window attributes.
<code>EWindAttr inAttribute</code>	The window attributes.		
Return	None		

ClickInContent()

Purpose	Respond to a click in the content region of a Window.				
Access	Virtual, Public				
Prototype	<code>virtual void ClickInContent(const EventRecord &inMacEvent);</code>				
Parameters	The parameter for this method is: <table><tr><td><code>const inMacEvent</code></td><td>The event record for the click.</td></tr><tr><td><code>EventRecord&</code></td><td></td></tr></table>	<code>const inMacEvent</code>	The event record for the click.	<code>EventRecord&</code>	
<code>const inMacEvent</code>	The event record for the click.				
<code>EventRecord&</code>					
Return	None				

ClickInDrag()

Purpose	Process a click in drag region of a Window.
Access	Virtual, Public

Prototype	<code>virtual void ClickInDrag(const EventRecord &inMacEvent);</code>		
Parameters	The parameter for this method is:		
	<code>const</code>	<code>inMacEvent</code>	The event record for the click.
	<code>EventRecord&</code>		
Return	None		

ClickInGoAway()

Purpose	Handle a click inside the close box of a Window.	
Access	Virtual, Public	
Prototype	<code>virtual void ClickInGoAway(const EventRecord &inMacEvent)</code>	
Parameters	The parameter for this method is:	
	<code>const</code>	<code>inMacEvent</code> The event record for the click.
	<code>EventRecord&</code>	
Return	None	

ClickInGrow()

Purpose	Handle a click in the Grow box of a Window.
Access	Virtual, Public
Prototype	<code>virtual void ClickInGrow(const EventRecord &inMacEvent);</code>
Parameters	The parameter for this method is:

LWindow

	<div><div>const</div><div>inMacEvent</div><div>The event record for the click.</div></div> <div>EventRecord&</div>
Return	None

ClickInZoom()

Purpose	Handle a click inside the zoom box of a Window		
Access	Virtual, Public		
Prototype	<pre>virtual void ClickInZoom(const EventRecord &inMacEvent, SInt16 inZoomDirection)</pre>		
Parameters	The parameters for this method are:		
	<div><div>const</div><div>inMacEvent</div><div>The event record for the click.</div></div> <div>EventRecord&</div>		
	<div><div>SInt16</div><div>inZoomDirection</div><div>The direction to zoom.</div></div>		
Return	None		

CreateWindow()

Purpose	Return a newly created Window object initialized from a PPob resource.		
Access	Static, Public		
Prototype	<pre>static Window* CreateWindow(ResIDT inWindowID, LCommander *inSuperCommander);</pre>		
Parameters	The parameters for this method are:		

ResIDT	inWindowID	The resource ID for the window.
LCommander*	inSuperCommander	The pointer to the Supercommander.

Return The pointer to the Window.

Deactivate()

Purpose Deactivate the Window. This is an override of [Deactivate\(\)](#) in [LPane](#).

DeactivateSelf()

Purpose This is called when a Window is being deactivated. This is an override of [DeactivateSelf\(\)](#) in [LPane](#).

DoClose()

Purpose Close a Window.

Access Virtual, Public

Prototype `virtual void DoClose();`

Parameters None

Return None

DoSetBounds()

Purpose	Change size and location of a Window.	
Access	Virtual, Public	
Prototype	<pre>virtual void DoSetBounds(const Rect &inBounds);</pre>	
Parameters	The parameter for this method is:	
	<pre>const inBounds Rect&</pre>	The Rect for the bounds. In global coordinates, this specifies the new size and location of the Window's port rectangle
Return	None	

DoSetPosition()

Purpose	Change the location of a Window.	
Access	Virtual, Public	
Prototype	<pre>virtual void DoSetPosition(Point inPosition);</pre>	
Parameters	The parameter for this method is:	
	<pre>Point inPosition</pre>	The top left corner of the Window's port rectangle is placed at <code>inPosition</code> , which is in global coordinates.
Return	None	

DoSetZoom()

Purpose	Zoom window to either the Standard or User state.	
Access	Virtual, Public	
Prototype	<pre>virtual void DoSetZoom(Boolean inZoomToStdState);</pre>	
Parameters	The parameter for this method is:	
	Boolean inZoomToStdState	Indicate whether to zoom to the standard state.
Return	None	

DrawSelf()

Purpose	This draws the window. This is an override of DrawSelf() in LPane .	
Access	Virtual, Protected	
Prototype	<pre>virtual void DrawSelf();</pre>	
Parameters	None	
Return	None	

DrawSizeBox()

Purpose	Draw standard size box for resizable Windows.	
Access	Virtual, Protected	
Prototype	<pre>virtual void DrawSizeBox();</pre>	
Parameters	None	
Return	None	

Enable()

Purpose	Enable the Window. This is an override of Enable() in LPane .
Access	Virtual, Public
Prototype	<code>virtual void Enable();</code>
Parameters	None
Return	None

EstablishPort()

Purpose	Make Window the current Port.
Access	Public
Prototype	<code>virtual Boolean EstablishPort();</code>
Parameters	None
Return	None
Remarks	If you call this function directly, you should call LView::OutOfFocus() with <code>nil</code> , since changing the current port may invalidate the Focus.

ExpandSubPane()

Purpose	Resize a subpane so that it is the same size as the Window, horizontally and/or vertically. This is an override of ExpandSubPane() in LView .
---------	---

FetchWindowObject()

Purpose	Return the PowerPlant Window object associated with a Mac OS WindowPtr.						
Access	Static, Public						
Prototype	<pre>static LWindow* FetchWindowObject(WindowPtr inWindowP);</pre>						
Parameters	<p>The parameter for this method is:</p> <hr/> <table><tr><td>WindowPt</td><td>inWindowP</td><td>The WindowPtr for the window.</td></tr><tr><td>r</td><td></td><td></td></tr></table> <hr/>	WindowPt	inWindowP	The WindowPtr for the window.	r		
WindowPt	inWindowP	The WindowPtr for the window.					
r							
Return	Returns nil if the WindowPtr is not a PowerPlant Window.						

FindCommandStatus()

Purpose	Pass back whether a Command is enabled and/or marked (in a Menu). This is an override of FindCommandStatus() in LCommander .
---------	--

FindWindowByID()

Purpose	<p>Return the Window object with the specified Pane ID number.</p> <p>This method loops through all Windows from front to back searching for the first one with the specified Pane ID. Note that it's common to use the same PPob (and WIND) resource to create multiple windows, all of which have the same Pane ID. This function returns the frontmost match, whether it is visible or not.</p>
Access	Static, Public
Prototype	<pre>static LWindow* FindWindowByID(PaneIDT inWindowID);</pre>

LWindow

Parameters	The parameter for this method is:		
	PaneIDT	inWindowID	The pane ID for the Window.
Return	Returns <code>nil</code> if no match found, else returns the pointer to the Window.		

GetAEPProperty()

Purpose	This method returns an AppleEvent property.		
Access	Virtual, Public		
Prototype	<pre>virtual void GetAEPProperty(DescType inProperty, const AEDesc &inRequestedType, AEDesc &outPropertyDesc) const;</pre>		
Parameters	The parameters for this method are:		
	DescType	inProperty	The descriptor type for the property.
	const AEDesc&	inRequestedType	The AppleEvent descriptor type.
	AEDesc&	outPropertyDesc	The property descriptor.

