ZigPad

1.0

Generated by Doxygen 1.7.4

Thu Jun 9 2011 10:16:56

Contents

Chapter 1

Todo List

Class SynchronizerConnection : The class is currently only used either as a outbound or a inbound connection, but both a input and output stream is opened for both. This should be improved somehow.

Chapter 2

Data Structure Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

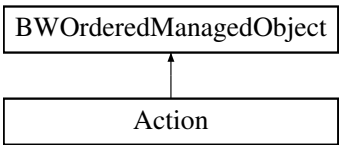
Chapter 5

Data Structure Documentation

5.1 Action Class Reference

`#import <Action.h>`

Inheritance diagram for Action:



Public Member Functions

- (void) - addParamsObject:
- (Param ∗) - getParamForKey:

Properties

- NSString ∗ name
- NSString ∗ type
- NSNumber ∗ favorite
- NSNumber ∗ refId
- NSSet ∗ sequences
- NSSet ∗ params

5.1.1 Detailed Description

Definition at line 15 of file Action.h.

5.1.2 Member Function Documentation

5.1.2.1 - (void) addParamsObject: dummy(Param *) *value*

Definition at line 50 of file Action.m.

5.1.2.2 - (Param *) getParamForKey: dummy(NSString *) *key*

Definition at line 78 of file Action.m.

5.1.3 Property Documentation

5.1.3.1 - (NSNumber*) favorite [read, write, retain]

Definition at line 20 of file Action.h.

5.1.3.2 - (NSString*) name [read, write, retain]

Definition at line 18 of file Action.h.

5.1.3.3 - (NSSet*) params [read, write, retain]

Definition at line 23 of file Action.h.

5.1.3.4 - (NSNumber*) refId [read, write, retain]

Definition at line 21 of file Action.h.

5.1.3.5 - (NSSet*) sequences [read, write, retain]

Definition at line 22 of file Action.h.

5.1.3.6 - (NSString*) type [read, write, retain]

Definition at line 19 of file Action.h.

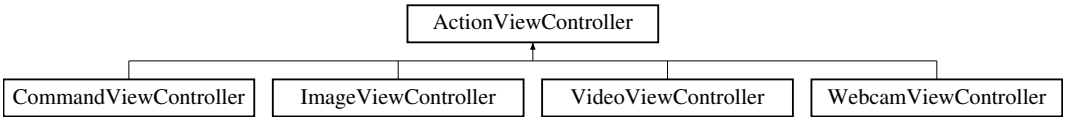
The documentation for this class was generated from the following files:

- ZigPad/coredata/Action.h
- ZigPad/coredata/Action.m

5.2 ActionViewController Class Reference

`#import <ActionViewController.h>`

Inheritance diagram for ActionViewController:



Public Member Functions

- (void) - next:
- (void) - previous
- (IBAction) - click:
- (void) - handleSwipeFrom:
- (void) - registerNotificationCenter
- (void) - unregisterNotificationCenter
- (void) - fireSyncEvent:

Static Public Member Functions

- (ActionViewController ∗) + getViewControllerFromAction:

Properties

- IBOutlet UILabel ∗ label
- IBOutlet UILabel ∗ actionLabel
- IBOutlet UIButton ∗ imageButton
- Presentation ∗ presentation
- BOOL isMaster

5.2.1 Detailed Description

Base view controller for all actions.

Extend this class when creating a new action controller and override the default implementations as appropriate.

Definition at line 25 of file ActionViewController.h.

5.2.2 Member Function Documentation

5.2.2.1 - (void) click: dummy(id) *sender*

Click callback for the image button.

Parameters

sender	Indicates the button that is executing this action.

Definition at line 125 of file ActionViewController.m.

5.2.2.2 - (void) fireSyncEvent: dummy(SyncEventSwipeDirection) *direction*

Send a sync event.

This sends a sync notification event for the currently active action.

Parameters

direction	Indicates the animation direction of the connected devices.

Definition at line 110 of file ActionViewController.m.

5.2.2.3 + (ActionViewController ∗) getViewControllerFromAction: dummy(Action ∗) *action*

Helper method that initiates the view controller for a given action.

Parameters

action	The action for which the view controller should be created.

Returns

A child-class of ActionViewController.

Definition at line 21 of file ActionViewController.m.

5.2.2.4 - (void) handleSwipeFrom: dummy(UISwipeGestureRecognizer ∗) *recogniser*

Callback to handle swipe events.

Parameters

recogniser	Instance of the swipe recognizer instance.

Definition at line 208 of file ActionViewController.m.

5.2.2.5 - (void) next: dummy(BOOL) *animated*

Switch to the next action in the current presentation.

Parameters

animated	If the switch should be animated (swipe) or not (click).

Definition at line 251 of file ActionViewController.m.

5.2.2.6 - (void) previous

Switch to the previous action in the current presentation.

Definition at line 291 of file ActionViewController.m.

5.2.2.7 - (void) registerNotificationCenter

Register for sync events through the notification center.

Definition at line 35 of file ActionViewController.m.

5.2.2.8 - (void) unregisterNotificationCenter

Unregister for sync events through the notification center.

Definition at line 42 of file ActionViewController.m.

5.2.3 Property Documentation

5.2.3.1 - (IBOutlet UILabel∗) actionLabel [read, write, retain]

Reference to the label that displays the current action name.

Definition at line 37 of file ActionViewController.h.

5.2.3.2 - (IBOutlet UIButton∗) imageButton [read, write, retain]

Definition at line 42 of file ActionViewController.h.

5.2.3.3 - (BOOL) isMaster [read, write, assign]

A flag to indicate if this is the currently active device.

Active means in this case that it is directly controlled through the UI as opposed to synchronization events..

Definition at line 55 of file ActionViewController.h.

5.2.3.4 - (IBOutlet UILabel∗) label [read, write, retain]

Reference to the label that displays the current sequence.

Definition at line 32 of file ActionViewController.h.

5.2.3.5 - (Presentation∗) presentation [read, write, retain]

The active presentation instance.

Definition at line 47 of file ActionViewController.h.

The documentation for this class was generated from the following files:

- ZigPad/ActionViewController.h
- ZigPad/ActionViewController.m

5.3 AnimatorHelper Class Reference

`#import <AnimatorHelper.h>`

Static Public Member Functions

- (void) + slideWithAnimation:direction:actualPage:nextPage:fullAnimated:pushOnStack:

5.3.1 Detailed Description

Definition at line 13 of file AnimatorHelper.h.

5.3.2 Member Function Documentation

5.3.2.1 + (void) slideWithAnimation: dummy(int) direction:(UIViewController∗) actualPage:(UIViewController∗) nextPage:(bool) fullAnimated:(bool) pushOnStack:(bool) *popOnStack*

Definition at line 16 of file AnimatorHelper.m.

The documentation for this class was generated from the following files:

- ZigPad/AnimatorHelper.h
- ZigPad/AnimatorHelper.m

5.4 AsyncReadPacket Class Reference

Public Member Functions

- (id) - initWithData:timeout:tag:readAllAvailable:terminator:maxLength:
- (unsigned) - readLengthForTerm
- (unsigned) - prebufferReadLengthForTerm
- (CFIndex) - searchForTermAfterPreBuffering:

Data Fields

- NSMutableData * buffer
- CFIndex bytesDone
- NSTimeInterval timeout
- CFIndex maxLength
- long tag
- NSData * term
- BOOL readAllAvailableData

5.4.1 Detailed Description

The AsyncReadPacket encompasses the instructions for any given read. The content of a read packet allows the code to determine if we're:

- reading to a certain length
- reading to a certain separator
- or simply reading the first chunk of available data

Definition at line 153 of file AsyncTCPSocket.m.

5.4.2 Member Function Documentation

5.4.2.1 - (id) initWithData: dummy(NSMutableData *) *d* timeout:(NSTimeInterval) *t* tag:(long) *i* readAllAvailable:(BOOL) *a* terminator:(NSData *) *e* maxLength:(CFIndex) *m*

Definition at line 179 of file AsyncTCPSocket.m.

5.4.2.2 - (unsigned) prebufferReadLengthForTerm

Assuming pre-buffering is enabled, returns the amount of data that can be read without going over the maxLength.

Definition at line 252 of file AsyncTCPSocket.m.

5.4.2.3 - (unsigned) readLengthForTerm

For read packets with a set terminator, returns the safe length of data that can be read without going over a terminator, or the maxLength.

It is assumed the terminator has not already been read.

Definition at line 205 of file AsyncTCPSocket.m.

5.4.2.4 - (CFIndex) searchForTermAfterPreBuffering: dummy(CFIndex) *numBytes*

For read packets with a set terminator, scans the packet buffer for the term. It is assumed the terminator had not been fully read prior to the new bytes.

If the term is found, the number of excess bytes after the term are returned. If the term is not found, this method will return -1.

Note: A return value of zero means the term was found at the very end.

Definition at line 269 of file AsyncTCPSocket.m.

5.4.3 Field Documentation

5.4.3.1 - (NSMutableData∗) buffer

Definition at line 156 of file AsyncTCPSocket.m.

5.4.3.2 - (CFIndex) bytesDone

Definition at line 157 of file AsyncTCPSocket.m.

5.4.3.3 - (CFIndex) maxLength

Definition at line 159 of file AsyncTCPSocket.m.

5.4.3.4 - (BOOL) readAllAvailableData

Definition at line 162 of file AsyncTCPSocket.m.

5.4.3.5 - (long) tag

Definition at line 160 of file AsyncTCPSocket.m.

5.4.3.6 - (NSData∗) term

Definition at line 161 of file AsyncTCPSocket.m.

5.4.3.7 - (NSTimeInterval) timeout

Definition at line 158 of file AsyncTCPSocket.m.

The documentation for this class was generated from the following file:

- ZigPad/AsyncTCPSocket.m

5.5 AsyncReceivePacket Class Reference

Public Member Functions

- (id) - initWithTimeout:tag:

Data Fields

- NSTimeInterval timeout
- long tag
- NSMutableData * buffer
- NSString * host
- UInt16 port

5.5.1 Detailed Description

The AsyncReceivePacket encompasses the instructions for a single receive/read.

Definition at line 163 of file AsyncUdpSocket.m.

5.5.2 Member Function Documentation

5.5.2.1 - (id) initWithTimeout: dummy(NSTimeInterval) *t* tag:(long) *i*

Definition at line 177 of file AsyncUdpSocket.m.

5.5.3 Field Documentation

5.5.3.1 - (NSMutableData*) buffer

Definition at line 168 of file AsyncUdpSocket.m.

5.5.3.2 - (NSString*) host

Definition at line 169 of file AsyncUdpSocket.m.

5.5.3.3 - (UInt16) port

Definition at line 170 of file AsyncUdpSocket.m.

5.5.3.4 - (long) tag

Definition at line 167 of file AsyncUdpSocket.m.

5.5.3.5 - (NSTimeInterval) timeout

Definition at line 166 of file AsyncUdpSocket.m.

The documentation for this class was generated from the following file:

- ZigPad/AsyncUdpSocket.m

5.6 AsyncSendPacket Class Reference

Public Member Functions

- (id) - initWithData:address:timeout:tag:

Data Fields

- NSData * buffer
- NSData * address
- NSTimeInterval timeout
- long tag

5.6.1 Detailed Description

The AsyncSendPacket encompasses the instructions for a single send/write.

Definition at line 122 of file AsyncUdpSocket.m.

5.6.2 Member Function Documentation

5.6.2.1 - (id) initWithData: dummy(NSData *) *d* address:(NSData *) *a* timeout:(NSTimeInterval) *t* tag:(long) *i*

Definition at line 135 of file AsyncUdpSocket.m.

5.6.3 Field Documentation

5.6.3.1 - (NSData∗) address

Definition at line 126 of file AsyncUdpSocket.m.

5.6.3.2 - (NSData∗) buffer

Definition at line 125 of file AsyncUdpSocket.m.

5.6.3.3 - (long) tag

Definition at line 128 of file AsyncUdpSocket.m.

5.6.3.4 - (NSTimeInterval) timeout

Definition at line 127 of file AsyncUdpSocket.m.

The documentation for this class was generated from the following file:

- ZigPad/AsyncUdpSocket.m

5.7 AsyncSpecialPacket Class Reference

Public Member Functions

- (id) - initWithTLSSettings:

Data Fields

- NSDictionary ∗ tlsSettings

5.7.1 Detailed Description

The AsyncSpecialPacket encompasses special instructions for interruptions in the read-/write queues. This class my be altered to support more than just TLS in the future.

Definition at line 353 of file AsyncTCPSocket.m.

5.7.2 Member Function Documentation

5.7.2.1 - (id) initWithTLSSettings: dummy(NSDictionary ∗) *settings*

Definition at line 363 of file AsyncTCPSocket.m.

5.7.3 Field Documentation

5.7.3.1 - (NSDictionary∗) tlsSettings

Definition at line 356 of file AsyncTCPSocket.m.

The documentation for this class was generated from the following file:

- ZigPad/AsyncTCPSocket.m

5.8 AsyncTCPSocket Class Reference

```
#import <AsyncTCPSocket.h>
```

Public Member Functions

- (id) - init
- (id) - initWithDelegate:
- (id) - initWithDelegate:userData:
- (NSString ∗) - description
- (id) - delegate
- (BOOL) - canSafelySetDelegate
- (void) - setDelegate:
- (long) - userData
- (void) - setUserData:
- (CFSocketRef) - getCFSocket
- (CFReadStreamRef) - getCFReadStream
- (CFWriteStreamRef) - getCFWriteStream
- (BOOL) - acceptOnPort:error:
- (BOOL) - acceptOnAddress:port:error:
- (BOOL) - connectToHost:onPort:error:
- (BOOL) - connectToHost:onPort:withTimeout:error:
- (BOOL) - connectToAddress:error:
- (BOOL) - connectToAddress:withTimeout:error:
- (void) - disconnect
- (void) - disconnectAfterReading
- (void) - disconnectAfterWriting
- (void) - disconnectAfterReadingAndWriting
- (BOOL) - isConnected
- (NSString ∗) - connectedHost
- (UInt16) - connectedPort
- (NSString ∗) - localHost
- (UInt16) - localPort
- (BOOL) - isIPv4
- (BOOL) - isIPv6

- (void) - readDataToLength:withTimeout:tag:
- (void) - readDataToData:withTimeout:tag:
- (void) - readDataToData:withTimeout:maxLength:tag:
- (void) - readDataWithTimeout:tag:
- (void) - writeData:withTimeout:tag:
- (float) - progressOfReadReturningTag:bytesDone:total:
- (float) - progressOfWriteReturningTag:bytesDone:total:
- (void) - startTLS:
- (void) - enablePreBuffering
- (BOOL) - moveToRunLoop:
- (BOOL) - setRunLoopModes:
- (NSData *) - unreadData
- (void) - startConnectTimeout:
- (void) - endConnectTimeout
- (CFSocketRef) - createAcceptSocketForAddress:error:
- (BOOL) - createSocketForAddress:error:
- (BOOL) - attachSocketsToRunLoop:error:
- (BOOL) - configureSocketAndReturnError:
- (BOOL) - connectSocketToAddress:error:
- (void) - doAcceptWithSocket:
- (void) - doSocketOpen:withCFSocketError:
- (BOOL) - createStreamsFromNative:error:
- (BOOL) - createStreamsToHost:onPort:error:
- (BOOL) - attachStreamsToRunLoop:error:
- (BOOL) - configureStreamsAndReturnError:
- (BOOL) - openStreamsAndReturnError:
- (void) - doStreamOpen
- (BOOL) - setSocketFromStreamsAndReturnError:
- (void) - closeWithError:
- (void) - recoverUnreadData
- (void) - emptyQueues
- (void) - close
- (NSError *) - getErrnoError
- (NSError *) - getAbortError
- (NSError *) - getStreamError
- (NSError *) - getSocketError
- (NSError *) - getConnectTimeoutError
- (NSError *) - getReadMaxedOutError
- (NSError *) - getReadTimeoutError
- (NSError *) - getWriteTimeoutError
- (NSError *) - errorFromCFStreamError:
- (BOOL) - isSocketConnected
- (BOOL) - areStreamsConnected
- (NSString *) - connectedHost:
- (UInt16) - connectedPort:
- (NSString *) - localHost:

- (UInt16) - localPort:
- (NSString *) - addressHost:
- (UInt16) - addressPort:
- (void) - doBytesAvailable
- (void) - completeCurrentRead
- (void) - endCurrentRead
- (void) - scheduleDequeueRead
- (void) - maybeDequeueRead
- (void) - doReadTimeout:
- (void) - doSendBytes
- (void) - completeCurrentWrite
- (void) - endCurrentWrite
- (void) - scheduleDequeueWrite
- (void) - maybeDequeueWrite
- (void) - maybeScheduleDisconnect
- (void) - doWriteTimeout:
- (void) - runLoopAddSource:
- (void) - runLoopRemoveSource:
- (void) - runLoopAddTimer:
- (void) - runLoopRemoveTimer:
- (void) - runLoopUnscheduleReadStream
- (void) - runLoopUnscheduleWriteStream
- (void) - maybeStartTLS
- (void) - onTLSStarted:
- (void) - doCFCallback:forSocket:withAddress:withData:
- (void) - doCFReadStreamCallback:forStream:
- (void) - doCFWriteStreamCallback:forStream:

Static Public Member Functions

- (NSData *) + CRLFData
- (NSData *) + CRData
- (NSData *) + LFData
- (NSData *) + ZeroData

Protected Attributes

- CFSocketRef theSocket
- CFSocketRef theSocket6
- CFReadStreamRef theReadStream
- CFWriteStreamRef theWriteStream
- CFRunLoopSourceRef theSource
- CFRunLoopSourceRef theSource6
- CFRunLoopRef theRunLoop
- CFSocketContext theContext

- NSArray * theRunLoopModes
- NSTimer * theConnectTimer
- NSMutableArray * theReadQueue
- AsyncReadPacket * theCurrentRead
- NSTimer * theReadTimer
- NSMutableData * partialReadBuffer
- NSMutableArray * theWriteQueue
- AsyncWritePacket * theCurrentWrite
- NSTimer * theWriteTimer
- id theDelegate
- UInt16 theFlags
- long theUserData

5.8.1 Detailed Description

Definition at line 109 of file AsyncTCPSocket.h.

5.8.2 Member Function Documentation

5.8.2.1 - (BOOL) acceptOnAddress: dummy(NSString *) *hostaddr* port:(UInt16) *port* error:(NSError **) *errPtr*

This method is the same as acceptOnPort:error: with the additional option of specifying which interface to listen on. So, for example, if you were writing code for a server that has multiple IP addresses, you could specify which address you wanted to listen on. Or you could use it to specify that the socket should only accept connections over ethernet, and not other interfaces such as wifi. You may also use the special strings "localhost" or "loopback" to specify that the socket only accept connections from the local machine.

To accept connections on any interface pass nil, or simply use the acceptOnPort:error: method.

To accept on a certain address, pass the address to accept on. To accept on any address, pass nil or an empty string. To accept only connections from localhost pass "localhost" or "loopback".

Definition at line 745 of file AsyncTCPSocket.m.

5.8.2.2 - (BOOL) acceptOnPort: dummy(UInt16) *port* error:(NSError **) *errPtr*

Tells the socket to begin listening and accepting connections on the given port. When a connection comes in, the AsyncSocket instance will call the various delegate methods (see above). The socket will listen on all available interfaces (e.g. wifi, ethernet, etc)

Definition at line 735 of file AsyncTCPSocket.m.

5.8.2.3 - (NSString ∗) addressHost: dummy(CFDataRef) *cfaddr*

5.8.2.4 - (UInt16) addressPort: dummy(CFDataRef) *cfaddr*

5.8.2.5 - (BOOL) areStreamsConnected

5.8.2.6 - (BOOL) attachSocketsToRunLoop: dummy(NSRunLoop ∗) *runLoop* error:(NSError ∗∗) *errPtr*

5.8.2.7 - (BOOL) attachStreamsToRunLoop: dummy(NSRunLoop ∗) *runLoop* error:(NSError ∗∗) *errPtr*

5.8.2.8 - (BOOL) canSafelySetDelegate

Definition at line 476 of file AsyncTCPSocket.m.

5.8.2.9 - (void) close

5.8.2.10 - (void) closeWithError: dummy(NSError ∗) *err*

5.8.2.11 - (void) completeCurrentRead

5.8.2.12 - (void) completeCurrentWrite

5.8.2.13 - (BOOL) configureSocketAndReturnError: dummy(NSError ∗∗) *errPtr*

5.8.2.14 - (BOOL) configureStreamsAndReturnError: dummy(NSError ∗∗) *errPtr*

5.8.2.15 - (NSString ∗) connectedHost

Returns the local or remote host and port to which this socket is connected, or nil and 0 if not connected. The host will be an IP address.

Definition at line 1919 of file AsyncTCPSocket.m.

5.8.2.16 - (NSString ∗) connectedHost: dummy(CFSocketRef) *socket*

5.8.2.17 - (UInt16) connectedPort

Definition at line 1927 of file AsyncTCPSocket.m.

5.8.2.18 - (UInt16) connectedPort: dummy(CFSocketRef) *socket*

5.8.2.19 - (BOOL) connectSocketToAddress: dummy(NSData ∗) *remoteAddr* error:(NSError ∗∗) *errPtr*

5.8.2.20 - (BOOL) connectToAddress: dummy(NSData ∗) *remoteAddr* error:(NSError ∗∗) *errPtr*

Connects to the given address, specified as a sockaddr structure wrapped in a NSData object. For example, a NSData object returned from NSNetservice's addresses method.

If you have an existing struct sockaddr you can convert it to a NSData object like so: struct sockaddr sa -> NSData ∗dsa = [NSData dataWithBytes:&remoteAddr length:remoteAddr.sa_-len]; struct sockaddr ∗sa -> NSData ∗dsa = [NSData dataWithBytes:remoteAddr length:remoteAddr->sa_len];

Definition at line 980 of file AsyncTCPSocket.m.

5.8.2.21 - (BOOL) connectToAddress: dummy(NSData ∗) *remoteAddr* withTimeout:(NSTimeInterval) *timeout* error:(NSError ∗∗) *errPtr*

This method is the same as connectToAddress:error: with an additional timeout option. To not time out use a negative time interval, or simply use the connectToAddress:error: method.

This method creates an initial CFSocket to the given address. The connection is then opened, and the corresponding CFReadStream and CFWriteStream will be created from the low-level sockets after the connection succeeds.

Thus the delegate will have access to the CFSocket and CFSocketNativeHandle (BSD socket) prior to connection, specifically in the onSocketWillConnect: method.

Note: The NSData parameter is expected to be a sockaddr structure. For example, an NSData object returned from NSNetservice addresses method. If you have an existing struct sockaddr you can convert it to an NSData object like so: struct sockaddr sa -> NSData ∗dsa = [NSData dataWithBytes:&remoteAddr length:remoteAddr.sa_len]; struct sockaddr ∗sa -> NSData ∗dsa = [NSData dataWithBytes:remoteAddr length:remoteAddr->sa_len];

Definition at line 999 of file AsyncTCPSocket.m.

5.8.2.22 - (BOOL) connectToHost: dummy(NSString ∗) *hostname* onPort:(UInt16) *port* error:(NSError ∗∗) *errPtr*

Connects to the given host and port. The host may be a domain name (e.g. "deusty.com") or an IP address string (e.g. "192.168.0.2")

Definition at line 931 of file AsyncTCPSocket.m.

5.8.2.23 - (BOOL) connectToHost: dummy(NSString ∗) *hostname* onPort:(UInt16) *port* withTimeout:(NSTimeInterval) *timeout* error:(NSError ∗∗) *errPtr*

This method is the same as connectToHost:onPort:error: with an additional timeout option. To not time out use a negative time interval, or simply use the connectToHost:onPort:error: method.

This method creates an initial CFReadStream and CFWriteStream to the given host on

the given port. The connection is then opened, and the corresponding CFSocket will be extracted after the connection succeeds.

Thus the delegate will have access to the CFReadStream and CFWriteStream prior to connection, specifically in the onSocketWillConnect: method.

Definition at line 943 of file AsyncTCPSocket.m.

5.8.2.24 + (NSData ∗) CRData

Definition at line 2920 of file AsyncTCPSocket.m.

5.8.2.25 - (CFSocketRef) createAcceptSocketForAddress: dummy(NSData ∗) *addr* error:(NSError ∗∗) *errPtr*

5.8.2.26 - (BOOL) createSocketForAddress: dummy(NSData ∗) *remoteAddr* error:(NSError ∗∗) *errPtr*

5.8.2.27 - (BOOL) createStreamsFromNative: dummy(CFSocketNativeHandle) *native* error:(NSError ∗∗) *errPtr*

5.8.2.28 - (BOOL) createStreamsToHost: dummy(NSString ∗) *hostname* onPort:(UInt16) *port* error:(NSError ∗∗) *errPtr*

5.8.2.29 + (NSData ∗) CRLFData

Definition at line 2915 of file AsyncTCPSocket.m.

5.8.2.30 - (id) delegate

Use "canSafelySetDelegate" to see if there is any pending business (reads and writes) with the current delegate before changing it. It is, of course, safe to change the delegate before connecting or accepting connections.

Definition at line 466 of file AsyncTCPSocket.m.

5.8.2.31 - (NSString ∗) description

Definition at line 2081 of file AsyncTCPSocket.m.

5.8.2.32 - (void) disconnect

Disconnects immediately. Any pending reads or writes are dropped.

Definition at line 1633 of file AsyncTCPSocket.m.

5.8.2.33 - (void) disconnectAfterReading

Disconnects after all pending reads have completed. After calling this, the read and write methods will do nothing. The socket will disconnect even if there are still pending writes.

Diconnects after all pending reads have completed.

Definition at line 1641 of file AsyncTCPSocket.m.

5.8.2.34 - (void) disconnectAfterReadingAndWriting

Disconnects after all pending reads and writes have completed. After calling this, the read and write methods will do nothing.

Disconnects after all pending reads and writes have completed.

Definition at line 1661 of file AsyncTCPSocket.m.

5.8.2.35 - (void) disconnectAfterWriting

Disconnects after all pending writes have completed. After calling this, the read and write methods will do nothing. The socket will disconnect even if there are still pending reads.

Disconnects after all pending writes have completed.

Definition at line 1651 of file AsyncTCPSocket.m.

5.8.2.36 - (void) doAcceptWithSocket: dummy(CFSocketNativeHandle) *newSocket*

5.8.2.37 - (void) doBytesAvailable

5.8.2.38 - (void) doCFCallback: dummy(CFSocketCallBackType) *type* forSocket:(CFSocketRef) *sock* withAddress:(NSData ∗) *address* withData:(const void ∗) *pData*

5.8.2.39 - (void) doCFReadStreamCallback: dummy(CFStreamEventType) *type* forStream:(CFReadStreamRef) *stream*

5.8.2.40 - (void) doCFWriteStreamCallback: dummy(CFStreamEventType) *type* forStream:(CFWriteStreamRef) *stream*

5.8.2.41 - (void) doReadTimeout: dummy(NSTimer ∗) *timer*

5.8.2.42 - (void) doSendBytes

5.8.2.43 - (void) doSocketOpen: dummy(CFSocketRef) *sock* withCFSocketError:(CFSocketError) *err*

5.8.2.44 - (void) doStreamOpen

5.8.2.45 - (void) doWriteTimeout: dummy(NSTimer *) *timer*

5.8.2.46 - (void) emptyQueues

5.8.2.47 - (void) enablePreBuffering

For handling readDataToData requests, data is necessarily read from the socket in small increments. The performance can be much improved by allowing AsyncSocket to read larger chunks at a time and store any overflow in a small internal buffer. This is termed pre-buffering, as some data may be read for you before you ask for it. If you use readDataToData a lot, enabling pre-buffering will result in better performance, especially on the iPhone.

The default pre-buffering state is controlled by the DEFAULT_PREBUFFERING definition. It is highly recommended one leave this set to YES.

This method exists in case pre-buffering needs to be disabled by default for some reason. In that case, this method exists to allow one to easily enable pre-buffering when ready.

See the header file for a full explanation of pre-buffering.

Definition at line 603 of file AsyncTCPSocket.m.

5.8.2.48 - (void) endConnectTimeout

5.8.2.49 - (void) endCurrentRead

5.8.2.50 - (void) endCurrentWrite

5.8.2.51 - (NSError *) errorFromCFStreamError: dummy(CFStreamError) *err*

5.8.2.52 - (NSError *) getAbortError

5.8.2.53 - (CFReadStreamRef) getCFReadStream

Definition at line 489 of file AsyncTCPSocket.m.

5.8.2.54 - (CFSocketRef) getCFSocket

Definition at line 481 of file AsyncTCPSocket.m.

5.8.2.55 - (CFWriteStreamRef) getCFWriteStream

Definition at line 494 of file AsyncTCPSocket.m.

5.8.2.56 - (NSError ∗) getConnectTimeoutError

5.8.2.57 - (NSError ∗) getErrnoError

5.8.2.58 - (NSError ∗) getReadMaxedOutError

5.8.2.59 - (NSError ∗) getReadTimeoutError

5.8.2.60 - (NSError ∗) getSocketError

5.8.2.61 - (NSError ∗) getStreamError

5.8.2.62 - (NSError ∗) getWriteTimeoutError

5.8.2.63 - (id) init

Definition at line 386 of file AsyncTCPSocket.m.

5.8.2.64 - (id) initWithDelegate: dummy(id) *delegate*

Definition at line 391 of file AsyncTCPSocket.m.

5.8.2.65 - (id) initWithDelegate: dummy(id) *delegate* userData:(long) *userData*

Definition at line 397 of file AsyncTCPSocket.m.

5.8.2.66 - (BOOL) isConnected

Definition at line 1914 of file AsyncTCPSocket.m.

5.8.2.67 - (BOOL) isIPv4

Definition at line 2071 of file AsyncTCPSocket.m.

5.8.2.68 - (BOOL) isIPv6

Definition at line 2076 of file AsyncTCPSocket.m.

5.8.2.69 - (BOOL) isSocketConnected

5.8.2.70 + (NSData ∗) LFData

Definition at line 2925 of file AsyncTCPSocket.m.

5.8.2.71 - (NSString ∗) localHost

Definition at line 1935 of file AsyncTCPSocket.m.

5.8.2.72 - (NSString ∗) localHost: dummy(CFSocketRef) *socket*

5.8.2.73 - (UInt16) localPort

Definition at line 1943 of file AsyncTCPSocket.m.

5.8.2.74 - (UInt16) localPort: dummy(CFSocketRef) *socket*

5.8.2.75 - (void) maybeDequeueRead

5.8.2.76 - (void) maybeDequeueWrite

5.8.2.77 - (void) maybeScheduleDisconnect

5.8.2.78 - (void) maybeStartTLS

5.8.2.79 - (BOOL) moveToRunLoop: dummy(NSRunLoop ∗) *runLoop*

When you create an AsyncSocket, it is added to the runloop of the current thread. So for manually created sockets, it is easiest to simply create the socket on the thread you intend to use it.

If a new socket is accepted, the delegate method onSocket:wantsRunLoopForNewSocket: is called to allow you to place the socket on a separate thread. This works best in conjunction with a thread pool design.

If, however, you need to move the socket to a separate thread at a later time, this method may be used to accomplish the task.

This method must be called from the thread/runloop the socket is currently running on.

Note: After calling this method, all further method calls to this object should be done from the given runloop. Also, all delegate calls will be sent on the given runloop.

See the header file for a full explanation of this method.