GetAEWindowAttribute()

Purpose	Determines the Window attribute based on the received AppleEvent.
Access	Public
Prototype	<pre>void GetAEWindowAttribute(EWindAttr inAttribute, AEDesc &outPropertyDesc) const;</pre>

Parameters	The parameters for this method are:		
	EWindAttr	inAttribute	The attributes for the Window.
	AEDesc&	outPropertyDesc	The property descriptor.
Return	None		

GetDescriptor()

Purpose	Return the title of a Window. This is an override of GetDescriptor() in LPane .
---------	---

GetGlobalBounds()

Purpose	Pass back the port rectangle of a Window in global coordinates.		
Access	Public		
Prototype	<pre>void GetGlobalBounds (Rect &outBounds) const;</pre>		
Parameters	The parameter for this method is:		
	Rect&	outBounds	The rectangle of the window in global coordinates.
Return	None		

GetMacPort()

Purpose	Return Toolbox GrafPort associated with a Window object. This is an override of GetMacPort() in LPane .
---------	---

GetMinMaxSize()

Purpose	This method gets the value of mMinMaxSize .	
Access	Public	
Prototype	<code>void GetMinMaxSize(Rect &outRect) const;</code>	
Parameters	The parameter for this method is:	
	Rect& outRect	The rectangle of the window in global coordinates.
Return	None	

GetStandardSize()

Purpose	This method returns the value of mStandardSize in the <code>outStdSize</code> parameter.	
Access	Public	
Prototype	<code>void GetStandardSize(SDimension16 & outStdSize) const;</code>	
Parameters	The parameter for this method is:	
	SDimension16& outStdSize	The rectangle of the window in global coordinates.
Return	None	

GlobalToPortPoint()

Purpose	Convert a point from global (screen) coordinates to a Window's Port coordinates. This is an override of GlobalToPortPoint() in LPane .
---------	--

HandleAppleEvent()

Purpose	This method handles an AppleEvent.		
Access	Virtual, Public		
Prototype	<pre>virtual void HandleAppleEvent(const AppleEvent &inAppleEvent, AppleEvent &outAEReply, AEDesc &outResult, long inAENumber);</pre>		
Parameters	The parameters for this method are:		
	const AppleEvent &	inAppleEvent	The reference to the AppleEvent.
	AppleEvent &	outAEReply	The AppleEvent reply.
	AEDesc&	outResult	The AppleEvent result.
	long	inAENumber	The AppleEvent number.
Return	None		

HandleClick()

Purpose	Respond to a click on a Window.		
	The inPart parameter is the part code returned by the Toolbox FindWindow routine.		
Access	Virtual, Public		
Prototype	<pre>virtual void HandleClick(const EventRecord &inMacEvent, SInt16 inPart);</pre>		
Parameters	The parameters for this method are:		

LWindow

	<code>const EventRecord&</code>	<code>inMacEvent</code>	The reference to the event information.
	<code>SInt16</code>	<code>inPart</code>	The part code returned by the Toolbox <code>FindWindow</code> routine.
Return	None		

HasAttribute()

Purpose	Checks the Window to see if it has a given attribute.		
Access	Public		
Prototype	<code>Boolean HasAttribute(EWindAttr inAttribute) const;</code>		
Parameters	The parameter for this method is:		
	<code>EWindAttr</code>	<code>inAttribute</code>	The Window attribute to check for.
Return	Returns <code>true</code> if the attribute is present, else <code>false</code> .		

HideSelf()

Purpose	Window is being made invisible. This is an override of HideSelf() in LPane .		
Access	Virtual, Protected		
Prototype	<code>virtual void HideSelf();</code>		
Parameters	None		
Return	None		

InitWindow()

Purpose	Private Initializer from data in a struct.				
Access	Private				
Prototype	<pre>void InitWindow(const SWindowInfo &inWindowInfo);</pre>				
Parameters	The parameter for this method is:				
	<table> <tr> <td>const SWindowInfo&</td><td>inWindowInfo</td><td>The Superwindow information.</td></tr> </table>	const SWindowInfo&	inWindowInfo	The Superwindow information.	
const SWindowInfo&	inWindowInfo	The Superwindow information.			
Return	None				

InvalPortRect()

Purpose	This method calls <code>InvalRect()</code> to invalidate the port rectangle. This is an override of InvalPortRect() in LPane .
---------	--

InvalPortRgn()

Purpose	This method calls <code>InvalRgn()</code> . This is an override of InvalPortRgn() in LPane .
---------	--

MakeMacWindow()

Purpose	Make a new Mac Window from a WIND resource template.
Access	Private
Prototype	<pre>void MakeMacWindow(SInt16 inWINDid);</pre>

LWindow

Parameters	The parameter for this method is:		
	SInt16	inWINDid	The resource ID of the WIND template.
	6		
Return	None		

MakeSelfSpecifier()

Purpose	AppleEvent Object Model Support	
Access	Virtual, Public	
Prototype	<pre>virtual void MakeSelfSpecifier(AEDesc &inSuperSpecifier, AEDesc &outSelfSpecifier) const;</pre>	
Parameters	The parameters for this method are:	
	AEDesc& inSuperSpecifier	The AppleEvent super specifier.
	AEDesc& outSelfSpecifier	The AppleEvent self specifier.
Return	None	

MoveWindowBy()

Purpose	Moves a Window.
Access	Public
Prototype	<pre>void MoveWindowBy(SInt16 inHorizDelta, SInt16 inVertDelta);</pre>
Parameters	The parameters for this method are:

SInt16	inHorizDelta	The amount to move the window in the horizontal directions.
SInt16	inVertDelta	The amount to move the window in the vertical directions.

Return None

MoveWindowTo()

Purpose Move the top left corner of the Window's port rect to the specified location in Global coordinates.

Access Public

Prototype `void MoveWindowTo(
 SInt16 inHoriz,
 SInt16 inVert);`

Parameters The parameters for this method are:

SInt16	inHoriz	The horizontal place to move the window to.
SInt16	inVert	The vertical place to move the window to.

Return None

ObeyCommand()

Purpose Handle a command. This is an override of [ObeyCommand\(\)](#) in [LCommander](#).

PortToGlobalPoint()

Purpose Convert a point from Port to Global (screen) coordinates. Refer to the discussion for [GlobalToPortPoint\(\)](#) for more comments. This is an override of [PortToGlobalPoint\(\)](#) in [LPane](#).

ResizeFrameBy()

Purpose Change the Frame size by the specified amounts. This is an override of [ResizeFrameBy\(\)](#) in [LPane](#).

ResizeWindowBy()

Purpose Change the size of a Window by the specified number of pixels.

Access Public

Prototype

```
void ResizeWindowBy(
    SInt16 inWidthDelta,
    SInt16 inHeightDelta );
```

Parameters The parameters for this method are:

SInt16	inWidthDelta	The horizontal width change to resize the window by.
6		
SInt16	inHeightDelta	The vertical height change to resize the window by.
6		

Return None

ResizeWindowTo()

Purpose	Change the size of a Window to the specified width and height (in pixels).	
Access	Public	
Prototype	<code>void ResizeWindowTo(SInt16 inWidth, SInt16 inHeight);</code>	
Parameters	The parameters for this method are:	
	<code>SInt16 inWidth</code>	The horizontal width for the Window.
	<code>SInt16 inHeight</code>	The vertical height for the Window.
Return	None	

Resume()

Purpose	Brings the window to the front.
Access	Virtual, Public
Prototype	<code>virtual void Resume();</code>
Parameters	None
Return	None

Select()

Purpose	Selects the Window.
Access	Virtual, Public

LWindow

Prototype	<code>virtual void Select();</code>
Parameters	None
Return	None

SendAESetBounds()

Purpose	AppleEvent for changing the size of a Window. inBounds is the new port rectangle for the Window, in Global coordinates. Set inExecuteAE to true to actually resize the Window, false to just send the event for script recording purposes.		
Access	Public		
Prototype	<code>virtual void SendAESetBounds(Rect *inBounds, Boolean inExecuteAE);</code>		
Parameters	The parameters for this method are:		
	Rect*	inBounds	The new port rectangle for the Window, in Global coordinates.
	Boolean	inExecuteAE	Set true to actually resize the Window, false to just send the event for script recording purposes.
Return	None		

SendAESetPosition()

Purpose	AppleEvent for moving a Window to a new position.
Access	Public

Prototype	virtual void SendAESetPosition(Point inPosition, Boolean inExecuteAE);		
Parameters	The parameters for this method are:		
	Point	inPosition	The location for the top left corner of the Window's port rectangle, in Global coordinates.
	Boolean	inExecuteAE	Set true to actually resize the Window, false to just send the event for script recording purposes.
Return	None		
Remarks	You'll use the false value for inExecuteAE when you have already moved the window in response to tracking user actions (the Toolbox trap DragWindow moves the Window).		

SendAESetZoom()

Purpose	AppleEvent for zooming a Window. This method figures out whether to zoom in or out based on the current window size and location
Access	Public
Prototype	virtual void SendAESetZoom();
Parameters	None
Return	None

SetAEPProperty()

Purpose	Set the AppleEvent property.
---------	------------------------------

LWindow

Access	Virtual, Public		
Prototype	<pre>virtual void SetAEProperty(DescType inProperty, const AEDesc &inValue, AEDesc& outAEReply);</pre>		
Parameters	The parameters for this method are:		
	DescType	inProperty	The property descriptor.
	const AEDesc&	inValue	The AppleEvent descriptor.
	AEDesc&	outAEReply	The AppleEvent reply.
Return	None		

SetAttribute()

Purpose	Sets an attribute for the Window.		
Access	Public		
Prototype	<code>void SetAttribute(EWindAttr inAttribute);</code>		
Parameters	The parameter for this method is:		
	<u>EWindAttr</u>	<u>inAttribute</u>	The attribute for the Window.
Return	None		

SetDescriptor()

Purpose	Set the title of a Window. This is an override of SetDescriptor() in LPane .
---------	--

SetForeAndBackColors()

Purpose Specify the foreground and/or background colors of a Window. This is an override of [SetForeAndBackColors\(\)](#) in [LPane](#).

SetMinMaxSize()

Purpose This is an accessor method for [mMinMaxSize](#).

Access Public

Prototype `void SetMinMaxSize(const Rect &inRect);`

Parameters The parameter for this method is:

<code>const Rect&</code>	<code>inRect</code>	The Rect to set mMinMaxSize to.
----------------------------------	---------------------	---

Return None

SetStandardSize()

Purpose Set the standard size ([mStandardSize](#)) for the Window.

Access Public

Prototype `void SetStandardSize(SDimension16 inStdSize);`

Parameters The parameter for this method is:

<code>SDimension16</code>	<code>inStdSize</code>	The Window's standard size (set the value of mStandardSize from this).
---------------------------	------------------------	--

Return None

Show()

Purpose Make a Window visible. This is an override of [Show\(\)](#) in [LPane](#).

ShowSelf()

Purpose Window is being made visible. This is an override of [ShowSelf\(\)](#) in [LPane](#).

Suspend()

Purpose This method hides the window if the `windAttr_HideOnSuspend` attribute is present, else it deactivates the window if it is active and suspend occurs.

Access Virtual, Public

Prototype `virtual void Suspend();`

Parameters None

Return None

UpdatePort()

Purpose Redraw invalidated area of the Window. The Mac WindowPtr maintains an update region that defines the area that needs to be redrawn. This is an override of [UpdatePort\(\)](#) in [LPane](#).

ValidPortRect()

Purpose	This method calls ValidRect(). This is an override of ValidPortRect() in LPane .
---------	--

ValidPortRgn()

Purpose	This method calls ValidRect(). This is an override of ValidPortRgn() in LPane .
---------	---

mMacWindowP

Purpose	This stores the Mac OS WindowPtr.
Access	Protected
Prototype	WindowPtr mMacWindowP;

mMinMaxSize

Purpose	This stores the min and max size for the Window.
Access	Protected
Prototype	Rect mMinMaxSize;

mStandardSize

Purpose	This stores the standard size for the Window.
Access	Protected

LWindow

Prototype `SDimension16 mStandardSize;`

mUserBounds

Purpose This stores the user bounds Rect for the Window.

Access Protected

Prototype `Rect mUserBounds;`

mAttributes

Purpose This stores the attributes for the Window.

Access Protected

Prototype `UInt32 mAttributes;`

mForeColor

Purpose This stores the foreground color for the Window.

Access Protected

Prototype `RGBColor mForeColor;`

mBackColor

Purpose This stores the background color for the Window.

Access Protected

Prototype `RGBColor mBackColor;`

mMoveOnlyUserZoom

Purpose This stores information about zooming a Window that is at standard size, but is partially offscreen.

Access Protected

Prototype Boolean mMoveOnlyUserZoom;

LYieldAttachment

Description	LYieldAttachment is a PowerPlant class that is used for yielding control to another thread before returning. It obviates the need to override LApplication::ProcessNextEvent() in order to give time to other threads.
Methods	The methods in this class are: LYieldAttachment() ExecuteSelf()
Data Members	The data members in this class are: mQuantum mNextTicks
Operation	The concept of an Attachment is a powerful underlying concept of PowerPlant's inner workings. For a detailed discussion on this topic, refer to <i>The PowerPlant Book</i> .
Source files	(Threads Classes) UThread.h UThread.cp
See also	LAttachment LThread

LYieldAttachment()

Purpose	The constructors create the object and initialize the data members to the passed-in values.
Access	Public
Prototype	<code>LYieldAttachment(SInt32 inQuantum);</code>
Parameters	The optional parameter to the constructor lets you specify how much time should elapse before control returns to the caller. This is useful for preventing <code>WaitNextEvent()</code> from being called too often.

ExecuteSelf()

Purpose	This method yields control to other threads. Note that control is always yielded at least once.
Access	Virtual, Public
Prototype	<code>void ExecuteSelf(MessageT inMessage, void* ioParam);</code>
Parameters	This method does not use its parameters.
Return	None

mQuantum

Purpose	The quantum, or wait time in system ticks between calls to LThread::Yield().
Access	Protected
Prototype	<code>SInt32 mQuantum;</code>

mNextTicks

Purpose	A private value used to decide when to yield.
Access	Protected
Prototype	<code>UInt32 mNextTicks;</code>

StAsyncOperation

Overview	StAsyncOperation is a PowerPlant helper class used by the networking classes to delay thread execution while an asynchronous operation executes.	
Methods	The methods in this class are:	
	StAsyncOperation()	~StAsyncOperation()
	AbortOperation()	GetThreadOperation()
	Int_AsyncResume()	WaitForResult()
Data Members	The data members in this class are:	
	mThread	mResult
Source files	(Networking Classes)	
	UNetworking.h	
	UNetworking.cp	
See also	LInterruptSafeList	
	LMacTCPDNSOperation	
	StMacTCPOperation	
	StMacTCPUDPOperation	
	StOpenTptOperation	

StAsyncOperation()

Purpose	The constructor creates the object.
Access	Public
Prototype	StAsyncOperation();

StAsyncOperation

~StAsyncOperation()

Purpose	The destructor destroys the object.
Access	Public
Prototype	<code>~StAsyncOperation();</code>

AbortOperation()

Purpose	Abort the operation.
Access	Virtual, Public
Prototype	<code>virtual void AbortOperation();</code>
Parameters	None
Return	None

GetThreadOperation()

Purpose	Return the operation object.			
Access	Static, Public			
Prototype	<code>static StAsyncOperation *GetThreadOperation(LThread *inThread);</code>			
Parameters	<p>This method has the following parameter:</p> <table><tr><td>LThread*</td><td>inThread</td><td>The thread.</td></tr></table>	LThread*	inThread	The thread.
LThread*	inThread	The thread.		
Return	The operation object.			

Int_AsyncResume()

Purpose	This routine might be called at interrupt time		
	IMPORTANT: If you override this operation, be sure to override AbortOperation() as well if necessary.		
Access	Virtual, Public		
Prototype	<code>virtual void Int_AsyncResume(OSStatusinResult);</code>		
Parameters	This method has the following parameter:		
	OSStatus	inResult	The status code.
Return	None		

WaitForResult()

Purpose	Wait for the result by blocking this thread.		
Access	Public		
Prototype	<code>void WaitForResult();</code>		
Parameters	None		
Return	None		

mThread

Purpose	Storage for the thread.
Access	Protected
Prototype	<code>LThread* mThread;</code>

StAsyncOperation

mResult

Purpose	The storage for the status code.
Access	Protected
Prototype	<code>OSStatus mResult;</code>

sPendingOperations

Purpose	A list of the pending operations.
Access	static Protected
Prototype	<code>static LInterruptSafeList *sPendingOperations;</code>

StCritical

Description	StCritical is a PowerPlant wrapper class for working with critical sections. It would be useful as a stack object.
Methods	The methods in this class are: StCritical() ~StCritical()
Data Members	None
Source files	(Threads Classes) UThread.h UThread.cp
See Also	LThread

StCritical()

Purpose	Wrapper constructor which calls ThreadBeginCritical() to signal the entry of a critical section.
Access	Public
Prototypes	StCritical();
Parameters	None

~StCritical()

Purpose	Wrapper destructor which calls ThreadEndCritical() to signal the exit of a critical section.
Access	Public
Prototype	~StCritical();

StCursor

Overview	StCursor is a PowerPlant class that temporarily changes the cursor to something else.
Methods	The methods in this class are: StCursor() ~StCursor()
Data Members	The data members in this class are: mRestoreID
Operation	Upon entry, specify the ResIDT of the 'CURS' to change to. Upon exit, we either restore the old cursor or default to the arrow. By default, StCursor will set to the watch cursor, and restore the original cursor.
Source files	(Utility Classes) UCursor.h UCursor.cp

StCursor()

Purpose	The default constructor initializes the object.
Access	Public
Prototype	<pre>StCursor(ResIDTinCursID = watchCursor, BooleaninRestoreOriginal = true); StCursor(const StCursor &inOriginal); StCursor&operator = (const StCursor &inOriginal);</pre>
Parameters	The cursor resource ID and whether to restore the cursor or not. The assignment operator and copy constructor require a reference to the original cursor object.