Definition at line 611 of file AsyncTCPSocket.m.

5.8.2.80 - (void) onTLSStarted: dummy(BOOL) *flag*

5.8.2.81 - (BOOL) openStreamsAndReturnError: dummy(NSError ∗∗) *errPtr*

5.8.2.82 - (float) progressOfReadReturningTag: dummy(long *) *tag* bytesDone:(CFIndex *) *done* total:(CFIndex *) *total*

Returns progress of current read or write, from 0.0 to 1.0, or NaN if no read/write (use isnan() to check). "tag", "done" and "total" will be filled in if they aren't NULL.

Definition at line 499 of file AsyncTCPSocket.m.

5.8.2.83 - (float) progressOfWriteReturningTag: dummy(long *) *tag* bytesDone:(CFIndex *) *done* total:(CFIndex *) *total*

Definition at line 518 of file AsyncTCPSocket.m.

5.8.2.84 - (void) readDataToData: dummy(NSData *) *data* withTimeout:(NSTimeInterval) *timeout* maxLength:(CFIndex) *length* tag:(long) *tag*

Same as readDataToData:withTimeout:tag, with the additional restriction that the amount of data read may not surpass the given maxLength (specified in bytes).

If you pass a maxLength parameter that is less than the length of the data parameter, the method will do nothing, and the delegate will not be called.

If the max length is surpassed, it is treated the same as a timeout - the socket is closed.

Pass -1 as maxLength if no length restriction is desired, or simply use the readDataTo-Data:withTimeout:tag method.

Definition at line 2235 of file AsyncTCPSocket.m.

5.8.2.85 - (void) readDataToData: dummy(NSData *) *data* withTimeout:(NSTimeInterval) *timeout* tag:(long) *tag*

This reads bytes until (and including) the passed "data" parameter, which acts as a separator. The bytes and the separator are returned by the delegate method.

If you pass nil or zero-length data as the "data" parameter, the method will do nothing, and the delegate will not be called.

To read a line from the socket, use the line separator (e.g. CRLF for HTTP, see below) as the "data" parameter. Note that this method is not character-set aware, so if a separator can occur naturally as part of the encoding for a character, the read will prematurely end.

Definition at line 2230 of file AsyncTCPSocket.m.

5.8.2.86 - (void) readDataToLength: dummy(CFIndex) *length* withTimeout:(NSTimeInterval) *timeout* tag:(long) *tag*

This will read a certain number of bytes into memory, and call the delegate method when those bytes have been read. If there is an error, partially read data is lost. If the length is 0, this method does nothing and the delegate is not called.

Definition at line 2210 of file AsyncTCPSocket.m.

5.8.2.87 - (void) readDataWithTimeout: dummy(NSTimeInterval) *timeout* tag:(long) *tag*

Reads the first available bytes that become available on the socket.

Definition at line 2256 of file AsyncTCPSocket.m.

5.8.2.88 - (void) recoverUnreadData

5.8.2.89 - (void) runLoopAddSource: dummy(CFRunLoopSourceRef) *source*

5.8.2.90 - (void) runLoopAddTimer: dummy(NSTimer ∗) *timer*

5.8.2.91 - (void) runLoopRemoveSource: dummy(CFRunLoopSourceRef) *source*

5.8.2.92 - (void) runLoopRemoveTimer: dummy(NSTimer ∗) *timer*

5.8.2.93 - (void) runLoopUnscheduleReadStream

5.8.2.94 - (void) runLoopUnscheduleWriteStream

5.8.2.95 - (void) scheduleDequeueRead

5.8.2.96 - (void) scheduleDequeueWrite

5.8.2.97 - (void) setDelegate: dummy(id) *delegate*

Definition at line 471 of file AsyncTCPSocket.m.

5.8.2.98 - (BOOL) setRunLoopModes: dummy(NSArray ∗) *runLoopModes*

Allows you to configure which run loop modes the socket uses. The default set of run loop modes is NSDefaultRunLoopMode.

If you'd like your socket to continue operation during other modes, you may want to add modes such as NSModalPanelRunLoopMode or NSEventTrackingRunLoopMode. Or you may simply want to use NSRunLoopCommonModes.

Accepted sockets will automatically inherit the same run loop modes as the listening socket.

Note: NSRunLoopCommonModes is defined in 10.5. For previous versions one can use kCFRunLoopCommonModes.

See the header file for a full explanation of this method.

Definition at line 673 of file AsyncTCPSocket.m.

5.8.2.99 - (BOOL) setSocketFromStreamsAndReturnError: dummy(NSError ∗∗) *errPtr*

5.8.2.100 - (void) setUserData: dummy(long) *userData*

Definition at line 461 of file AsyncTCPSocket.m.

5.8.2.101 - (void) startConnectTimeout: dummy(NSTimeInterval) *timeout*

5.8.2.102 - (void) startTLS: dummy(NSDictionary ∗) *tlsSettings*

Secures the connection using SSL/TLS.

This method may be called at any time, and the TLS handshake will occur after all pending reads and writes are finished. This allows one the option of sending a protocol dependent StartTLS message, and queuing the upgrade to TLS at the same time, without having to wait for the write to finish. Any reads or writes scheduled after this method is called will occur over the secured connection.

The possible keys and values for the TLS settings are well documented. Some possible keys are:

- kCFStreamSSLLevel
- kCFStreamSSLAllowsExpiredCertificates
- kCFStreamSSLAllowsExpiredRoots
- kCFStreamSSLAllowsAnyRoot
- kCFStreamSSLValidatesCertificateChain
- kCFStreamSSLPeerName
- kCFStreamSSLCertificates
- kCFStreamSSLIsServer

Please refer to Apple's documentation for associated values, as well as other possible keys.

If you pass in nil or an empty dictionary, this method does nothing and the delegate will not be called.

Definition at line 2723 of file AsyncTCPSocket.m.

5.8.2.103 - (NSData ∗) unreadData

In the event of an error, this method may be called during onSocket:willDisconnectWithError: to read any data that's left on the socket.

Definition at line 1712 of file AsyncTCPSocket.m.

5.8.2.104 - (long) userData

Definition at line 456 of file AsyncTCPSocket.m.

5.8.2.105 - (void) writeData: dummy(NSData *) *data* withTimeout:(NSTimeInterval) *timeout* tag:(long) *tag*

Writes data to the socket, and calls the delegate when finished.

If you pass in nil or zero-length data, this method does nothing and the delegate will not be called.

Definition at line 2573 of file AsyncTCPSocket.m.

5.8.2.106 + (NSData *) ZeroData

Definition at line 2930 of file AsyncTCPSocket.m.

5.8.3 Field Documentation

5.8.3.1 - (NSMutableData*) partialReadBuffer [protected]

Definition at line 127 of file AsyncTCPSocket.h.

5.8.3.2 - (NSTimer*) theConnectTimer [protected]

Definition at line 122 of file AsyncTCPSocket.h.

5.8.3.3 - (CFSocketContext) theContext [protected]

Definition at line 119 of file AsyncTCPSocket.h.

5.8.3.4 - (AsyncReadPacket*) theCurrentRead [protected]

Definition at line 125 of file AsyncTCPSocket.h.

5.8.3.5 - (AsyncWritePacket*) theCurrentWrite [protected]

Definition at line 130 of file AsyncTCPSocket.h.

5.8.3.6 - (id) theDelegate [protected]

Definition at line 133 of file AsyncTCPSocket.h.

5.8.3.7 - (UInt16) theFlags [protected]

Definition at line 134 of file AsyncTCPSocket.h.

5.8.3.8 - (NSMutableArray*) theReadQueue [protected]

Definition at line 124 of file AsyncTCPSocket.h.

5.8.3.9 - (CFReadStreamRef) theReadStream [protected]

Definition at line 113 of file AsyncTCPSocket.h.

5.8.3.10 - (NSTimer*) theReadTimer [protected]

Definition at line 126 of file AsyncTCPSocket.h.

5.8.3.11 - (CFRunLoopRef) theRunLoop [protected]

Definition at line 118 of file AsyncTCPSocket.h.

5.8.3.12 - (NSArray*) theRunLoopModes [protected]

Definition at line 120 of file AsyncTCPSocket.h.

5.8.3.13 - (CFSocketRef) theSocket [protected]

Definition at line 111 of file AsyncTCPSocket.h.

5.8.3.14 - (CFSocketRef) theSocket6 [protected]

Definition at line 112 of file AsyncTCPSocket.h.

5.8.3.15 - (CFRunLoopSourceRef) theSource [protected]

Definition at line 116 of file AsyncTCPSocket.h.

5.8.3.16 - (CFRunLoopSourceRef) theSource6 [protected]

Definition at line 117 of file AsyncTCPSocket.h.

5.8.3.17 - (long) theUserData [protected]

Definition at line 136 of file AsyncTCPSocket.h.

5.8.3.18 - (NSMutableArray*) theWriteQueue [protected]

Definition at line 129 of file AsyncTCPSocket.h.

5.8.3.19 - (CFWriteStreamRef) theWriteStream [protected]

Definition at line 114 of file AsyncTCPSocket.h.

5.8.3.20 - (NSTimer*) theWriteTimer [protected]

Definition at line 131 of file AsyncTCPSocket.h.

The documentation for this class was generated from the following files:

- ZigPad/AsyncTCPSocket.h
- ZigPad/AsyncTCPSocket.m

5.9 AsyncUdpSocket Class Reference

```
#import <AsyncUdpSocket.h>
```

Public Member Functions

- (id) - init
- (id) - initWithDelegate:
- (id) - initWithDelegate:userData:
- (id) - initIPv4
- (id) - initIPv6
- (id) - delegate
- (void) - setDelegate:
- (long) - userData
- (void) - setUserData:
- (NSString *) - localHost
- (UInt16) - localPort
- (NSString *) - connectedHost
- (UInt16) - connectedPort
- (BOOL) - isConnected
- (BOOL) - isClosed
- (BOOL) - isIPv4
- (BOOL) - isIPv6

- (unsigned int) - maximumTransmissionUnit
- (BOOL) - bindToPort:error:
- (BOOL) - bindToAddress:port:error:
- (BOOL) - connectToHost:onPort:error:
- (BOOL) - connectToAddress:error:
- (BOOL) - joinMulticastGroup:error:
- (BOOL) - joinMulticastGroup:withAddress:error:
- (BOOL) - enableBroadcast:error:
- (BOOL) - sendData:withTimeout:tag:
- (BOOL) - sendData:toHost:port:withTimeout:tag:
- (BOOL) - sendData:toAddress:withTimeout:tag:
- (void) - receiveWithTimeout:tag:
- (void) - close
- (void) - closeAfterSending
- (void) - closeAfterReceiving
- (void) - closeAfterSendingAndReceiving
- (UInt32) - maxReceiveBufferSize
- (void) - setMaxReceiveBufferSize:
- (BOOL) - moveToRunLoop:
- (BOOL) - setRunLoopModes:
- (NSArray ∗) - runLoopModes
- (void) - runLoopAddSource:
- (void) - runLoopRemoveSource:
- (void) - runLoopAddTimer:
- (void) - runLoopRemoveTimer:
- (NSString ∗) - addressHost4:
- (NSString ∗) - addressHost6:
- (NSString ∗) - addressHost:
- (void) - emptyQueues
- (void) - closeSocket4
- (void) - closeSocket6
- (void) - maybeScheduleClose
- (NSError ∗) - getErrnoError
- (NSError ∗) - getSocketError
- (NSError ∗) - getIPv4UnavailableError
- (NSError ∗) - getIPv6UnavailableError
- (NSError ∗) - getSendTimeoutError
- (NSError ∗) - getReceiveTimeoutError
- (NSString ∗) - connectedHost:
- (UInt16) - connectedPort:
- (NSString ∗) - localHost:
- (UInt16) - localPort:
- (BOOL) - canAcceptBytes:
- (void) - scheduleDequeueSend
- (void) - maybeDequeueSend
- (void) - doSend:

- (void) - completeCurrentSend
- (void) - failCurrentSend:
- (void) - endCurrentSend
- (void) - doSendTimeout:
- (BOOL) - hasBytesAvailable:
- (void) - scheduleDequeueReceive
- (void) - maybeDequeueReceive
- (void) - doReceive4
- (void) - doReceive6
- (void) - doReceive:
- (BOOL) - maybeCompleteCurrentReceive
- (void) - failCurrentReceive:
- (void) - endCurrentReceive
- (void) - doReceiveTimeout:

Protected Attributes

- CFSocketRef theSocket4
- CFSocketRef theSocket6
- CFRunLoopSourceRef theSource4
- CFRunLoopSourceRef theSource6
- CFRunLoopRef theRunLoop
- CFSocketContext theContext
- NSArray * theRunLoopModes
- NSMutableArray * theSendQueue
- AsyncSendPacket * theCurrentSend
- NSTimer * theSendTimer
- NSMutableArray * theReceiveQueue
- AsyncReceivePacket * theCurrentReceive
- NSTimer * theReceiveTimer
- id theDelegate
- UInt16 theFlags
- long theUserData
- NSString * cachedLocalHost
- UInt16 cachedLocalPort
- NSString * cachedConnectedHost
- UInt16 cachedConnectedPort
- UInt32 maxReceiveBufferSize

5.9.1 Detailed Description

Definition at line 31 of file AsyncUdpSocket.h.

5.9.2 Member Function Documentation

5.9.2.1 - (NSString ∗) addressHost4: dummy(struct sockaddr_in ∗) *pSockaddr4*

5.9.2.2 - (NSString ∗) addressHost6: dummy(struct sockaddr_in6 ∗) *pSockaddr6*

5.9.2.3 - (NSString ∗) addressHost: dummy(struct sockaddr ∗) *pSockaddr*

5.9.2.4 - (BOOL) bindToAddress: dummy(NSString ∗) *host* port:(UInt16) *port* error:(NSError ∗∗) *errPtr*

Binds the underlying socket(s) to the given address and port. The sockets(s) will be able to receive data only on the given interface.

To receive data on any interface, pass nil or "". To receive data only on the loopback interface, pass "localhost" or "loopback".

On success, returns YES. Otherwise returns NO, and sets errPtr. If you don't care about the error, you can pass nil for errPtr.

Definition at line 774 of file AsyncUdpSocket.m.

5.9.2.5 - (BOOL) bindToPort: dummy(UInt16) *port* error:(NSError ∗∗) *errPtr*

Binds the UDP socket to the given port and optional address. Binding should be done for server sockets that receive data prior to sending it. Client sockets can skip binding, as the OS will automatically assign the socket an available port when it starts sending data.

You cannot bind a socket after its been connected. You can only bind a socket once. You can still connect a socket (if desired) after binding.

On success, returns YES. Otherwise returns NO, and sets errPtr. If you don't care about the error, you can pass nil for errPtr.

Binds the underlying socket(s) to the given port. The socket(s) will be able to receive data on any interface.

On success, returns YES. Otherwise returns NO, and sets errPtr. If you don't care about the error, you can pass nil for errPtr.

Definition at line 759 of file AsyncUdpSocket.m.

5.9.2.6 - (BOOL) canAcceptBytes: dummy(CFSocketRef) *sockRef*

5.9.2.7 - (void) close

Closes the socket immediately. Any pending send or receive operations are dropped.

Definition at line 1283 of file AsyncUdpSocket.m.

5.9.2.8 - (void) closeAfterReceiving

Closes after all pending receive operations have completed. After calling this, the sendData: and receive: methods will do nothing. In other words, you won't be able to add any more send or receive operations to the queue. The socket will close even if there are still pending send operations.

Definition at line 1311 of file AsyncUdpSocket.m.

5.9.2.9 - (void) closeAfterSending

Closes after all pending send operations have completed. After calling this, the sendData: and receive: methods will do nothing. In other words, you won't be able to add any more send or receive operations to the queue. The socket will close even if there are still pending receive operations.

Definition at line 1303 of file AsyncUdpSocket.m.