StCursor

~StCursor()

Purpose	The destructor destroys the object.
Access	Public, Virtual
Prototype	<code>virtual ~StCursor();</code>
Parameters	None

mRestoreID

Purpose	The resource ID for restoring.
Access	Protected
Prototype	<code>ResIDT mRestoreID;</code>

StDialogHandler

Overview	StDialogHandler is a PowerPlant class that is used for creating dialog boxes.								
Methods	<p>The methods in this class are:</p> <table><tr><td>StDialogHandler()</td><td>~StDialogHandler()</td></tr><tr><td>AllowSubRemoval()</td><td>DoDialog()</td></tr><tr><td>FindCommandStatus()</td><td>GetDialog()</td></tr><tr><td>ListenToMessage()</td><td>SetSleepTime()</td></tr></table>	StDialogHandler()	~StDialogHandler()	AllowSubRemoval()	DoDialog()	FindCommandStatus()	GetDialog()	ListenToMessage()	SetSleepTime()
StDialogHandler()	~StDialogHandler()								
AllowSubRemoval()	DoDialog()								
FindCommandStatus()	GetDialog()								
ListenToMessage()	SetSleepTime()								
Data Members	<p>The data members in this class are:</p> <table><tr><td>mDialog</td><td>mMessage</td></tr><tr><td>mSleepTime</td><td></td></tr></table>	mDialog	mMessage	mSleepTime					
mDialog	mMessage								
mSleepTime									
Operation	Refer to <i>The PowerPlant Book</i> for important information about mixing the PowerPlant dialog classes with the Mac OS Toolbox calls.								
Source files	<p>(Utility Classes)</p> <p>UModalDialogs.h</p> <p>UModalDialogs.cp</p>								
Ancestors	<p>LAttachable</p> <p>LCommander</p> <p>LEventDispatcher</p> <p>LListener</p>								
See also	LBroadcaster								

StDialogHandler

StDialogHandler()

Purpose	The constructor creates an object from the passed-in parameters.	
Access	Public	
Prototype	<code>StDialogHandler(ResIDT inDialogResID, LCommander *inSuper);</code>	
Parameters	The parameters for this constructor are:	

ResIDT	inDialogResID	The resource ID for the dialog resource.
LCommander*	inSuper	The SuperCommander for this dialog.

~StDialogHandler()

Purpose	The destructor destroys the object.
Access	Virtual, Public
Prototype	<code>virtual ~StDialogHandler();</code>

AllowSubRemoval()

Purpose	This method indicates whether removal of subcommanders is allowed. It is an override of AllowSubRemoval() in LCommander .
---------	---

DoDialog()

Purpose	Handle an event for a dialog box.
Access	Virtual, Public

Prototype	<code>virtual MessageT DoDialog();</code>
Parameters	None
Return	If the event triggers a Broadcaster to broadcast a message, the last such message heard by the DialogHandler is returned. Otherwise, this function returns <code>msg_Nothing</code> .
Remarks	Call this function repeatedly to handle events.

FindCommandStatus()

Purpose	This method passes back the status of a command. It is an override of FindCommandStatus() in LCommander .
---------	---

GetDialog()

Purpose	This is an accessor method for mDialog .
Access	Public
Prototype	<code>LWindow* GetDialog();</code>
Parameters	None
Return	Returns the pointer mDialog to the window for the dialog box.

ListenToMessage()

Purpose	This method provides the message processing behavior for an LListener-inherited object. It is an override of ListenToMessage() in LListener .
---------	---

StDialogHandler

SetSleepTime()

Purpose	This method sets the value of the mSleepTime data member.
Access	Public
Prototype	<code>void SetSleepTime(SInt32 inSleepTime);</code>
Parameters	SInt32 indicating the sleep time to store in mSleepTime .
Return	None

mDialog

Purpose	The pointer to the dialog box.
Access	Protected
Prototype	<code>LWindow *mDialog;</code>

mMessage

Purpose	A place to store the message.
Access	Protected
Prototype	<code>MessageT mMessage;</code>

mSleepTime

Purpose	Storage for the sleep time parameter.
Access	Protected
Prototype	<code>SInt32 mSleepTime;</code>

StMacTCPOperation

Overview	StMacTCPOperation is a PowerPlant class that is used for managing TCPParamBlock and thread blocking for MacTCP interface calls. This keeps a global reference to the MacTCP driver and DNS code segment.	
Methods	The methods in this class are: StMacTCPOperation() AsyncRun() GetParamBlock() Run() ~StMacTCPOperation() GetCompletionProc() Int TCPCompletionProc()	
Data Members	The data members in this class are: mTCPParamBlock sTCPParamBlockDeleteQueue sMacTCPCompletionProc	
Source files	(Networking Classes) UMacTCPSupport.h UMacTCPSupport.cp	
See also	StAsyncOperation StMacTCPListenOperation StMacTCPSendOperation	

StMacTCPOperation()

Purpose	The constructor creates the object.
Access	Public
Prototype	StMacTCPOperation(SInt16 inOperationCode, StreamPtr inStreamPtr);

StMacTCPOperation

Parameters	This constructor has the following parameters:		
	SInt16	inOperationCode	The operation code.
	StreamPtr	inStreamPtr	The stream pointer.

~StMacTCPOperation()

Purpose	The destructor destroys the object.
Access	Public
Prototype	<code>~StMacTCPOperation();</code>

AsyncRun()

Purpose	Run the operation and return immediately. The completion will appear as a notification on the endpoint. Functions using this method can NOT be aborted. Presently used only for LMacTCPTCPEndpoint::SendDisconnect() to mimic behavior of OT.		
Access	Public		
Prototype	<code>void AsyncRun(LMacTCPTCPEndpoint* inEndpoint, UInt16 inEventCode);</code>		
Parameters	This method has the following parameters:		
	LMacTCPTCPEndpoint*	inEndpoint	The endpoint.
	UInt16	inEventCode	The event code.
Return	None		

GetCompletionProc()

Purpose	Return the value of sMacTCPCompletionProc .
Access	Inline, Public
Prototype	<code>inline TCPIOCompletionUPP GetCompletionProc();</code>
Parameters	None
Return	The Completion routine Universal Procedure Pointer (UPP).

GetParamBlock()

Purpose	Retrieve the value of mTCPParamBlock .
Access	Inline, Public
Prototype	<code>inline STCPPParamBlock&GetParamBlock();</code>
Parameters	None
Return	The parameter block.

Int_TCPCompletionProc()

Purpose	The completion routine for the operation.
Access	Protected, Static
Prototype	<pre>pascal void Int_TCPCompletionProc(STCPPParamBlock*inParamBlock #if !GENERATINGCFM : __A0 #endif);</pre>
Parameters	This method has the following parameters:

StMacTCPOperation

	STCParamBlock*	inParamBlock	The endpoint.
Return	None		

Run()

Purpose	Attempt the operation. Check for immediate failure.
Access	Public
Prototype	<code>void Run();</code>
Parameters	None
Return	None

mTCParamBlock

Purpose	The TCP parameter block.
Access	Protected
Prototype	<code>STCParamBlock * mTCParamBlock;</code>

sMacTCPCompletionProc

Purpose	The MacTCP completion procedure.
Access	Private, Static
Prototype	<code>TCPIOCompletionUPP sMacTCPCompletionProc;</code>

sTCPParamBlockDeleteQueue

Purpose	The MacTCP parameter block deletion queue.
Access	Private, Static
Prototype	LTCPParamBlockDeleteQueue * sTCPParamBlockDeleteQueue;

StMacTCPUDPOperation

Overview	StMacTCPUDPOperation is a PowerPlant class that is used for managing UDPPParamBlock and thread blocking for MacTCP related UDP interface calls.	
Methods	The methods in this class are:	
	StMacTCPUDPOperation()	~StMacTCPUDPOperation()
	GetCompletionProc()	GetParamBlock()
	Int_UDPCompletionProc()	Run()
Data Members	The data members in this class are:	
	mUDPPParamBlock	sMacTCPUDPCompletionProc
	sUDPPParamBlockDeleteQueue	
Source files	(Networking Classes)	
	UMacTCPSupport.h	
	UMacTCPSupport.cp	
See also	LGlobalsContext	
	StAsyncOperation	
	StMacTCPUDPSendOperation	

StMacTCPUDPOperation()

Purpose	The constructor creates the object.
Access	Public
Prototype	<pre>StMacTCPUDPOperation(SInt16 inOperationCode, StreamPtr inStreamPtr);</pre>

StMacTCPUDPOperation

Parameters None

~StMacTCPUDPOperation()

Purpose The destructor destroys the object.

Access Public

Prototype `~StMacTCPUDPOperation();`

GetCompletionProc()

Purpose Return the value of [sMacTCPUDPCCompletionProc](#).

Access Inline, Public

Prototype `inline UDPIOCompletionUPP GetCompletionProc();`

Parameters None

Return The procedure pointer.

GetParamBlock()

Purpose Return the value of [mUDPParamBlock](#).

Access Inline, Public

Prototype `inline SUDPParamBlock&GetParamBlock();`

Parameters None

Return The parameter block.

Int_UDPCompletionProc()

Purpose	The completion procedure.		
Access	Static, Protected		
Prototype	<pre>static pascal void Int_UDPCompletionProc(SUDPParamBlock* inParamBlock #if !GENERATINGCFM : __A0 #endif);</pre>		
Parameters	This method has the following parameters: SUDPParamBlock* inParamBlock The parameter block.		
Return	None		

Run()

Purpose	Attempt the operation. Check for immediate failure.		
Access	Public		
Prototype	<pre>void Run();</pre>		
Parameters	None		
Return	None		

mUDPParamBlock

Purpose	The parameter block.		
Access	Protected		
Prototype	<pre>SUDPParamBlock * mUDPParamBlock;</pre>		

StMacTCPUDPOperation

sMacTCPUDPCompletionProc

Purpose	The completion routine Universal Procedure Pointer (UPP).
Access	Static, Private
Prototype	<code>UDPIOCompletionUPP sMacTCPUDPCompletionProc;</code>

sUDPPParamBlockDeleteQueue

Purpose	The parameter block deletion queue.
Access	Static, Private
Prototype	<code>LUDPPParamBlockDeleteQueue *sUDPPParamBlockDeleteQueue;</code>

StMutex

Description	StMutex is a PowerPlant wrapper class for working with mutual exclusions (mutexes).
Methods	The methods in this class are: StMutex() ~StMutex()
Data Members	mMutex
Source files	(Threads Classes) UThread.h UThread.cp
See Also	LMutexSemaphore LThread

StMutex()

Purpose	Constructor waits on the given mutual exclusion semaphore.
Access	Public
Prototypes	StMutex (LMutexSemaphore& inMutex);
Parameters	A reference to the mutex to wait for.

~StMutex()

Purpose	Destructor releases the semaphore.
Access	Public
Prototype	~StMutex ();

StMutex

mMutex

Purpose	The mutex to wait for.
Access	Private
Prototype	LMutexSemaphore&mMutex;

StOpenTptOperation

Overview	StOpenTptOperation is a PowerPlant class that is used for encapsulating an Open Transport operation. It manages thread blocking for OpenTransport interface calls.	
Methods	The methods in this class are:	
	StOpenTptOperation()	~StOpenTptOperation()
	GetCookie()	GetEventCode()
	GetResultCode()	Int_TimerProc()
	OperationTimedOut()	SetEventCode()
	WaitForCompletion()	WaitForResult()
Data Members	The data members in this class are:	
	mNotifHandler	mEventCode
	mCookieTest	mTestCookie
	mResultCode	mCookie
	mOperationTimeout	sOTOpTimerUPP
Source files	(Networking Classes)	
	UOpenTptOperation.h	
	UOpenTptOperation.cp	
See also	LGlobalsContext	

StOpenTptOperation()

Purpose	The constructor creates the object.
Access	Public
Prototype	<code>StOpenTptOperation(LOpenTptNotifHandler * inNotifHandler,</code>

StOpenTptOperation

	<pre>OTEventCode inEventCode, void * inCookieTest, Boolean inTestCookie);</pre>		
Parameters	This constructor has the following parameter:		
	LOpenTptNotifHandler*	inNotifHandler	This is the Open Transport notification handler.
	OTEventCode	inEventCode	The event code.
	void*	inCookieTest	The cookie test value.
	Boolean	inTestCookie	Whether to test the cookie.

~StOpenTptOperation()

Purpose	The destructor destroys the object.
Access	Public
Prototype	<code>~StOpenTptOperation();</code>

GetCookie()

Purpose	Returns the value of mCookie .
Access	Public
Prototype	<code>void* GetCookie();</code>
Parameters	None
Return	The value of mCookie .

GetEventCode()

Purpose	Return the value of mEventCode .
Access	Public
Prototype	<code>OTEventCode GetEventCode();</code>
Parameters	None
Return	The value of mEventCode .

GetResultCode()

Purpose	Return the value of mResultCode .
Access	Public
Prototype	<code>OTResult GetResultCode();</code>
Parameters	None
Return	The value of mResultCode .

Int_TimerProc()

Purpose	Install a Time Manager proc.
Access	Static, Protected
Prototype	<pre>pascal void Int_TimerProc(TMTaskPtr tmTaskPtr #if !GENERATINGCFM : __A1 #endif);</pre>
Parameters	The parameter is the pointer to a Time Manager task block.
Return	None

StOpenTptOperation

OperationTimedOut()

Purpose	Return the value of mOperationTimeout .
Access	Public
Prototype	<code>Boolean OperationTimedOut();</code>
Parameters	None
Return	The value of mOperationTimeout .

SetEventCode()

Purpose	Set the value of mEventCode .		
Access	Public		
Prototype	void SetEventCode(OTEventCode inEventCode) ;		
Parameters	This method has the following parameter:		
	OTEventCode	inEventCode	The Open Transport event code.
Return	None		

WaitForCompletion()

Purpose	Wait for the Time Manager task to complete.
Access	Public
Prototype	<code>void WaitForCompletion(UInt32 inTimeoutSeconds);</code>
Parameters	This method has the following parameter:

	UInt32	inTimeoutSeconds	The Open Transport event code.
Return	None		

WaitForResult()

Purpose	This method breaks into the debugger with a call to <code>DebugStr()</code> .
Access	Public
Prototype	<code>void WaitForResult();</code>
Parameters	None
Return	None

mNotifHandler

Purpose	The notification handler.
Access	Protected
Prototype	<code>LOpenTptNotifHandler * mNotifHandler;</code>

mEventCode

Purpose	The event code.
Access	Protected
Prototype	<code>OTEventCode mEventCode;</code>

StOpenTptOperation

mCookieTest

Purpose	The test cookie.
Access	Protected
Prototype	<code>void * mCookieTest;</code>

mTestCookie

Purpose	Whether to test the cookie.
Access	Protected
Prototype	<code>Boolean mTestCookie;</code>

mResultCode

Purpose	The result code.
Access	Protected
Prototype	<code>OTResult mResultCode;</code>

mCookie

Purpose	The cookie pointer.
Access	Protected
Prototype	<code>void* mCookie;</code>

mOperationTimeout

Purpose	Operation timed out.
Access	Protected
Prototype	Boolean mOperationTimeout;

sOTOpTimerUPP

Purpose	The Open Transport operation timer proc pointer.edure
Access	Static, Protected
Prototype	TimerUPP sOTOpTimerUPP;