5.9.2.10 - (void) closeAfterSendingAndReceiving

Closes after all pending send and receive operations have completed. After calling this, the sendData: and receive: methods will do nothing. In other words, you won't be able to add any more send or receive operations to the queue.

Definition at line 1319 of file AsyncUdpSocket.m.

5.9.2.11 - (void) closeSocket4

5.9.2.12 - (void) closeSocket6

5.9.2.13 - (void) completeCurrentSend

5.9.2.14 - (NSString ∗) connectedHost

Returns the remote address info for the socket.

Note: Since UDP is connectionless by design, connected address info will not be available unless the socket is explicitly connected to a remote host/port

Definition at line 1445 of file AsyncUdpSocket.m.

5.9.2.15 - (NSString ∗) connectedHost: dummy(CFSocketRef) *socket*

5.9.2.16 - (UInt16) connectedPort

Definition at line 1455 of file AsyncUdpSocket.m.

5.9.2.17 - (UInt16) connectedPort: dummy(CFSocketRef) *socket*

5.9.2.18 - (BOOL) connectToAddress: dummy(NSData *) *remoteAddr* error:(NSError **) *errPtr*

Connects the underlying UDP socket to the remote address. If the address is an IPv4 address, the IPv4 socket is connected, and the IPv6 socket is invalidated and released. If the address is an IPv6 address, the IPv6 socket is connected, and the IPv4 socket is invalidated and released.

The address is a native address structure, as may be returned from API's such as Bonjour. An address may be created manually by simply wrapping a sockaddr_in or sockaddr_in6 in an NSData object.

On success, returns YES. Otherwise returns NO, and sets errPtr. If you don't care about the error, you can pass nil for errPtr.

Definition at line 980 of file AsyncUdpSocket.m.

5.9.2.19 - (BOOL) connectToHost: dummy(NSString *) *host* onPort:(UInt16) *port* error:(NSError **) *errPtr*

Connects the UDP socket to the given host and port. By design, UDP is a connectionless protocol, and connecting is not needed.

Choosing to connect to a specific host/port has the following effect:

- You will only be able to send data to the connected host/port.
- You will only be able to receive data from the connected host/port.
- You will receive ICMP messages that come from the connected host/port, such as "connection refused".

Connecting a UDP socket does not result in any communication on the socket. It simply changes the internal state of the socket.

You cannot bind a socket after its been connected. You can only connect a socket once.

On success, returns YES. Otherwise returns NO, and sets errPtr. If you don't care about the error, you can pass nil for errPtr.

Connects the underlying UDP socket to the given host and port. If an IPv4 address is resolved, the IPv4 socket is connected, and the IPv6 socket is invalidated and released. If an IPv6 address is resolved, the IPv6 socket is connected, and the IPv4 socket is invalidated and released.

On success, returns YES. Otherwise returns NO, and sets errPtr. If you don't care about the error, you can pass nil for errPtr.

Definition at line 880 of file AsyncUdpSocket.m.

5.9.2.20 - (id) delegate

Definition at line 335 of file AsyncUdpSocket.m.

5.9.2.21 - (void) doReceive4

5.9.2.22 - (void) doReceive6

5.9.2.23 - (void) doReceive: dummy(CFSocketRef) *sockRef*

5.9.2.24 - (void) doReceiveTimeout: dummy(NSTimer ∗) *timer*

5.9.2.25 - (void) doSend: dummy(CFSocketRef) *sockRef*

5.9.2.26 - (void) doSendTimeout: dummy(NSTimer ∗) *timer*

5.9.2.27 - (void) emptyQueues

5.9.2.28 - (BOOL) enableBroadcast: dummy(BOOL) *flag* error:(NSError ∗∗) *errPtr*

By default, the underlying socket in the OS will not allow you to send broadcast messages. In order to send broadcast messages, you need to enable this functionality in the socket.

A broadcast is a UDP message to addresses like "192.168.255.255" or "255.255.255.255" that is delivered to every host on the network. The reason this is generally disabled by default is to prevent accidental broadcast messages from flooding the network.

Definition at line 1206 of file AsyncUdpSocket.m.

5.9.2.29 - (void) endCurrentReceive

5.9.2.30 - (void) endCurrentSend

5.9.2.31 - (void) failCurrentReceive: dummy(NSError ∗) *error*

5.9.2.32 - (void) failCurrentSend: dummy(NSError ∗) *error*

5.9.2.33 - (NSError ∗) getErrnoError

5.9.2.34 - (NSError ∗) getIPv4UnavailableError

5.9.2.35 - (NSError ∗) getIPv6UnavailableError

5.9.2.36 - (NSError ∗) getReceiveTimeoutError

5.9.2.37 - (NSError ∗) getSendTimeoutError

5.9.2.38 - (NSError ∗) getSocketError

5.9.2.39 - (BOOL) hasBytesAvailable: dummy(CFSocketRef) *sockRef*

5.9.2.40 - (id) init

Creates new instances of AsyncUdpSocket.

Definition at line 293 of file AsyncUdpSocket.m.

5.9.2.41 - (id) initIPv4

Creates new instances of AsyncUdpSocket that support only IPv4 or IPv6. The other init methods will support both, unless specifically binded or connected to one protocol. If you know you'll only be using one protocol, these init methods may be a bit more efficient.

Definition at line 308 of file AsyncUdpSocket.m.

5.9.2.42 - (id) initIPv6

Definition at line 313 of file AsyncUdpSocket.m.

5.9.2.43 - (id) initWithDelegate: dummy(id) *delegate*

Definition at line 298 of file AsyncUdpSocket.m.

5.9.2.44 - (id) initWithDelegate: dummy(id) *delegate* userData:(long) *userData*

Definition at line 303 of file AsyncUdpSocket.m.

5.9.2.45 - (BOOL) isClosed

Returns whether or not this socket has been closed. The only way a socket can be closed is if you explicitly call one of the close methods.

Definition at line 1645 of file AsyncUdpSocket.m.

5.9.2.46 - (BOOL) isConnected

Returns whether or not this socket has been connected to a single host. By design, UDP is a connectionless protocol, and connecting is not needed. If connected, the socket will only be able to send/receive data to/from the connected host.

Definition at line 1635 of file AsyncUdpSocket.m.

5.9.2.47 - (BOOL) isIPv4

Returns whether or not this socket supports IPv4. By default this will be true, unless the socket is specifically initialized as IPv6 only, or is binded or connected to an IPv6

address.

Definition at line 1650 of file AsyncUdpSocket.m.

5.9.2.48 - (BOOL) isIPv6

Returns whether or not this socket supports IPv6. By default this will be true, unless the socket is specifically initialized as IPv4 only, or is binded or connected to an IPv4 address.

This method will also return false on platforms that do not support IPv6. Note: The iPhone does not currently support IPv6.

Definition at line 1655 of file AsyncUdpSocket.m.

5.9.2.49 - (BOOL) joinMulticastGroup: dummy(NSString ∗) *group* error:(NSError ∗∗) *errPtr*

Join multicast group

Group should be an IP address (eg "225.228.0.1")

Join multicast group

Group should be a multicast IP address (eg. "239.255.250.250" for IPv4). Address is local interface for IPv4, but currently defaults under IPv6.

Definition at line 1062 of file AsyncUdpSocket.m.

5.9.2.50 - (BOOL) joinMulticastGroup: dummy(NSString ∗) *group* withAddress:(NSString ∗) *interface* error:(NSError ∗∗) *errPtr*

Definition at line 1067 of file AsyncUdpSocket.m.

5.9.2.51 - (NSString ∗) localHost

Returns the local address info for the socket.

Note: Address info may not be available until after the socket has been bind'ed, or until after data has been sent.

Definition at line 1425 of file AsyncUdpSocket.m.

5.9.2.52 - (NSString ∗) localHost: dummy(CFSocketRef) *socket*

5.9.2.53 - (UInt16) localPort

Definition at line 1435 of file AsyncUdpSocket.m.

5.9.2.54 - (UInt16) localPort: dummy(CFSocketRef) *socket*

5.9.2.55 - (unsigned int) maximumTransmissionUnit

Returns the mtu of the socket. If unknown, returns zero.

Sending data larger than this may result in an error. This is an advanced topic, and one should understand the wide range of mtu's on networks and the internet. Therefore this method is only for reference and may be of little use in many situations.

Definition at line 1660 of file AsyncUdpSocket.m.

5.9.2.56 - (UInt32) maxReceiveBufferSize

Gets/Sets the maximum size of the buffer that will be allocated for receive operations. The default size is 9216 bytes.

The theoretical maximum size of any IPv4 UDP packet is UINT16_MAX = 65535. The theoretical maximum size of any IPv6 UDP packet is UINT32_MAX = 4294967295.

In practice, however, the size of UDP packets will be much smaller. Indeed most protocols will send and receive packets of only a few bytes, or will set a limit on the size of packets to prevent fragmentation in the IP layer.

If you set the buffer size too small, the sockets API in the OS will silently discard any extra data, and you will not be notified of the error.

5.9.2.57 - (BOOL) maybeCompleteCurrentReceive

5.9.2.58 - (void) maybeDequeueReceive

5.9.2.59 - (void) maybeDequeueSend

5.9.2.60 - (void) maybeScheduleClose

5.9.2.61 - (BOOL) moveToRunLoop: dummy(NSRunLoop *) *runLoop*

When you create an AsyncUdpSocket, it is added to the runloop of the current thread. So it is easiest to simply create the socket on the thread you intend to use it.

If, however, you need to move the socket to a separate thread at a later time, this method may be used to accomplish the task.

This method must be called from the thread/runloop the socket is currently running on.

Note: After calling this method, all further method calls to this object should be done from the given runloop. Also, all delegate calls will be sent on the given runloop.

See the header file for a full explanation of this method.

Definition at line 416 of file AsyncUdpSocket.m.

5.9.2.62 - (void) receiveWithTimeout: dummy(NSTimeInterval) *timeout* tag:(long) *tag*

Asynchronously receives a single datagram packet.

If the receive succeeds, the onUdpSocket:didReceiveData:fromHost:port:tag delegate method will be called. Otherwise, a timeout will occur, and the onUdpSocket:didNotReceiveDataWithTag: delegate method will be called.

Definition at line 1978 of file AsyncUdpSocket.m.

5.9.2.63 - (void) runLoopAddSource: dummy(CFRunLoopSourceRef) *source*

5.9.2.64 - (void) runLoopAddTimer: dummy(NSTimer ∗) *timer*

5.9.2.65 - (NSArray ∗) runLoopModes

Returns the current run loop modes the AsyncSocket instance is operating in. The default set of run loop modes is NSDefaultRunLoopMode.

Definition at line 516 of file AsyncUdpSocket.m.

5.9.2.66 - (void) runLoopRemoveSource: dummy(CFRunLoopSourceRef) *source*

5.9.2.67 - (void) runLoopRemoveTimer: dummy(NSTimer ∗) *timer*

5.9.2.68 - (void) scheduleDequeueReceive

5.9.2.69 - (void) scheduleDequeueSend

5.9.2.70 - (BOOL) sendData: dummy(NSData ∗) *data* toAddress:(NSData ∗) *remoteAddr* withTimeout:(NSTimeInterval) *timeout* tag:(long) *tag*

Asynchronously sends the given data, with the given timeout and tag, to the given address.

This method cannot be used with a connected socket.

If data is nil or zero-length, this method does nothing and immediately returns NO. If the socket is connected, this method does nothing and immediately returns NO.

Definition at line 1741 of file AsyncUdpSocket.m.

5.9.2.71 - (BOOL) sendData: dummy(NSData ∗) *data* toHost:(NSString ∗) *host* port:(UInt16) *port* withTimeout:(NSTimeInterval) *timeout* tag:(long) *tag*

Asynchronously sends the given data, with the given timeout and tag, to the given host and port.

This method cannot be used with a connected socket.

If data is nil or zero-length, this method does nothing and immediately returns NO. If the socket is connected, this method does nothing and immediately returns NO. If unable to resolve host to a valid IPv4 or IPv6 address, this method returns NO.

Definition at line 1713 of file AsyncUdpSocket.m.

5.9.2.72 - (BOOL) sendData: dummy(NSData *) *data* withTimeout:(NSTimeInterval) *timeout* tag:(long) *tag*

Asynchronously sends the given data, with the given timeout and tag.

This method may only be used with a connected socket.

If data is nil or zero-length, this method does nothing and immediately returns NO. If the socket is not connected, this method does nothing and immediately returns NO.

Definition at line 1695 of file AsyncUdpSocket.m.

5.9.2.73 - (void) setDelegate: dummy(id) *delegate*

Definition at line 340 of file AsyncUdpSocket.m.

5.9.2.74 - (void) setMaxReceiveBufferSize: dummy(UInt32) *max*

Definition at line 408 of file AsyncUdpSocket.m.

5.9.2.75 - (BOOL) setRunLoopModes: dummy(NSArray *) *runLoopModes*

Allows you to configure which run loop modes the socket uses. The default set of run loop modes is NSDefaultRunLoopMode.

If you'd like your socket to continue operation during other modes, you may want to add modes such as NSModalPanelRunLoopMode or NSEventTrackingRunLoopMode. Or you may simply want to use NSRunLoopCommonModes.

Note: NSRunLoopCommonModes is defined in 10.5. For previous versions one can use kCFRunLoopCommonModes.

See the header file for a full explanation of this method.

Definition at line 467 of file AsyncUdpSocket.m.

5.9.2.76 - (void) setUserData: dummy(long) *userData*

Definition at line 350 of file AsyncUdpSocket.m.

5.9.2.77 - (long) userData

Definition at line 345 of file AsyncUdpSocket.m.

5.9.3 Field Documentation

5.9.3.1 - (NSString*) cachedConnectedHost [protected]

Definition at line 58 of file AsyncUdpSocket.h.

5.9.3.2 - (UInt16) cachedConnectedPort [protected]

Definition at line 59 of file AsyncUdpSocket.h.

5.9.3.3 - (NSString*) cachedLocalHost [protected]

Definition at line 55 of file AsyncUdpSocket.h.

5.9.3.4 - (UInt16) cachedLocalPort [protected]

Definition at line 56 of file AsyncUdpSocket.h.

5.9.3.5 - (UInt32) maxReceiveBufferSize [protected]

Definition at line 61 of file AsyncUdpSocket.h.

5.9.3.6 - (CFSocketContext) theContext [protected]

Definition at line 39 of file AsyncUdpSocket.h.

5.9.3.7 - (AsyncReceivePacket*) theCurrentReceive [protected]

Definition at line 47 of file AsyncUdpSocket.h.

5.9.3.8 - (AsyncSendPacket*) theCurrentSend [protected]

Definition at line 43 of file AsyncUdpSocket.h.

5.9.3.9 - (id) theDelegate [protected]

Definition at line 50 of file AsyncUdpSocket.h.

5.9.3.10 - (UInt16) theFlags [protected]

Definition at line 51 of file AsyncUdpSocket.h.

5.9.3.11 - (NSMutableArray∗) theReceiveQueue [protected]

Definition at line 46 of file AsyncUdpSocket.h.

5.9.3.12 - (NSTimer∗) theReceiveTimer [protected]

Definition at line 48 of file AsyncUdpSocket.h.

5.9.3.13 - (CFRunLoopRef) theRunLoop [protected]

Definition at line 38 of file AsyncUdpSocket.h.

5.9.3.14 - (NSArray∗) theRunLoopModes [protected]

Definition at line 40 of file AsyncUdpSocket.h.

5.9.3.15 - (NSMutableArray∗) theSendQueue [protected]

Definition at line 42 of file AsyncUdpSocket.h.

5.9.3.16 - (NSTimer∗) theSendTimer [protected]

Definition at line 44 of file AsyncUdpSocket.h.

5.9.3.17 - (CFSocketRef) theSocket4 [protected]

Definition at line 33 of file AsyncUdpSocket.h.