StSetupGlobals

Overview	StSetupGlobals is a PowerPlant class that is used for saving and restoring the global variables context.
Methods	The methods in this class are: StSetupGlobals() ~StSetupGlobals()
Data Members	The data members in this class are: mOldGlobals
Source files	(Networking Classes) UCallbackUtils.h UCallbackUtils.cp
See also	LGlobalsContext

StSetupGlobals()

Purpose	For 68K, the constructor sets the A5 (or A4) to the value stored in the given LGlobalsContext object. Destructor restores A5/A4 to its previous value. For PowerPC or 68K Code Fragment Manager, there is no need to set up A5/A4/RTOC since the Mixed Mode Manager will do this automatically. The constructor and destructor do nothing in this case.					
Access	Inline, Public					
Prototype	<pre>inline StSetupGlobals(LGlobalsContext& inGlobalsContext);</pre>					
Parameters	This constructor has the following parameter: <table><tr><td>LGlobalsContext&</td><td>inGlobalsContext</td><td>This is a reference to the globals context.</td></tr></table>			LGlobalsContext&	inGlobalsContext	This is a reference to the globals context.
LGlobalsContext&	inGlobalsContext	This is a reference to the globals context.				

StSetupGlobals

~StSetupGlobals()

Purpose	For 68K, the destructor calls <code>SetA4()</code> or <code>SetA5()</code> . For PowerPC or 68K Code Fragment Manager, there is no need to restore A5/A4/ROTC since the Mixed Mode Manager will do this automatically. The constructor and destructor do nothing in this case.
Access	Inline, Public
Prototype	<code>inline ~StSetupGlobals();</code>
Parameters	None

mOldGlobals

Purpose	Storage for the old globals.
Access	Private
Prototype	<code>long mOldGlobals;</code>

TArray

Overview	TArray is a PowerPlant template class that is a template wrapper for LArray.	
Methods	The methods in this class are:	
	TArray()	~TArray()
	AddItem()	AssignItemsAt()
	FetchIndexOf()	FetchInsertIndexOf()
	FetchItemAt()	FetchItemPtr()
	InsertItemsAt()	Remove()
	Operator[]	
Data Members	There are no data members in this class.	
Operation	You can't store actual objects in a TArray or LArray . Only use TArray and LArray to store pointers to objects (created via new), built-in numerical data types, or data structures (simple structs).	
Source files	(Array Classes)	
	TArray.h	
Ancestors	LArray	
See Also	LComparator	

TArray()

Purpose	Construct an array.
Access	Public
Prototype	<pre>TArray(LComparator *inComparator = nil, Boolean inKeepSorted = false); TArray(UInt32 inItemCount, LComparator *inComparator = nil, Boolean inKeepSorted = false);</pre>

TArray

```
TArray( Handle inItemsHandle,  
LComparator *inComparator = nil,  
Boolean inIsSorted = false,  
Boolean inKeepSorted = false );
```

Parameters The parameters for these constructors are:

LComparator*	inComparator	A pointer to the comparator to use for the array.
Boolean	inKeepSorted	A value indicating whether the array should be kept sorted.
UInt32	inItemCount	The number of items to put in the array.
Handle	inItemsHandle	A handle to the items to put in the array.
Boolean	inIsSorted	A value indicating whether the items are sorted.

~TArray()

Purpose	The destructor destroys the array.
Access	Virtual, Public
Prototype	virtual ~TArray();

AddItem()

Purpose	This adds an item to the array. This is an override of AddItem() in LArray .
---------	--

AssignItemsAt()

Purpose Assign the same value to items in the array starting at the specified index. This is an override of [AssignItemsAt\(\)](#) in [LArray](#).

FetchIndexOf()

Purpose Returns the index of the specified item within the array. This is an override of [FetchIndexOf\(\)](#) in [LArray](#).

FetchInsertIndexOf()

Purpose Return the index at which the specified item would be inserted. This is an override of [FetchInsertIndexOf\(\)](#) in [LArray](#).

FetchItemAt()

Purpose Pass back the item at the specified index. This is an override of [FetchItemAt\(\)](#) in [LArray](#).

FetchItemPtr()

Purpose Returns a pointer to an item.

Access Public

Prototype `T* FetchItemPtr(ArrayIndexT inAtIndex) const;`

Parameters The parameter for this method is:

TArray

ArrayIndexT	inAtIndex	The array index to fetch from.
-------------	-----------	--------------------------------

Return A pointer to the array item at the specified index.

InsertItemsAt()

Purpose Insert items at the specified position in an array. This is an override of [InsertItemsAt\(\)](#) in [LArray](#).

Remove()

Purpose Remove an item from an array. This is an override of [Remove\(\)](#) in [LArray](#).

Operator[]

Purpose The array operator is overloaded for this class to provide convenient array-like access to array items, even though they are implemented differently than the row/column arrangement in memory.

Access Public

Prototype `T& operator [] (ArrayIndexT inAtIndex);`
`const T& operator [] (ArrayIndexT inAtIndex)`
`const;`

Parameters The parameter for this method is:

ArrayIndexT	inAtIndex	The array index to access.
-------------	-----------	----------------------------

Return A reference to the array element/item of interest.

UCursor

Overview	UCursor is a PowerPlant class that wraps calls to the Toolbox routine named <code>SetTheCursor()</code> .												
Methods	<p>The methods in this class are:</p> <table><tr><td>GetCurrentID()</td><td>InitTheCursor()</td></tr><tr><td>InAnimatedCursor()</td><td>SetArrow()</td></tr><tr><td>SetCurrentID()</td><td>SetCross()</td></tr><tr><td>SetInAnimatedCursor()</td><td>SetIBeam()</td></tr><tr><td>SetPlus()</td><td>SetTheCursor()</td></tr><tr><td>SetWatch()</td><td></td></tr></table>	GetCurrentID()	InitTheCursor()	InAnimatedCursor()	SetArrow()	SetCurrentID()	SetCross()	SetInAnimatedCursor()	SetIBeam()	SetPlus()	SetTheCursor()	SetWatch()	
GetCurrentID()	InitTheCursor()												
InAnimatedCursor()	SetArrow()												
SetCurrentID()	SetCross()												
SetInAnimatedCursor()	SetIBeam()												
SetPlus()	SetTheCursor()												
SetWatch()													
Data Members	<p>The data members in this class are:</p> <table><tr><td>sCurrentID</td><td>sInAnimatedCursor</td></tr></table>	sCurrentID	sInAnimatedCursor										
sCurrentID	sInAnimatedCursor												
Operation	<p>Given the ResIDT for a 'CURS' resource, set the cursor to that. If the cursor is currently hidden, <code>SetTheCursor()</code> will not make it visible. There is no Apple-supported or recommend way to determine cursor visibility.</p> <p>You could refer to the "develop" Q&A section from the June 1997 issue of MacTech Magazine for information on how to determine cursor visibility, but use this at your own risk.</p>												
Source files	<p>(Utility Classes)</p> <p><code>UCursor.h</code></p> <p><code>UCursor.cp</code></p>												
See Also	StCursor												

GetCurrentID()

Purpose	Retrieves the value of sCurrentID .
---------	---

UCursor

Access	Static, Public, Inline
Prototype	<code>staticResIDTGetCurrentID(){ return sCurrentID; }</code>
Parameters	None

InitTheCursor()

Purpose	Cursor Initialization.
Access	Static, Public
Prototype	<code>staticvoidInitTheCursor();</code>
Parameters	None

InAnimatedCursor()

Purpose	Returns the value of sInAnimatedCursor .
Access	Static, Public, Inline
Prototype	<code>staticBooleanInAnimatedCursor(){ return sInAnimatedCursor; }</code>
Parameters	None

SetArrow()

Purpose	Sets the cursor to an arrow.
Access	Static, Public, Inline
Prototype	<code>staticvoidSetArrow(){ SetTheCursor(0); }</code>
Parameters	None

SetCurrentID()

Purpose	Sets the cursor ID.
Access	Static, Public, Inline
Prototype	<pre>static void SetCurrentID(const ResIDT inCurrentID) { sCurrentID = inCurrentID; }</pre>
Parameters	The resource ID to set.