5.9.3.18 - (CFSocketRef) theSocket6 [protected]

Definition at line 34 of file AsyncUdpSocket.h.

5.9.3.19 - (CFRunLoopSourceRef) theSource4 [protected]

Definition at line 36 of file AsyncUdpSocket.h.

5.9.3.20 - (CFRunLoopSourceRef) theSource6 [protected]

Definition at line 37 of file AsyncUdpSocket.h.

5.9.3.21 - (long) theUserData [protected]

Definition at line 53 of file AsyncUdpSocket.h.

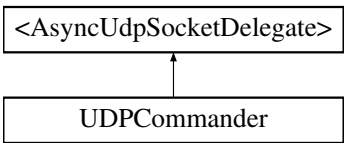
The documentation for this class was generated from the following files:

- ZigPad/AsyncUdpSocket.h
- ZigPad/AsyncUdpSocket.m

5.10 <AsyncUdpSocketDelegate> Protocol Reference

`#import <AsyncUdpSocket.h>`

Inheritance diagram for <AsyncUdpSocketDelegate>:



Public Member Functions

- (void) - onUdpSocket:didSendDataWithTag:
- (void) - onUdpSocket:didNotSendDataWithTag:dueToError:
- (BOOL) - onUdpSocket:didReceiveData:withTag:fromHost:port:
- (void) - onUdpSocket:didNotReceiveDataWithTag:dueToError:
- (void) - onUdpSocketDidClose:

5.10.1 Detailed Description

Definition at line 320 of file AsyncUdpSocket.h.

5.10.2 Member Function Documentation

5.10.2.1 - (void) onUdpSocket: dummy(AsyncUdpSocket *) *sock* didNotReceiveDataWithTag:(long) *tag* dueToError:(NSError *) *error* [optional]

Called if an error occurs while trying to receive a requested datagram. This is generally due to a timeout, but could potentially be something else if some kind of OS error occurred.

5.10.2.2 - (void) onUdpSocket: dummy(AsyncUdpSocket *) sock didNotSendDataWithTag:(long) tag dueToError:(NSError *) error [optional]

Called if an error occurs while trying to send a datagram. This could be due to a timeout, or something more serious such as the data being too large to fit in a sigle packet.

5.10.2.3 - (BOOL) onUdpSocket: dummy(AsyncUdpSocket *) sock didReceiveData:(NSData *) data withTag:(long) tag fromHost:(NSString *) host port:(UInt16) port [optional]

Called when the socket has received the requested datagram.

Due to the nature of UDP, you may occasionally receive undesired packets. These may be rogue UDP packets from unknown hosts, or they may be delayed packets arriving after retransmissions have already occurred. It's important these packets are properly ignored, while not interfering with the flow of your implementation. As an aid, this delegate method has a boolean return value. If you ever need to ignore a received packet, simply return NO, and AsyncUdpSocket will continue as if the packet never arrived. That is, the original receive request will still be queued, and will still timeout as usual if a timeout was set. For example, say you requested to receive data, and you set a timeout of 500 milliseconds, using a tag of 15. If rogue data arrives after 250 milliseconds, this delegate method would be invoked, and you could simply return NO. If the expected data then arrives within the next 250 milliseconds, this delegate method will be invoked, with a tag of 15, just as if the rogue data never appeared.

Under normal circumstances, you simply return YES from this method.

5.10.2.4 - (void) onUdpSocket: dummy(AsyncUdpSocket *) sock didSendDataWithTag:(long) tag [optional]

Called when the datagram with the given tag has been sent.

5.10.2.5 - (void) onUdpSocketDidClose: dummy(AsyncUdpSocket *) sock [optional]

Called when the socket is closed. A socket is only closed if you explicitly call one of the close methods.

The documentation for this protocol was generated from the following file:

- ZigPad/AsyncUdpSocket.h

5.11 AsyncWritePacket Class Reference

Public Member Functions

- (id) - initWithData:timeout:tag:

Data Fields

- NSData * buffer
- CFIndex bytesDone
- long tag
- NSTimeInterval timeout

5.11.1 Detailed Description

The AsyncWritePacket encompasses the instructions for any given write.

Definition at line 312 of file AsyncTCPSocket.m.

5.11.2 Member Function Documentation

5.11.2.1 - (id) initWithData: dummy(NSData *) *d* timeout:(NSTimeInterval) *t* tag:(long) *i*

Definition at line 325 of file AsyncTCPSocket.m.

5.11.3 Field Documentation

5.11.3.1 - (NSData*) buffer

Definition at line 315 of file AsyncTCPSocket.m.

5.11.3.2 - (CFIndex) bytesDone

Definition at line 316 of file AsyncTCPSocket.m.

5.11.3.3 - (long) tag

Definition at line 317 of file AsyncTCPSocket.m.

5.11.3.4 - (NSTimeInterval) timeout

Definition at line 318 of file AsyncTCPSocket.m.

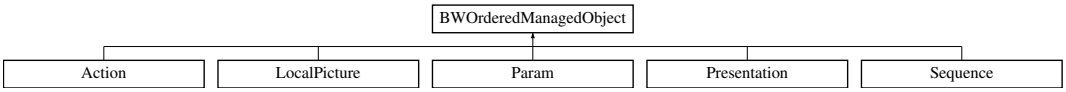
The documentation for this class was generated from the following file:

- ZigPad/AsyncTCPSocket.m

5.12 BWOrderedManagedObject Class Reference

`#import <BWOrderedManagedObject.h>`

Inheritance diagram for BWOrderedManagedObject:



Public Member Functions

- (NSArray ∗) - orderedKeys
- (NSString ∗) - orderingKeyForRelationshipKey:
- (NSArray ∗) - orderingForKey:
- (void) - setOrdering:forKey:
- (NSArray ∗) - orderedValueForKey:
- (NSMutableArray ∗) - mutableOrderedValueForKey:
- (unsigned) - countOfOrderedValueForKey:
- (NSManagedObject ∗) - objectInOrderedValueForKey:atIndex:
- (void) - insertObject:inOrderedValueForKey:atIndex:
- (void) - removeObjectFromOrderedValueForKey:atIndex:
- (void) - replaceObjectInOrderedValueForKey:atIndex:withObject:
- (void) - moveObjectsInOrderedValueForKey:fromIndexes:toIndex:

5.12.1 Detailed Description

A subclass of NSManagedObject that provide support for imposing an ordering on to-many relationships

By default, to-many Core Data relationships are unordered, but there are some cases where you want to impose a specific ordering on a relationship. The classic example of this is iTunes, where each playlist can have a list of tracks, but those tracks can be put in a specific order by the user.

BWOrderedManagedObject manages this by using an additional property to store the ordering for each relationship that you want to be ordered. By default, the string "Ordering" is appended to the name of the relationship key to form the key that the ordering data is stored in (e.g. the ordering for the "tracks" relationship is stored under the key "tracksOrdering")

BWOrderedManagedObject also provides a set of methods that allow for inserting, deleting, and rearranging objects in a specific order.

Definition at line 35 of file BWOrderedManagedObject.h.

5.12.2 Member Function Documentation

5.12.2.1 - (unsigned) countOfOrderedValueForKey: dummy(NSString∗) *key*

Definition at line 195 of file BWOrderedManagedObject.m.

5.12.2.2 - (void) insertObject: dummy(NSManagedObject∗) *object* inOrderedValueForKey:(NSString∗) *key* atIndex:(NSUInteger) *index*

Definition at line 210 of file BWOrderedManagedObject.m.

5.12.2.3 - (void) moveObjectsInOrderedValueForKey: dummy(NSString∗) *key* fromIndexes:(NSIndexSet∗) *indexes* toIndex:(unsigned) *newIndex*

moveObjectsInOrderedValueForKey:fromIndexes:toIndex: Moves a set of objects to a different position in the array This is a convenience method for the typical case where a user is rearranging objects in a relationship by drag and drop in a table view or other ordered view.

Parameters

key	The relationship whose objects you want to move
indexes	The indexes of the objects that are being moved (i.e. the dragged selection in the table view)
newIndex	The index to which the objects will be moved (i.e. the drop position in the table view)

Definition at line 246 of file BWOrderedManagedObject.m.

5.12.2.4 - (NSMutableArray ∗) mutableOrderedValueForKey: dummy(NSString∗) *key*

mutableOrderedValueForKey: Returns a mutable proxy array that can be used to manipulate a relationship The mutable array returned by this method will pass through any operations to the underlying relationship, in a similar manner to mutableArrayValueForKey:

Parameters

key	The key for the relationship whose objects you want to manipulate

Returns

A mutable proxy array

Definition at line 190 of file BWOrderedManagedObject.m.

5.12.2.5 - (NSManagedObject ∗) objectInOrderedValueForKey: dummy(NSString∗) *key* atIndex:(NSUInteger) *index*

Definition at line 200 of file BWOrderedManagedObject.m.

5.12.2.6 - (NSArray ∗) orderedKeys

orderedKeys

Returns the list of to-many relationships that have an ordering

By default, this method will compile a list of keys based on the entity definition of the object. Any to-many relationship which has the corresponding ordering key also defined will be returned. This method can be overridden if you want to use some other method of determining which of your relationships should be ordered.

Definition at line 145 of file BWOrderedManagedObject.m.

5.12.2.7 - (NSArray ∗) orderedValueForKey: dummy(NSString∗) *key*

orderedValueForKey: Returns the objects in a relationship in order This method returns an array of the NSManagedObjects in the given relationship, in the order specified by the relationship's ordering key.

Parameters

key	The key for the relationship whose objects you want to obtain

Returns

The ordered array of objects for the relationship

Definition at line 185 of file BWOrderedManagedObject.m.

5.12.2.8 - (NSArray ∗) orderingForKey: dummy(NSString∗) *key*

orderingForKey Returns the ordering for a given relationship

The ordering for a relationship is stored as an array of NSURLs. Each URL corresponds to the URIRepresentation of an object in the relationship, and the order of the URLs in the array determines the order of the objects.

Definition at line 162 of file BWOrderedManagedObject.m.

5.12.2.9 - (NSString ∗) orderingKeyForRelationshipKey: dummy(NSString∗) *key*

Returns the key used to store the ordering for a given relationship

By default, this method simply appends "Ordering" to the given key to form the ordering key. You can override this method if you wish to use a different technique to map from relationship to ordering.

Definition at line 140 of file BWOrderedManagedObject.m.

5.12.2.10 - (void) removeObjectFromOrderedValueForKey: dummy(NSString∗) *key* atIndex:(NSUInteger) *index*

Definition at line 221 of file BWOrderedManagedObject.m.

5.12.2.11 - (void) replaceObjectInOrderedValueForKey: dummy(NSString∗) *key* atIndex:(NSUInteger) *index* withObject:(NSManagedObject∗) *newObject*

Definition at line 233 of file BWOrderedManagedObject.m.

5.12.2.12 - (void) setOrdering: dummy(NSArray∗) *newOrdering* forKey:(NSString∗) *key*

setOrdering:forKey: Sets a new ordering for a given relationship

Sets a new ordering for a relationship. You typically don't need to call this method directly, as the various insert/delete methods will usually suffice. The newOrdering array should contain NSURL objects containing the URIRepresentation of the objects in the relationship, in the order you want them to be.

Definition at line 176 of file BWOrderedManagedObject.m.

The documentation for this class was generated from the following files:

- ZigPad/coredata/BWOrderedManagedObject.h
- ZigPad/coredata/BWOrderedManagedObject.m

5.13 BWOrderedValueProxy Class Reference

Public Member Functions

- (id) - initWithContainer:key:

Protected Attributes

- BWOrderedManagedObject ∗ container
- NSString ∗ key

5.13.1 Detailed Description

Definition at line 11 of file BWOrderedManagedObject.m.

5.13.2 Member Function Documentation

5.13.2.1 - (id) initWithContainer: dummy(BWOrderedManagedObject*) *inContainer* key:(NSString*) *inKey*

Definition at line 28 of file BWOrderedManagedObject.m.

5.13.3 Field Documentation

5.13.3.1 - (BWOrderedManagedObject*) container [protected]

Definition at line 13 of file BWOrderedManagedObject.m.

5.13.3.2 - (NSString*) key [protected]

Definition at line 14 of file BWOrderedManagedObject.m.

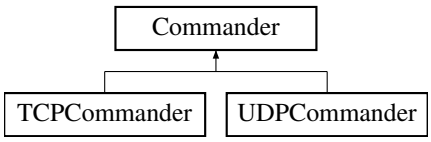
The documentation for this class was generated from the following file:

- ZigPad/coredata/BWOrderedManagedObject.m

5.14 Commander Class Reference

`#import <Commander.h>`

Inheritance diagram for Commander:



Public Member Functions

- (void) - sendString:
- (void) - sendAction:

Static Public Member Functions

- (Commander *) + defaultCommander
- (void) + close

5.14.1 Detailed Description

Allows to send strings over TCP or UDP.

Definition at line 22 of file Commander.h.

5.14.2 Member Function Documentation

5.14.2.1 + (void) close

Close the default commander, e.g. if settings have changed.

Definition at line 38 of file Commander.m.

5.14.2.2 + (Commander ∗) defaultCommander

Get the default commander instance.

Definition at line 23 of file Commander.m.

5.14.2.3 - (void) sendAction: dummy(Action∗) *action*

Send a command parameter of an action.

This is a helper method, which will look for an Param with key "command" and send that as a string.

Parameters

action	The action object of which the command Param should be sent.

Definition at line 61 of file Commander.m.

5.14.2.4 - (void) sendString: dummy(NSString∗) *message*

Send a string.

This is an abstract method and must be overriden by the actual implementation.

Parameters

message	The string that should be sent.

Definition at line 56 of file Commander.m.

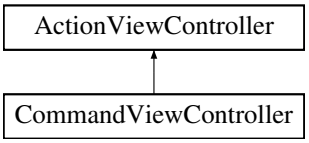
The documentation for this class was generated from the following files:

- ZigPad/Commander.h
- ZigPad/Commander.m

5.15 CommandViewController Class Reference

`#import <CommandViewController.h>`

Inheritance diagram for CommandViewController:



Properties

- IBOutlet UIImageView * repeatToggle

5.15.1 Detailed Description

Definition at line 13 of file CommandViewController.h.

5.15.2 Property Documentation

5.15.2.1 - (IBOutlet UIImageView*) repeatToggle [read, write, retain]

Definition at line 18 of file CommandViewController.h.

The documentation for this class was generated from the following file:

- ZigPad/CommandViewController.h

5.16 Config Class Reference

`#import <Config.h>`

Public Member Functions

- (void) - addAction:
- (void) - addParam:
- (void) - addSequence:
- (void) - addActionRef:
- (void) - addPresentation:
- (void) - addSequenceRef:
- (void) - addServer:
- (void) - saveToDB

- (void) - clearDB
- (void) - printDB
- (void) - rollback

5.16.1 Detailed Description

This Class is used to handle events called from Importer Class.

The methods decide how to persist contents (attributes) from the actual XML tag with CoreData.

It is very important that the methods are called in the correct order, specifically, the addActionRef() and addSequenceRef() methods must only be called after the corresponding actions and sequences have been added.

Definition at line 21 of file Config.h.

5.16.2 Member Function Documentation

5.16.2.1 - (void) addAction: dummy(NSDictionary∗) *attrib*

Analyzes an action tag and puts attributes into the core database.

Parameters

attrib	A dictionary with all attributes of the xml tag.

Definition at line 164 of file Config.m.

5.16.2.2 - (void) addActionRef: dummy(NSDictionary∗) *attrib*

Creates a reference to an already existing action for the last added sequence.

Parameters

attrib	A dictionary with all attributes of the xml tag.

Definition at line 231 of file Config.m.

5.16.2.3 - (void) addParam: dummy(NSDictionary∗) *attrib*

Analyzes a param tag and puts attributes into the core database.

Parameters

attrib	A dictionary with all attributes of the xml tag.

Definition at line 180 of file Config.m.

5.16.2.4 - (void) addPresentation: dummy(NSDictionary*) *attrib*

Analyzes a presentation tag and puts attributes into the core database.

Parameters

attrib	A dictionary with all attributes of the xml tag.

Definition at line 249 of file Config.m.

5.16.2.5 - (void) addSequence: dummy(NSDictionary*) *attrib*

Analyzes a sequence tag and puts attributes into the core database.

Parameters

attrib	A dictionary with all attributes of the xml tag.

Definition at line 207 of file Config.m.

5.16.2.6 - (void) addSequenceRef: dummy(NSDictionary*) *attrib*

Creates a reference to an already existing sequence for the last added presentation.

Parameters

attrib	A dictionary with all attributes of the xml tag.

Definition at line 262 of file Config.m.

5.16.2.7 - (void) addServer: dummy(NSDictionary*) *attrib*

Analyzes a server tag and stores the settings.

The settings are stored through the ZigPadSettings class.

Parameters

attrib	A dictionary with all attributes of the xml tag.

Definition at line 283 of file Config.m.

5.16.2.8 - (void) clearDB

Clear the database.

This must be called before updating the database to avoid duplicates.

Definition at line 137 of file Config.m.

5.16.2.9 - (void) printDB

Prints the current database information into the log.

Definition at line 304 of file Config.m.

5.16.2.10 - (void) rollback

Abort the current import and return to the previous state.

Definition at line 338 of file Config.m.

5.16.2.11 - (void) saveToDB

Persist the added information into the database.

Definition at line 294 of file Config.m.

The documentation for this class was generated from the following files:

- ZigPad/Config.h
- ZigPad/Config.m

5.17 Database Class Reference

```
#import <Database.h>
```

Public Member Functions

- (NSURL *) - applicationDocumentsDirectory
- (id) - createEntity:

Static Public Member Functions

- (Database *) + sharedInstance

Properties

- NSManagedObjectContext * managedObjectContext
- NSManagedObjectModel * managedObjectModel
- NSPersistentStoreCoordinator * persistentStoreCoordinator

5.17.1 Detailed Description

Definition at line 13 of file Database.h.

5.17.2 Member Function Documentation

5.17.2.1 - (NSURL ∗) applicationDocumentsDirectory

Returns the URL to the application's Documents directory.

Definition at line 146 of file Database.m.

5.17.2.2 - (id) createEntity: dummy(NSString ∗) *name*

Definition at line 151 of file Database.m.

5.17.2.3 + (Database ∗) sharedInstance

Definition at line 21 of file Database.m.

5.17.3 Property Documentation

5.17.3.1 - (NSManagedObjectContext ∗) managedObjectContext [read, retain]

Returns the managed object context for the application. If the context doesn't already exist, it is created and bound to the persistent store coordinator for the application.

Definition at line 23 of file Database.h.

5.17.3.2 - (NSManagedObjectModel ∗) managedObjectModel [read, retain]

Returns the managed object model for the application. If the model doesn't already exist, it is created from the application's model.

Definition at line 24 of file Database.h.

5.17.3.3 - (NSPersistentStoreCoordinator ∗) persistentStoreCoordinator [read, retain]

Returns the persistent store coordinator for the application. If the coordinator doesn't already exist, it is created and the application's store added to it.

Definition at line 25 of file Database.h.

The documentation for this class was generated from the following files:

- ZigPad/coredata/Database.h
- ZigPad/coredata/Database.m

5.18 DataCache Class Reference

`#import <DataCache.h>`

Public Member Functions

- (id) - initWithCapacity:
- (id) - objectForKey:
- (void) - setObject:forKey:

Protected Attributes

- int fCapacity
- NSMutableDictionary * fDictionary
- NSMutableArray * fAge

5.18.1 Detailed Description

Definition at line 41 of file DataCache.h.

5.18.2 Member Function Documentation

5.18.2.1 - (id) initWithCapacity: dummy(int) *cap*

Definition at line 50 of file DataCache.m.

5.18.2.2 - (id) objectForKey: dummy(id) *key*

Definition at line 73 of file DataCache.m.

5.18.2.3 - (void) setObject: dummy(id) *value* forKey:(id) *key*

Definition at line 86 of file DataCache.m.

5.18.3 Field Documentation

5.18.3.1 - (NSMutableArray*) fAge [protected]

Definition at line 45 of file DataCache.h.

5.18.3.2 - (int) fCapacity [protected]

Definition at line 43 of file DataCache.h.

5.18.3.3 - (NSMutableDictionary*) fDictionary [protected]

Definition at line 44 of file DataCache.h.

The documentation for this class was generated from the following files:

- ZigPad/DataCache.h
- ZigPad/DataCache.m

5.19 FavoritesViewController Class Reference

```
#import <FavoritesViewController.h>
```

Properties

- IBOutlet UIView * mySubview
- IBOutlet UILabel * favoritesLabel
- int startIndex
- NSArray * favorites

5.19.1 Detailed Description

Definition at line 20 of file FavoritesViewController.h.

5.19.2 Property Documentation

5.19.2.1 - (NSArray*) favorites [read, write, retain]

Definition at line 27 of file FavoritesViewController.h.

5.19.2.2 - (IBOutlet UILabel*) favoritesLabel [read, write, retain]

Definition at line 25 of file FavoritesViewController.h.

5.19.2.3 - (IBOutlet UIView*) mySubview [read, write, retain]

Definition at line 24 of file FavoritesViewController.h.

5.19.2.4 - (int) startIndex [read, write, assign]

Definition at line 26 of file FavoritesViewController.h.

The documentation for this class was generated from the following file:

- ZigPad/FavoritesViewController.h

5.20 FlowCoverRecord Class Reference

Public Member Functions

- (id) - initWithTexture:

Properties

- GLuint texture

5.20.1 Detailed Description

Definition at line 88 of file FlowCoverView.m.

5.20.2 Member Function Documentation

5.20.2.1 - (id) initWithTexture: dummy(GLuint) *t*

Definition at line 99 of file FlowCoverView.m.

5.20.3 Property Documentation

5.20.3.1 - (GLuint) texture [read, write, assign]

Definition at line 90 of file FlowCoverView.m.

The documentation for this class was generated from the following file:

- ZigPad/FlowCoverView.m

5.21 FlowCoverView Class Reference

```
#import <FlowCoverView.h>
```

Public Member Functions

- (void) - draw

Protected Attributes

- NSTimer * timer
- double startTime
- double startOff
- double startPos
- double startSpeed
- double runDelta
- BOOL touchFlag
- CGPoint startTouch
- double lastPos
- IBOutlet id< FlowCoverViewDelegate > delegate
- DataCache * cache
- GLint backingWidth
- GLint backingHeight
- EAGLContext * context
- GLuint viewRenderbuffer
- GLuint viewFramebuffer
- GLuint depthRenderbuffer

Properties

- id< FlowCoverViewDelegate > delegate
- double offset

5.21.1 Detailed Description

Definition at line 55 of file FlowCoverView.h.

5.21.2 Member Function Documentation

5.21.2.1 - (void) draw

Definition at line 411 of file FlowCoverView.m.

5.21.3 Field Documentation

5.21.3.1 - (GLint) backingHeight [protected]

Definition at line 78 of file FlowCoverView.h.

5.21.3.2 - (GLint) backingWidth [protected]

Definition at line 77 of file FlowCoverView.h.

5.21.3.3 - (DataCache*) cache [protected]

Definition at line 74 of file FlowCoverView.h.

5.21.3.4 - (EAGLContext*) context [protected]

Definition at line 79 of file FlowCoverView.h.

5.21.3.5 - (IBOutlet id<FlowCoverViewDelegate>) delegate [protected]

Definition at line 72 of file FlowCoverView.h.

5.21.3.6 - (GLuint) depthRenderbuffer [protected]

Definition at line 81 of file FlowCoverView.h.

5.21.3.7 - (double) lastPos [protected]

Definition at line 69 of file FlowCoverView.h.

5.21.3.8 - (double) runDelta [protected]

Definition at line 65 of file FlowCoverView.h.

5.21.3.9 - (double) startOff [protected]

Definition at line 62 of file FlowCoverView.h.

5.21.3.10 - (double) startPos [protected]

Definition at line 63 of file FlowCoverView.h.

5.21.3.11 - (double) startSpeed [protected]

Definition at line 64 of file FlowCoverView.h.

5.21.3.12 - (double) startTime [protected]

Definition at line 61 of file FlowCoverView.h.

5.21.3.13 - (CGPoint) startTouch [protected]

Definition at line 67 of file FlowCoverView.h.

5.21.3.14 - (NSTimer*) timer [protected]

Definition at line 60 of file FlowCoverView.h.

5.21.3.15 - (BOOL) touchFlag [protected]

Definition at line 66 of file FlowCoverView.h.

5.21.3.16 - (GLuint) viewFramebuffer [protected]

Definition at line 80 of file FlowCoverView.h.

5.21.3.17 - (GLuint) viewRenderbuffer [protected]

Definition at line 80 of file FlowCoverView.h.

5.21.4 Property Documentation

5.21.4.1 - (id<FlowCoverViewDelegate>) delegate [read, write, assign]

Definition at line 84 of file FlowCoverView.h.

5.21.4.2 - (double) offset [read, write, assign]

Definition at line 58 of file FlowCoverView.h.

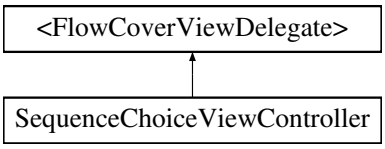
The documentation for this class was generated from the following files:

- ZigPad/FlowCoverView.h
- ZigPad/FlowCoverView.m

5.22 <FlowCoverViewDelegate> Protocol Reference

`#import <FlowCoverView.h>`

Inheritance diagram for <FlowCoverViewDelegate>:



Public Member Functions

- (int) - flowCoverNumberImages:
- (UIImage ∗) - flowCover:cover:
- (void) - flowCover:didSelect:
- (void) - flowCover:highlighted:

5.22.1 Detailed Description

Definition at line 98 of file FlowCoverView.h.

5.22.2 Member Function Documentation

5.22.2.1 - (UIImage ∗) flowCover: dummy(FlowCoverView ∗) *view* cover:(int) *cover*

5.22.2.2 - (void) flowCover: dummy(FlowCoverView ∗) *view* didSelect:(int) *cover*

5.22.2.3 - (void) flowCover: dummy(FlowCoverView ∗) *view* highlighted:(int) *cover*

5.22.2.4 - (int) flowCoverNumberImages: dummy(FlowCoverView ∗) *view*

The documentation for this protocol was generated from the following file:

- ZigPad/FlowCoverView.h

5.23 HudDemoAppDelegate Class Reference

```
#import <HudDemoAppDelegate.h>
```

Protected Attributes

- UIWindow ∗ window
- UINavigationController ∗ navController

Properties

- IBOutlet UIWindow * window
- IBOutlet UINavigationController * navController

5.23.1 Detailed Description

Definition at line 13 of file HudDemoAppDelegate.h.

5.23.2 Field Documentation

5.23.2.1 - (UINavigationController*) navController [protected]

Definition at line 15 of file HudDemoAppDelegate.h.

5.23.2.2 - (UIWindow*) window [protected]

Definition at line 14 of file HudDemoAppDelegate.h.

5.23.3 Property Documentation

5.23.3.1 - (IBOutlet UINavigationController*) navController [read, write, retain]

Definition at line 19 of file HudDemoAppDelegate.h.

5.23.3.2 - (IBOutlet UIWindow*) window [read, write, retain]

Definition at line 18 of file HudDemoAppDelegate.h.

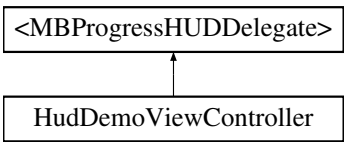
The documentation for this class was generated from the following file:

- ZigPad/MBProgressHUD/Demo/Classes/HudDemoAppDelegate.h

5.24 HudDemoViewController Class Reference

```
#import <HudDemoViewController.h>
```

Inheritance diagram for HudDemoViewController:



Public Member Functions

- (IBAction) - showSimple:
- (IBAction) - showWithLabel:
- (IBAction) - showWithDetailsLabel:
- (IBAction) - showWithLabelDeterminate:
- (IBAction) - showWithCustomView:
- (IBAction) - showWithLabelMixed:
- (IBAction) - showUsingBlocks:
- (IBAction) - showOnWindow:
- (void) - myTask
- (void) - myProgressTask
- (void) - myMixedTask

Protected Attributes

- MBProgressHUD * HUD

5.24.1 Detailed Description

Definition at line 12 of file HudDemoViewController.h.

5.24.2 Member Function Documentation

5.24.2.1 - (void) myMixedTask

Definition at line 213 of file HudDemoViewController.m.

5.24.2.2 - (void) myProgressTask

Definition at line 203 of file HudDemoViewController.m.

5.24.2.3 - (void) myTask

Definition at line 195 of file HudDemoViewController.m.

5.24.2.4 - (IBAction) showOnWindow: dummy(id) *sender*

Definition at line 176 of file HudDemoViewController.m.

5.24.2.5 - (IBAction) showSimple: dummy(id) *sender*

Definition at line 46 of file HudDemoViewController.m.

5.24.2.6 - (IBAction) showUsingBlocks: dummy(id) *sender*

Definition at line 157 of file HudDemoViewController.m.

5.24.2.7 - (IBAction) showWithCustomView: dummy(id) *sender*

Definition at line 115 of file HudDemoViewController.m.

5.24.2.8 - (IBAction) showWithDetailsLabel: dummy(id) *sender*

Definition at line 79 of file HudDemoViewController.m.

5.24.2.9 - (IBAction) showWithLabel: dummy(id) *sender*

Definition at line 63 of file HudDemoViewController.m.

5.24.2.10 - (IBAction) showWithLabelDeterminate: dummy(id) *sender*

Definition at line 96 of file HudDemoViewController.m.

5.24.2.11 - (IBAction) showWithLabelMixed: dummy(id) *sender*

Definition at line 141 of file HudDemoViewController.m.

5.24.3 Field Documentation

5.24.3.1 - (MBProgressHUD*) HUD [protected]

Definition at line 13 of file HudDemoViewController.h.

The documentation for this class was generated from the following files:

- ZigPad/MBProgressHUD/Demo/Classes/HudDemoViewController.h
- ZigPad/MBProgressHUD/Demo/Classes/HudDemoViewController.m

5.25 ImageDownloader Class Reference

`#import <ImageDownloader.h>`

Static Public Member Functions

- (NSData ∗) + downloadImage:

5.25.1 Detailed Description

Definition at line 12 of file ImageDownloader.h.

5.25.2 Member Function Documentation

5.25.2.1 + (NSData ∗) downloadImage: dummy(NSString∗) *url*

Definition at line 14 of file ImageDownloader.m.

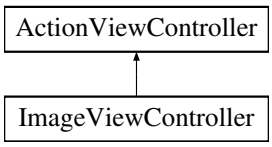
The documentation for this class was generated from the following files:

- ZigPad/ImageDownloader.h
- ZigPad/ImageDownloader.m

5.26 ImageViewController Class Reference

`#import <ImageViewController.h>`

Inheritance diagram for ImageViewController:



Properties

- IBOutlet UIImageView ∗ myImageView

5.26.1 Detailed Description

Definition at line 14 of file ImageViewController.h.

5.26.2 Property Documentation

5.26.2.1 - (IBOutlet UIImageView*) myImageView [read, write, retain]

Definition at line 18 of file ImageViewController.h.

The documentation for this class was generated from the following file:

- ZigPad/ImageViewController.h

5.27 Importer Class Reference

`#import <Importer.h>`

Public Member Functions

- (bool) - parseXMLFile:

5.27.1 Detailed Description

Definition at line 12 of file Importer.h.