SetCross()

Purpose	Sets the cursor to a cross.
Access	Static, Inline, Public
Prototype	<pre>static void SetCross() { SetTheCursor(crossCursor); }</pre>
Parameters	None
Return	None

SetInAnimatedCursor()

Purpose	Sets the value of InAnimatedCursor .
Access	Static, Inline, Public
Prototype	<pre>static void SetInAnimatedCursor(Boolean inInAnimated) { sInAnimatedCursor = inInAnimated; }</pre>
Parameters	Boolean value to set.
Return	None

SetIBeam()

Purpose	Sets the cursor to an I-beam.
Access	Static, Inline, Public
Prototype	<code>static void SetIBeam(){ SetTheCursor(iBeamCursor); }</code>
Parameters	None
Return	None

SetPlus()

Purpose	Sets the cursor to a plus sign (+).
Access	Static, Inline, Public
Prototype	<code>staticvoidSetPlus(){ SetTheCursor(plusCursor); }</code>
Parameters	None
Return	None.

SetTheCursor()

Purpose	Sets the cursor to a value.
Access	Static, Public
Prototype	<code>staticvoidSetTheCursor(ResIDT inCursID);</code>
Parameters	The cursor resource ID.

SetWatch()

Purpose	Sets the cursor to a watch.
Access	Static, Inline, Public
Prototype	<code>static void SetWatch() { SetTheCursor(watchCursor); }</code>
Parameters	None
Return	None.

sCurrentID

Purpose	Current resource ID.
Access	Protected
Prototype	<code>static ResIDTsCurrentID;</code>

sInAnimatedCursor

Purpose	Cursor animation.
Access	Protected
Prototype	<code>static BooleansInAnimatedCursor;</code>

UDNSCache

Overview	UDNSCache is a PowerPlant class that is used for a local cache for DNS entries.	
Methods	The methods in this class are:	
	AddToDNSCache()	CheckCache()
	CreateDNSCacheElem()	GetAddressFromCache()
	GetNameFromCache()	
Data Members	The data members in this class are:	
	sOTDNSNameCache	sOTDNSAddressCache
Source files	(Networking Classes)	
	UDNSCache.h	
	UDNSCache.cp	

AddToDNSCache()

Purpose	Add an entry to the DNS cache.	
Access	Static, Public	
Prototype	static void AddToDNSCache(UInt32 inHostIP, ConstStringPtr inHostName);	
Parameters	The parameters for this method are:	
	UInt32	inHostIP The host IP address.
	ConstStringPtr	inHostName The host name string.
Return	None	

CheckCache()

Purpose	Verify the cache.
Access	Static, Public
Prototype	<code>static void CheckCache();</code>
Parameters	None
Return	None

CreateDNSCacheElem()

Purpose	Create a DNS cache element.		
Access	Static, Protected		
Prototype	<code>static void CreateDNSCacheElem(UINT32 inHostIP, ConstStringPtr inHostName, SDNSCacheElem& outElem);</code>		
Parameters	The parameters for this method are:		
	UINT32	inHostIP	The host IP address.
	ConstStringPtr	inHostName	The host name string.
	SDNSCacheElem&	outElem	The element.
Return	None		

GetAddressFromCache()

Purpose	Retrieve a DNS entry from the cache.
Access	Static, Public
Prototype	<code>static Boolean GetAddressFromCache(</code>

```

        UInt32 inHostIP,
        LStr255& outHostName );

```

Parameters The parameters for this method are:

UInt32	inHostIP	The host IP address.
LStr255&	outHostName	The host name string.

Return None

GetNameFromCache()

Purpose Retrieve a DNS name from the cache.

Access Static, Public

Prototype `static UInt32 GetNameFromCache (ConstStringPtr
inHostName) ;`

Parameters The parameters for this method are:

ConstStringPtr	inHostName	The host name string.
----------------	------------	-----------------------

Return Returns the IP address if found in the cache. If not found, returns zero.

sOTDNSNameCache

Purpose The name cache array.

Access Static, Private

Prototype `static LArray* sOTDNSNameCache;`

UDNSCache

sOTDNSAddressCache

Purpose	The address cache array.
Access	Static, Private
Prototype	<code>static LArray* sOTDNSAddressCache;</code>

UMacTCPSupport

Overview	UMacTCPSupport is a PowerPlant class that is used for helpers for the MacTCP classes. Keeps global reference to MacTCP driver and DNS code segment.
Methods	The methods in this class are: GetMacTCPRefNum() OpenMacTCPDriver() HasMacTCP()
Data Members	The data members in this class are: sMacTCPRefNum sCloseResolverTask
Source files	(Networking Classes) UMacTCPSupport.h UMacTCPSupport.cp

GetMacTCPRefNum()

Purpose	Return the MacTCP reference number, held in sMacTCPRefNum .
Access	Static, Public
Prototype	<code>static SInt16 GetMacTCPRefNum();</code>

HasMacTCP()

Purpose	Determine whether MacTCP is present.
Access	Static, Public
Prototype	<code>static Boolean HasMacTCP();</code>
Parameters	None

UMacTCPSupport

Return Returns true if MacTCP is present, false otherwise.

OpenMacTCPDriver()

Purpose Open the MacTCP driver if not already open.

Access Static, Public

Prototype `static void OpenMacTCPDriver();`

Parameters None

Return None

sMacTCPRefNum

Purpose The MacTCP reference number.

Access Static, Private

Prototype `static SInt16 sMacTCPRefNum;`

sCloseResolverTask

Purpose Storage for the LMacTCP_CloseResolver.

Access Static, Private

Prototype `static LMacTCP_CloseResolver* sCloseResolverTask;`

UNetworkFactory

Overview	UNetworkFactory is a PowerPlant class that is used for easily creating TCP endpoints and mappers based on the current running system software.						
Methods	<p>The methods in this class are:</p> <table><tr><td>CreateInternetMapper()</td><td>CreateTCPEndpoint()</td></tr><tr><td>CreateUDPEndpoint()</td><td>HasMacTCP()</td></tr><tr><td>HasOpenTransport()</td><td>HasTCP()</td></tr></table>	CreateInternetMapper()	CreateTCPEndpoint()	CreateUDPEndpoint()	HasMacTCP()	HasOpenTransport()	HasTCP()
CreateInternetMapper()	CreateTCPEndpoint()						
CreateUDPEndpoint()	HasMacTCP()						
HasOpenTransport()	HasTCP()						
Data Members	There are no data members in this class.						
Operation	Calling simple methods in this utility class automatically chooses MacTCP or OpenTransport support transparently.						
Source files	(Networking Classes) UNetworkFactory.h UNetworkFactory.cp						
See also	LEndpoint LTCPEndpoint LUDPEndpoint						

CreateInternetMapper()

Purpose	Create the best name mapper object for Internet address resolution given the available system software.
Access	Static, Public
Prototype	<code>LInternetMapper* CreateInternetMapper () ;</code>
Parameters	None
Return	A pointer to the mapper object.

CreateTCPEndpoint()

Purpose	Create the best endpoint object for TCP/IP connections given the available system software.	
Access	Static, Public	
Prototype	<code>LTCPEndpoint* CreateTCPEndpoint(UInt32 inReceiveBufferSize)</code>	
Parameters	The parameter for these constructors is:	
	<code>UInt32 inReceiveBuf</code>	The pointer to the thread.
	<code>ferSize</code>	
Return	A pointer to the LTCPEndpoint .	

CreateUDPEndpoint()

Purpose	Create the best endpoint object for TCP/IP connections given the available system software.	
Access	Static, Public	
Prototype	<code>LUDPEndpoint* CreateUDPEndpoint();</code>	
Parameters	None	
Return	A pointer to the LUDPEndpoint.	

HasMacTCP()

Purpose	Indicates whether MacTCP is present.
Access	Static, Public
Prototype	<code>Boolean HasMacTCP();</code>

Parameters	None
Return	Return true if MacTCP is present.