5.27.2 Member Function Documentation

5.27.2.1 - (bool) parseXMLFile: dummy(NSString*) *fileName*

Definition at line 30 of file Importer.m.

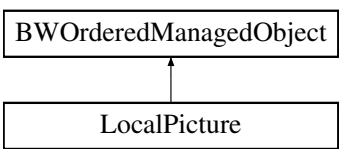
The documentation for this class was generated from the following files:

- ZigPad/Importer.h
- ZigPad/Importer.m

5.28 LocalPicture Class Reference

`#import <LocalPicture.h>`

Inheritance diagram for LocalPicture:



Properties

- NSData * picture
- Sequence * sequence
- Param * param

5.28.1 Detailed Description

Definition at line 15 of file LocalPicture.h.

5.28.2 Property Documentation

5.28.2.1 - (Param*) param [read, write, retain]

Definition at line 20 of file LocalPicture.h.

5.28.2.2 - (NSData*) picture [read, write, retain]

Definition at line 18 of file LocalPicture.h.

5.28.2.3 - (Sequence*) sequence [read, write, retain]

Definition at line 19 of file LocalPicture.h.

The documentation for this class was generated from the following file:

- ZigPad/coredata/LocalPicture.h

5.29 MBProgressHUD Class Reference

```
#import <MBProgressHUD.h>
```

Public Member Functions

- (id) - initWithWindow:
- (id) - initWithView:
- (void) - show:
- (void) - hide:
- (void) - showWhileExecuting:onTarget:withObject:animated:
- (void) - hideUsingAnimation:
- (void) - showUsingAnimation:
- (void) - fillRoundedRect:inContext:
- (void) - done

- (void) - updateLabelText:
- (void) - updateDetailsLabelText:
- (void) - updateProgress
- (void) - updateIndicators
- (void) - handleGraceTimer:
- (void) - handleMinShowTimer:
- (void) - setTransformForCurrentOrientation:

Static Public Member Functions

- (MBProgressHUD *) + showHUDAddedTo:animated:
- (BOOL) + hideHUDForView:animated:

Protected Attributes

- SEL methodForExecution
- id targetForExecution
- id objectForExecution
- BOOL useAnimation
- float width
- float height
- NSTimer * graceTimer
- NSTimer * minShowTimer
- NSDate * showStarted
- UIView * indicator
- UILabel * label
- UILabel * detailsLabel
- BOOL isFinished
- CGAffineTransform rotationTransform

Properties

- UIView * customView
- MBProgressHUDMode mode
- MBProgressHUDAnimation animationType
- id< MBProgressHUDDelegate > delegate
- NSString * labelText
- NSString * detailsLabelText
- float opacity
- float xOffset
- float yOffset
- float margin
- float graceTime
- float minShowTime
- BOOL taskInProgress

- BOOL removeFromSuperViewOnHide
- UIFont * labelFont
- UIFont * detailsLabelFont
- float progress

5.29.1 Detailed Description

Displays a simple HUD window containing a progress indicator and two optional labels for short messages.

This is a simple drop-in class for displaying a progress HUD view similar to Apples private UIProgressHUD class. The MBProgressHUD window spans over the entire space given to it by the initWithFrame constructor and catches all user input on this region, thereby preventing the user operations on components below the view. The HUD itself is drawn centered as a rounded semi-transparent view witch resizes depending on the user specified content.

This view supports three modes of operation:

- MBProgressHUDModeIndeterminate - shows a UIActivityIndicatorView
- MBProgressHUDModeDeterminate - shows a custom round progress indicator (MBRoundProgressView)
- MBProgressHUDModeCustomView - shows an arbitrary, user specified view (

  See also

  customView)

  All three modes can have optional labels assigned:

- If the labelText property is set and non-empty then a label containing the provided content is placed below the indicator view.
- If also the detailsLabelText property is set then another label is placed below the first label.

Definition at line 97 of file MBProgressHUD.h.

5.29.2 Member Function Documentation

5.29.2.1 - (void) done

5.29.2.2 - (void) fillRoundedRect: dummy(CGRect) *rect* inContext:(CGContextRef) *context*

5.29.2.3 - (void) handleGraceTimer: dummy(NSTimer *) *theTimer*

5.29.2.4 - (void) handleMinShowTimer: dummy(NSTimer *) *theTimer*

5.29.2.5 - (void) hide: dummy(BOOL) *animated*

Hide the HUD, this still calls the hudWasHidden delegate. This is the counterpart of the hide: method. Use it to hide the HUD when your task completes.

Parameters

animated	If set to YES the HUD will disappear using the current animationType. If set to NO the HUD will not use animations while disappearing.

Definition at line 422 of file MBProgressHUD.m.

5.29.2.6 + (BOOL) hideHUDForView: dummy(UIView ∗) *view* animated:(BOOL) *animated*

Finds a HUD sibview and hides it. The counterpart to this method is showHUDAddedTo:animated:.

Parameters

view	The view that is going to be searched for a HUD subview.
animated	If set to YES the HUD will disappear using the current animationType. If set to NO the HUD will not use animations while disappearing.

Returns

YES if a HUD was found and removed, NO otherwise.

See also

+ hideHUDForView:animated:

Definition at line 200 of file MBProgressHUD.m.

5.29.2.7 - (void) hideUsingAnimation: dummy(BOOL) *animated*

5.29.2.8 - (id) initWithView: dummy(UIView ∗) *view*

A convenience constructor that initializes the HUD with the view's bounds. Calls the designated constructor with view.bounds as the parameter

Parameters

view	The view instance that will provide the bounds for the HUD. Should probably be the same instance as the HUD's superview (i.e., the view that the HUD will be added to).

Definition at line 225 of file MBProgressHUD.m.

5.29.2.9 - (id) initWithWindow: dummy(UIWindow ∗) *window*

A convenience constructor that initializes the HUD with the window's bounds. Calls the designated constructor with window.bounds as the parameter.

Parameters

window	The window instance that will provide the bounds for the HUD. Should probably be the same instance as the HUD's superview (i.e., the window that the HUD will be added to).

Definition at line 221 of file MBProgressHUD.m.

5.29.2.10 - (void) setTransformForCurrentOrientation: dummy(BOOL) *animated*

5.29.2.11 - (void) show: dummy(BOOL) *animated*

Display the HUD. You need to make sure that the main thread completes its run loop soon after this method call so the user interface can be updated. Call this method when your task is already set-up to be executed in a new thread (e.g., when using something like NSOperation or calling an asynchronous call like NSUrlRequest).

Parameters

animated	If set to YES the HUD will disappear using the current animationType. If set to NO the HUD will not use animations while disappearing.

Definition at line 404 of file MBProgressHUD.m.

5.29.2.12 + (MBProgressHUD ∗) showHUDAddedTo: dummy(UIView ∗) *view* animated:(BOOL) *animated*

Creates a new hud, adds it to provided view and shows it. The counterpart to this method is hideHUDForView:animated:.

Parameters

view	The view that the HUD will be added to
animated	If set to YES the HUD will disappear using the current animationType. If set to NO the HUD will not use animations while disappearing.

Returns

A reference to the created HUD.

See also

+ hideHUDForView:animated:

Definition at line 193 of file MBProgressHUD.m.

5.29.2.13 - (void) showUsingAnimation: dummy(BOOL) *animated*

5.29.2.14 - (void) showWhileExecuting: dummy(SEL) *method* onTarget:(id) *target* withObject:(id) *object* animated:(BOOL) *animated*

Shows the HUD while a background task is executing in a new thread, then hides the HUD.

This method also takes care of NSAutoreleasePools so your method does not have to be concerned with setting up a pool.

Parameters

method	The method to be executed while the HUD is shown. This method will be executed in a new thread.
target	The object that the target method belongs to.
object	An optional object to be passed to the method.
animated	If set to YES the HUD will disappear using the current animationType. If set to NO the HUD will not use animations while disappearing.

Definition at line 455 of file MBProgressHUD.m.

5.29.2.15 - (void) updateDetailsLabelText: dummy(NSString ∗) *newText*

5.29.2.16 - (void) updateIndicators

5.29.2.17 - (void) updateLabelText: dummy(NSString ∗) *newText*

5.29.2.18 - (void) updateProgress

5.29.3 Field Documentation

5.29.3.1 - (UILabel∗) detailsLabel [protected]

Definition at line 124 of file MBProgressHUD.h.

5.29.3.2 - (NSTimer∗) graceTimer [protected]

Definition at line 118 of file MBProgressHUD.h.

5.29.3.3 - (float) height [protected]

Definition at line 111 of file MBProgressHUD.h.

5.29.3.4 - (UIView∗) indicator [protected]

Definition at line 122 of file MBProgressHUD.h.

5.29.3.5 - (BOOL) isFinished [protected]

Definition at line 135 of file MBProgressHUD.h.

5.29.3.6 - (UILabel*) label [protected]

Definition at line 123 of file MBProgressHUD.h.

5.29.3.7 - (SEL) methodForExecution [protected]

Definition at line 102 of file MBProgressHUD.h.

5.29.3.8 - (NSTimer*) minShowTimer [protected]

Definition at line 119 of file MBProgressHUD.h.

5.29.3.9 - (id) objectForExecution [protected]

Definition at line 104 of file MBProgressHUD.h.

5.29.3.10 - (CGAffineTransform) rotationTransform [protected]

Definition at line 140 of file MBProgressHUD.h.

5.29.3.11 - (NSDate*) showStarted [protected]

Definition at line 120 of file MBProgressHUD.h.

5.29.3.12 - (id) targetForExecution [protected]

Definition at line 103 of file MBProgressHUD.h.

5.29.3.13 - (BOOL) useAnimation [protected]

Definition at line 105 of file MBProgressHUD.h.

5.29.3.14 - (float) width [protected]

Definition at line 110 of file MBProgressHUD.h.

5.29.4 Property Documentation

5.29.4.1 - (MBProgressHUDAnimation) animationType [read, write, assign]

The animation type that should be used when the HUD is shown and hidden.

See also

MBProgressHUDAnimation

Definition at line 100 of file MBProgressHUD.h.

5.29.4.2 - (UIView *) customView [read, write, retain]

The UIView (i.e., a UIIMageView) to be shown when the HUD is in MBProgressHUD-ModeCustomView. For best results use a 37 by 37 pixel view (so the bounds match the build in indicator bounds).

Definition at line 138 of file MBProgressHUD.h.

5.29.4.3 - (id< MBProgressHUDDelegate >) delegate [read, write, assign]

The HUD delegate object. If set the delegate will receive hudWasHidden callbacks when the HUD was hidden. The delegate should conform to the MBProgressHUDDelegate protocol and implement the hudWasHidden method. The delegate object will not be retained.

Definition at line 128 of file MBProgressHUD.h.

5.29.4.4 - (UIFont *) detailsLabelFont [read, write, retain]

Font to be used for the details label. Set this property if the default is not adequate.

Definition at line 133 of file MBProgressHUD.h.

5.29.4.5 - (NSString *) detailsLabelText [read, write, copy]

An optional details message displayed below the labelText message. This message is displayed only if the labelText property is also set and is different from an empty string ("").

Definition at line 130 of file MBProgressHUD.h.

5.29.4.6 - (float) graceTime [read, write, assign]

Definition at line 116 of file MBProgressHUD.h.

5.29.4.7 - (UIFont *) labelFont [read, write, retain]

Font to be used for the main label. Set this property if the default is not adequate.

Definition at line 132 of file MBProgressHUD.h.

5.29.4.8 - (NSString *) labelText [read, write, copy]

An optional short message to be displayed below the activity indicator. The HUD is automatically resized to fit the entire text. If the text is too long it will get clipped by displaying "..." at the end. If left unchanged or set to "", then no message is displayed.

Definition at line 129 of file MBProgressHUD.h.

5.29.4.9 - (float) margin [read, write, assign]

The amounth of space between the HUD edge and the HUD elements (labels, indicators or custom views).

Defaults to 20.0

Definition at line 113 of file MBProgressHUD.h.

5.29.4.10 - (float) minShowTime [read, write, assign]

The minimum time (in seconds) that the HUD is shown. This avoids the problem of the HUD being shown and than instantly hidden. Defaults to 0 (no minimum show time).

Definition at line 117 of file MBProgressHUD.h.

5.29.4.11 - (MBProgressHUDMode) mode [read, write, assign]

MBProgressHUD operation mode. Switches between indeterminate (MBProgressHUDModeIndeterminate) and determinate progress (MBProgressHUDModeDeterminate). The default is MBProgressHUDModeIndeterminate.

See also

MBProgressHUDMode

Definition at line 99 of file MBProgressHUD.h.

5.29.4.12 - (float) opacity [read, write, assign]

The opacity of the HUD window. Defaults to 0.9 (90% opacity).

Definition at line 131 of file MBProgressHUD.h.

5.29.4.13 - (float) progress [read, write, assign]

The progress of the progress indicator, from 0.0 to 1.0. Defaults to 0.0.

Definition at line 126 of file MBProgressHUD.h.

5.29.4.14 - (BOOL) removeFromSuperViewOnHide [read, write, assign]

Removes the HUD from it's parent view when hidden. Defaults to NO.

Definition at line 136 of file MBProgressHUD.h.

5.29.4.15 - (BOOL) taskInProgress [read, write, assign]

Indicates that the executed operation is in progress. Needed for correct graceTime operation. If you don't set a graceTime (different than 0.0) this does nothing. This property is automatically set when using showWhileExecuting:onTarget:withObject:animated:. When threading is done outside of the HUD (i.e., when the show: and hide: methods are used directly), you need to set this property when your task starts and completes in order to have normal graceTime functunality.

Definition at line 115 of file MBProgressHUD.h.

5.29.4.16 - (float) xOffset [read, write, assign]

The x-axis offset of the HUD relative to the centre of the superview.

Definition at line 108 of file MBProgressHUD.h.

5.29.4.17 - (float) yOffset [read, write, assign]

The y-ayis offset of the HUD relative to the centre of the superview.

Definition at line 107 of file MBProgressHUD.h.

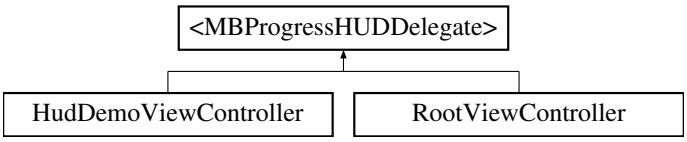
The documentation for this class was generated from the following files:

- ZigPad/MBProgressHUD/MBProgressHUD.h
- ZigPad/MBProgressHUD/MBProgressHUD.m

5.30 <MBProgressHUDDelegate> Protocol Reference

`#import <MBProgressHUD.h>`

Inheritance diagram for <MBProgressHUDDelegate>:



Public Member Functions

- (void) - hudWasHidden

5.30.1 Detailed Description

Definition at line 51 of file MBProgressHUD.h.

5.30.2 Member Function Documentation

5.30.2.1 - (void) hudWasHidden [required]

A callback function that is called after the HUD was fully hidden from the screen.

The documentation for this protocol was generated from the following file:

- ZigPad/MBProgressHUD/MBProgressHUD.h

5.31 MBRoundProgressView Class Reference

```
#import <MBProgressHUD.h>
```

Public Member Functions

- (id) - initWithDefaultSize

5.31.1 Detailed Description

A progress view for showing definite progress by filling up a circle (similar to the indicator for building in xcode).

Definition at line 67 of file MBProgressHUD.h.

5.31.2 Member Function Documentation

5.31.2.1 - (id) initWithDefaultSize

Create a 37 by 37 pixel indicator. This is the same size as used by the larger UIActivityIndicatorView.

Definition at line 631 of file MBProgressHUD.m.