HasOpenTransport()

Purpose	Indicates whether Open Transport is present.
Access	Static, Public
Prototype	<code>Boolean HasOpenTransport () ;</code>
Parameters	None
Return	Return true if OpenTransport is present.

HasTCP()

Purpose	Determines whether MacTCP or OpenTransport are installed.
Access	Static, Public
Prototype	<code>Boolean HasTCP () ;</code>
Parameters	None
Return	Returns true if either MacTCP or Open Transport are installed.

UOpenTptSupport

Overview	UOpenTptSupport is a PowerPlant class that is used for helping other Open Transport classes.	
Methods	The methods in this class are: GetOTGestalt() HasOpenTransportTCP() StartOpenTransport() HasOpenTransport() OTAddressToPPAddress()	
Data Members	The data members in this class are: sOTGestaltTested sOTGestaltResult sCloseOpenTptTask	
Source files	(Networking Classes) UOpenTptSupport.h UOpenTptSupport.cp	

GetOTGestalt()

Purpose	This method is internal only, and is used to perform Open Transport-specific gestalt activity.
Access	Static, Private
Prototype	<code>static void GetOTGestalt();</code>
Parameters	None
Return	None

HasOpenTransport()

Purpose	This method determines whether the machine has Open Transport installed.
Access	Static, Public
Prototype	<code>static Boolean HasOpenTransport();</code>
Parameters	None
Return	Returns true if Open Transport is installed on the machine.

HasOpenTransportTCP()

Purpose	Checks for certain Open Transport gestalt parameters.
Access	Static, Public
Prototype	<code>static Boolean HasOpenTransportTCP();</code>
Parameters	None
Return	Returns true if certain gestalt parameters for Open Transport are set.

OTAddressToPPAddress()

Purpose	Convert an Open Transport address to PowerPlant LInternetAddress .			
Access	Static, Public			
Prototype	<code>static LInternetAddress* OTAddressToPPAddress (OTAddress* inAddress);</code>			
Parameters	<p>This method has the following parameter:</p> <table><tr><td>OTAddress*</td><td>inAddress</td><td>The address to convert.</td></tr></table>	OTAddress*	inAddress	The address to convert.
OTAddress*	inAddress	The address to convert.		
Return	An LInternetAddress .			

StartOpenTransport()

Purpose	Initialize OpenTransport.
Access	Static, Public
Prototype	<code>static void StartOpenTransport();</code>
Parameters	None
Return	None

sOTGestaltTested

Purpose	Whether gestalt has been queried.
Access	Static, Private
Prototype	<code>Boolean sOTGestaltTested;</code>

sCloseOpenTptTask

Purpose	Whether to close the Open Transport task or not.
Access	Static, Private
Prototype	<code>LOpenTpt_CloseOpenTpt* sCloseOpenTptTask;</code>

sOTGestaltResult

Purpose	The result of querying <code>Gestalt()</code> .
Access	Static, Private
Prototype	<code>Int32 sOTGestaltResult;</code>

UReanimator

Overview	UReanimator provides functions for creating objects using data from PPob resources.
Methods	<p>The methods in this class are:</p> <p>CreateView() LinkListenerToBroadcasters()</p> <p>LinkListenerToControls() ObjectsFromStream()</p> <p>ReadObjects()</p>
Data Members	There are no data members in this class.
Operation	This class consists of a few static methods that can be called from anywhere in your code. There are no constructors or destructors for this class.
Source files	(Utility Classes)
	<code>UReanimator.h</code>
	<code>UReanimator.cp</code>
Ancestors	None
See Also	LListener
	LCommander
	LView

CreateView()

Purpose	Return a newly created View object initialized from a PPob resource.
Access	Static, Public
Prototype	<pre>static LView* CreateView(ResIDT inViewID, LView* inSuperView,</pre>

UReanimator

`LCommander* inSuperCommander);`

Parameters

<code>ResIDT</code>	<code>inViewID</code>	Resource ID of view to create
<code>LView*</code>	<code>inSuperView</code>	Pointer to object's super-view
<code>LCommander*</code>	<code>inSuperCommander</code>	Pointer to object's super-commander

Returns Pointer to the created view.

LinkListenerToBroadcasters()

Purpose Associate a Listener with one or more Broadcasters specified as a list of Pane ID's stored in a resource.

Access Static, Public

Prototype `static void LinkListenerToBroadcasters(
 LListener* inListener,
 LView* inControlContainer,
 ResIDT inResListID);`

Parameters

<code>LListener*</code>	<code>inListener</code>	Pointer to the listener
<code>LView*</code>	<code>inControlContainer</code>	Pointer to the view containing the listener
<code>ResIDT</code>	<code>inResListID</code>	Resource ID of the broadcaster list

Returns None

Remarks Only Panes that are also Broadcasters are linked to the Listener. Panes that are not Broadcasters are ignored.

LinkListenerToControls()

Purpose	Associate a Listener with Controls specified as a list of Pane IDs sotred in a 'RidL' resoruce.											
Access	Static, Public											
Prototype	<pre>static void LinkListenerToControls(LListener* inListener, LView* inControlContainer, ResIDT inResListID)</pre>											
Parameters	<table><tr><td>LListener*</td><td>inListener</td><td>Pointer to the listener</td></tr><tr><td>LView*</td><td>inControlContainer</td><td>Pointer to the view containing the listener</td></tr><tr><td>ResIDT</td><td>inResListID</td><td>'RidL' resource ID</td></tr></table>			LListener*	inListener	Pointer to the listener	LView*	inControlContainer	Pointer to the view containing the listener	ResIDT	inResListID	'RidL' resource ID
LListener*	inListener	Pointer to the listener										
LView*	inControlContainer	Pointer to the view containing the listener										
ResIDT	inResListID	'RidL' resource ID										
Return	None											
Remarks	You should probably use LinkListenerToBroadcasters() instead of this function. LinkListenerToBroadcasters() is more general since it handles all panes that are also Broadcasters rather than just Controls.											

ObjectsFromStream()

Purpose	Create new objects from the data in a Stream and return a pointer to the first object created. As it encounters PPob class IDs in the stream, it calls URegistrar::CreateObject() to create those classes from data in the stream.		
Access	Static, Public		
Prototype	static void* ObjectsFromStream(LStream* inStream)		
Parameters	LStream*	inStream	Pointer to the data steram

UReanimator

Return	Pointer to first created object.
Remarks	This function is primarily for use by the ReadObjects() function, which first loads the PPob resource into memory, then calls this function to read thememory as a stream.

ReadObjects()

Purpose	Create new objects from data in a PPob resource and return a pointer to the first object created.		
	Signals an exception if the version number of teh PPob resource does not match the version number expected by the library.		
Access	Static, Public		
Prototype	<pre>static void* ReadObjects(OSType inResType, ResIDT inResID)</pre>		
Parameters			
	OSType	inResType	Resource type
	ResIDT	inResID	ID of PPob resource
Return	A pointer to the first created object.		
Remarks	Before calling, be sure to properly set the default Commander, View, and Attachable.		

UTextDrawing

Overview	UTextDrawing is a PowerPlant class that is used for drawing text in LTextButton and LCaption objects.
Methods	There is only one method in this class. DrawWithJustification()
Data Members	There are no data members in this class.
Operation	The method in this class provides the same functionality as the MacOS <code>TextBox()</code> routine, but does not erase the box before drawing. This improves performance. This means that you must erase the text before drawing if you change it dynamically.
Source files	(Utility Classes) <code>UDrawingUtils.h</code> <code>UDrawingUtils.cp</code>
Ancestors	None
See also	LAttachment LCaption LTextButton

DrawWithJustification()

Purpose	This method draws text in LTextButton and LCaption objects.
Access	Static, Public
Prototype	<pre>static void DrawWithJustification(Ptr inText, SInt32 inLength, const Rect &inRect, SInt16 inJustification, Boolean inFirstLeading = true)</pre>

UTextDrawing

Parameters	The parameters for this method are:		
	Ptr	inText	A pointer to the text to draw.
	SInt32	inLength	The length of the text string.
	const Rect&	inRect	The Rect to draw into.
	SInt16	inJustification	The justification to use when drawing the text.
	Boolean	inFirstLeading	If true (the default), then leading will be added above the first line drawn.
Return	None		