The documentation for this class was generated from the following files:

- ZigPad/MBProgressHUD/MBProgressHUD.h
- ZigPad/MBProgressHUD/MBProgressHUD.m

5.32 NSObject(AsyncSocketDelegate) Class Reference

```
#import <AsyncTCPSocket.h>
```

Public Member Functions

- (void) - onSocket:willDisconnectWithError:
- (void) - onSocketDidDisconnect:
- (void) - onSocket:didAcceptNewSocket:
- (NSRunLoop *) - onSocket:wantsRunLoopForNewSocket:
- (BOOL) - onSocketWillConnect:
- (void) - onSocket:didConnectToHost:port:
- (void) - onSocket:didReadData:withTag:
- (void) - onSocket:didReadPartialDataOfLength:tag:
- (void) - onSocket:didWriteDataWithTag:
- (void) - onSocket:didSecure:

5.32.1 Detailed Description

Definition at line 32 of file AsyncTCPSocket.h.

5.32.2 Member Function Documentation

5.32.2.1 - (void) onSocket: dummy(AsyncTCPSocket *) sock didAcceptNewSocket:(AsyncTCPSocket *) newSocket

Called when a socket accepts a connection. Another socket is spawned to handle it. The new socket will have the same delegate and will call "onSocket:didConnectToHost:port:".

5.32.2.2 - (void) onSocket: dummy(AsyncTCPSocket *) *sock* didConnectToHost:(NSString *) *host* port:(UInt16) *port*

Called when a socket connects and is ready for reading and writing. The host parameter will be an IP address, not a DNS name.

5.32.2.3 - (void) onSocket: dummy(AsyncTCPSocket *) *sock* didReadData:(NSData *) *data* withTag:(long) *tag*

Called when a socket has completed reading the requested data into memory. Not called if there is an error.

5.32.2.4 - (void) onSocket: dummy(AsyncTCPSocket *) *sock* didReadPartialDataOfLength:(CFIndex) *partialLength* tag:(long) *tag*

Called when a socket has read in data, but has not yet completed the read. This would occur if using readToData: or readToLength: methods. It may be used to for things such as updating progress bars.

5.32.2.5 - (void) onSocket: dummy(AsyncTCPSocket *) *sock* didSecure:(BOOL) *flag*

Called after the socket has completed SSL/TLS negotiation. This method is not called unless you use the provided startTLS method.

5.32.2.6 - (void) onSocket: dummy(AsyncTCPSocket *) *sock* didWriteDataWithTag:(long) *tag*

Called when a socket has completed writing the requested data. Not called if there is an error.

5.32.2.7 - (NSRunLoop *) onSocket: dummy(AsyncTCPSocket *) *sock* wantsRunLoopForNewSocket:(AsyncTCPSocket *) *newSocket*

Called when a new socket is spawned to handle a connection. This method should return the run-loop of the thread on which the new socket and its delegate should operate. If omitted, [NSRunLoop currentRunLoop] is used.

5.32.2.8 - (void) onSocket: dummy(AsyncTCPSocket *) *sock* willDisconnectWithError:(NSError *) *err*

In the event of an error, the socket is closed. You may call "unreadData" during this callback to get the last bit of data off the socket. When connecting, this delegate method may be called before"onSocket:didAcceptNewSocket:" or "onSocket:didConnectToHost:".

5.32.2.9 - (void) onSocketDidDisconnect: dummy(AsyncTCPSocket ∗) *sock*

Called when a socket disconnects with or without error. If you want to release a socket after it disconnects, do so here. It is not safe to do that during "onSocket:willDisconnectWithError:".

5.32.2.10 - (BOOL) onSocketWillConnect: dummy(AsyncTCPSocket ∗) *sock*

Called when a socket is about to connect. This method should return YES to continue, or NO to abort. If aborted, will result in AsyncSocketCanceledError.

If the connectToHost:onPort:error: method was called, the delegate will be able to access and configure the CFReadStream and CFWriteStream as desired prior to connection.

If the connectToAddress:error: method was called, the delegate will be able to access and configure the CFSocket and CFSocketNativeHandle (BSD socket) as desired prior to connection. You will be able to access and configure the CFReadStream and CFWriteStream in the onSocket:didConnectToHost:port: method.

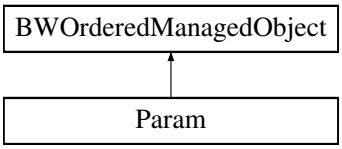
The documentation for this class was generated from the following file:

- ZigPad/AsyncTCPSocket.h

5.33 Param Class Reference

`#import <Param.h>`

Inheritance diagram for Param:



Properties

- NSString ∗ key
- NSString ∗ value
- Action ∗ action
- LocalPicture ∗ localPicture

5.33.1 Detailed Description

Definition at line 17 of file Param.h.

5.33.2 Property Documentation

5.33.2.1 - (Action*) action [read, write, retain]

Definition at line 22 of file Param.h.

5.33.2.2 - (NSString*) key [read, write, retain]

Definition at line 20 of file Param.h.

5.33.2.3 - (LocalPicture*) localPicture [read, write, retain]

Definition at line 23 of file Param.h.

5.33.2.4 - (NSString*) value [read, write, retain]

Definition at line 21 of file Param.h.

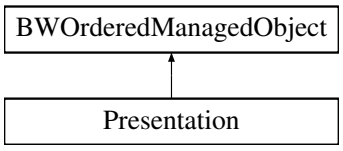
The documentation for this class was generated from the following file:

- ZigPad/coredata/Param.h

5.34 Presentation Class Reference

`#import <Presentation.h>`

Inheritance diagram for Presentation:



Public Member Functions

- (void) - addSequencesObject:
- (Action *) - getNextAction
- (Action *) - getPreviousAction
- (Action *) - jumpToSequence:
- (Action *) - jumpToAction:sequenceIndex:
- (void) - resetPresentation

Properties

- NSString * name
- NSString * comment
- id sequencesOrdering
- NSNumber * refId
- NSSet * sequences
- NSArray * orderedSequences
- Sequence * activeSequence
- Action * activeAction
- NSMutableArray * indexMapping
- int currentSequenceIndex
- int currentActionIndex

5.34.1 Detailed Description

Definition at line 15 of file Presentation.h.

5.34.2 Member Function Documentation

5.34.2.1 - (void) addSequencesObject: dummy(Sequence *) *value*

Definition at line 32 of file Presentation.m.

5.34.2.2 - (Action *) getNextAction

Definition at line 107 of file Presentation.m.

5.34.2.3 - (Action *) getPreviousAction

Definition at line 141 of file Presentation.m.

5.34.2.4 - (Action *) jumpToAction: dummy(int) *actionIndex* sequenceIndex:(int) *sequenceIndex*

Definition at line 172 of file Presentation.m.

5.34.2.5 - (Action *) jumpToSequence: dummy(int) *index*

Definition at line 168 of file Presentation.m.

5.34.2.6 - (void) resetPresentation

Definition at line 192 of file Presentation.m.

5.34.3 Property Documentation

5.34.3.1 - (Action*) activeAction [read, write, retain]

Definition at line 27 of file Presentation.h.

5.34.3.2 - (Sequence*) activeSequence [read, write, retain]

Definition at line 26 of file Presentation.h.

5.34.3.3 - (NSString*) comment [read, write, retain]

Definition at line 19 of file Presentation.h.

5.34.3.4 - (int) currentActionIndex [read, assign]

Definition at line 32 of file Presentation.h.

5.34.3.5 - (int) currentSequenceIndex [read, assign]

Definition at line 31 of file Presentation.h.

5.34.3.6 - (NSMutableArray*) indexMapping [read, write, retain]

Definition at line 29 of file Presentation.h.

5.34.3.7 - (NSString*) name [read, write, retain]

Definition at line 18 of file Presentation.h.

5.34.3.8 - (NSArray *) orderedSequences [read, assign]

Definition at line 24 of file Presentation.h.

5.34.3.9 - (NSNumber*) refId [read, write, retain]

Definition at line 21 of file Presentation.h.

5.34.3.10 - (NSSet*) sequences [read, write, retain]

Definition at line 23 of file Presentation.h.

5.34.3.11 - (id) sequencesOrdering [read, write, retain]

Definition at line 20 of file Presentation.h.

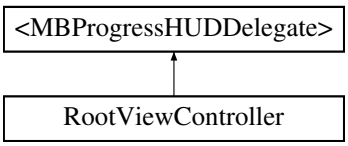
The documentation for this class was generated from the following files:

- ZigPad/coredata/Presentation.h
- ZigPad/coredata/Presentation.m

5.35 RootViewController Class Reference

`#import <RootViewController.h>`

Inheritance diagram for RootViewController:



Public Member Functions

- (IBAction) - update:
- (IBAction) - popupSettingView:
- (void) - runUpdate

Protected Attributes

- MBProgressHUD * HUD

Properties

- NSFetchedResultsController * fetchedResultsController
- Presentation * activePresentation

5.35.1 Detailed Description

The view controller for the initial table view.

Definition at line 18 of file RootViewController.h.

5.35.2 Member Function Documentation

5.35.2.1 - (IBAction) popupSettingView: dummy(id) *sender*

Displays the settings screen.

Parameters

sender	Indicates the button that is executing this action.

Definition at line 146 of file RootViewController.m.

5.35.2.2 - (void) runUpdate

Delegate callback for running the import.

Definition at line 176 of file RootViewController.m.

5.35.2.3 - (IBAction) update: dummy(id) *sender*

Starts the configuration import.

Parameters

sender	Indicates the button that is executing this action.

Definition at line 156 of file RootViewController.m.

5.35.3 Field Documentation

5.35.3.1 - (MBProgressHUD*) HUD [protected]

Reference of the MBProgressHUD for progress display during import.

Definition at line 23 of file RootViewController.h.

5.35.4 Property Documentation

5.35.4.1 - (Presentation*) activePresentation [read, write, retain]

Reference to the active presentation.

Definition at line 58 of file RootViewController.h.

5.35.4.2 - (NSFetchedResultsController *) fetchedResultsController [read, write, retain]

Reference to the results controller for the presentations.

Definition at line 53 of file RootViewController.h.

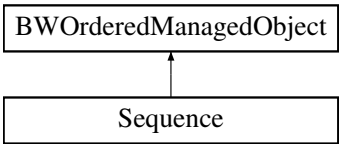
The documentation for this class was generated from the following files:

- ZigPad/RootViewController.h
- ZigPad/RootViewController.m

5.36 Sequence Class Reference

`#import <Sequence.h>`

Inheritance diagram for Sequence:



Public Member Functions

- (void) - addActionsObject:

Properties

- NSString * name
- NSString * command
- id actionsOrdering
- NSNumber * refId
- NSSet * presentations
- LocalPicture * icon
- NSSet * actions
- NSArray * orderedActions

5.36.1 Detailed Description

Definition at line 16 of file Sequence.h.

5.36.2 Member Function Documentation

5.36.2.1 - (void) addActionsObject: dummy(Action ∗) *value*

Definition at line 50 of file Sequence.m.

5.36.3 Property Documentation

5.36.3.1 - (NSSet∗) actions [read, write, retain]

Definition at line 26 of file Sequence.h.

5.36.3.2 - (id) actionsOrdering [read, write, retain]

Definition at line 21 of file Sequence.h.

5.36.3.3 - (NSString∗) command [read, write, retain]

Definition at line 20 of file Sequence.h.

5.36.3.4 - (LocalPicture∗) icon [read, write, retain]

Definition at line 25 of file Sequence.h.

5.36.3.5 - (NSString∗) name [read, write, retain]

Definition at line 19 of file Sequence.h.

5.36.3.6 - (NSArray ∗) orderedActions [read, assign]

Definition at line 27 of file Sequence.h.

5.36.3.7 - (NSSet∗) presentations [read, write, retain]

Definition at line 24 of file Sequence.h.

5.36.3.8 - (NSNumber∗) refId [read, write, retain]

Definition at line 22 of file Sequence.h.

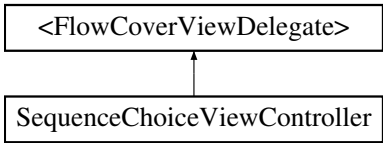
The documentation for this class was generated from the following files:

- ZigPad/coredata/Sequence.h
- ZigPad/coredata/Sequence.m

5.37 SequenceChoiceViewController Class Reference

`#import <SequenceChoiceViewController.h>`

Inheritance diagram for SequenceChoiceViewController:



Properties

- IBOutlet UILabel ∗ label
- Presentation ∗ presentation

5.37.1 Detailed Description

Definition at line 13 of file SequenceChoiceViewController.h.

5.37.2 Property Documentation

5.37.2.1 - (IBOutlet UILabel∗) label [read, write, retain]

Definition at line 19 of file SequenceChoiceViewController.h.

5.37.2.2 - (Presentation∗) presentation [read, write, retain]

Definition at line 20 of file SequenceChoiceViewController.h.

The documentation for this class was generated from the following file:

- ZigPad/SequenceChoiceViewController.h

5.38 SettingsViewController Class Reference

`#import <SettingsViewController.h>`

Public Member Functions

- (IBAction) - changeConfigHost:
- (IBAction) - switchSimulationMode:

- (IBAction) - switchVibrationMode:
- (IBAction) - switchSynchronizationMode:

Properties

- IBOutlet UISwitch * simSwitch
- IBOutlet UISwitch * vibrationSwitch
- IBOutlet UISwitch * synchronizeSwitch
- IBOutlet UITextField * textfield

5.38.1 Detailed Description

Definition at line 13 of file SettingsViewController.h.

5.38.2 Member Function Documentation

5.38.2.1 - (IBAction) changeConfigHost: dummy(id) *sender*

Definition at line 23 of file SettingsViewController.m.

5.38.2.2 - (IBAction) switchSimulationMode: dummy(id) *sender*

Definition at line 31 of file SettingsViewController.m.

5.38.2.3 - (IBAction) switchSynchronizationMode: dummy(id) *sender*

Definition at line 43 of file SettingsViewController.m.

5.38.2.4 - (IBAction) switchVibrationMode: dummy(id) *sender*

Definition at line 57 of file SettingsViewController.m.

5.38.3 Property Documentation

5.38.3.1 - (IBOutlet UISwitch*) simSwitch [read, write, retain]

Definition at line 17 of file SettingsViewController.h.

5.38.3.2 - (IBOutlet UISwitch*) synchronizeSwitch [read, write, retain]

Definition at line 19 of file SettingsViewController.h.

5.38.3.3 - (IBOutlet UITextField*) textfield [read, write, retain]

Definition at line 21 of file SettingsViewController.h.

5.38.3.4 - (IBOutlet UISwitch*) vibrationSwitch [read, write, retain]

Definition at line 18 of file SettingsViewController.h.

The documentation for this class was generated from the following files:

- ZigPad/SettingsViewController.h
- ZigPad/SettingsViewController.m

5.39 statement Class Reference

5.39.1 Detailed Description

Indicate which class should be used as the actual implementation.

When changing this, remember to also update the

The documentation for this class was generated from the following file:

- ZigPad/Commander.m

5.40 SyncEvent Class Reference

```
#import <SyncEvent.h>
```

Public Member Functions

- (id) - initWithBytes:
- (const uint8_t *) - bytes

Properties

- SyncEventCommand command
- uint8_t argument_upperByte
- uint8_t argument_lowerByte
- uint16_t argument
- SyncEventSwipeDirection direction
- int bytesLength
- SynchronizerConnection * connection

5.40.1 Detailed Description

A class that contaions information about a synchronization notification event.

Definition at line 44 of file SyncEvent.h.

5.40.2 Member Function Documentation

5.40.2.1 - (const uint8_t *) bytes

Returns a byte array that can be sent to another device.

Definition at line 52 of file SyncEvent.m.

5.40.2.2 - (id) initWithBytes: dummy(const uint8_t *) *bytes*

Create an instance based on a received bytes array.

Definition at line 20 of file SyncEvent.m.

5.40.3 Property Documentation

5.40.3.1 - (uint16_t) argument [read, write, assign]

The 16bit unsigned int argument.

The meaning depends on the used command. Some command also use each byte seperately, in that case, those bytes can be accessed directly through the argument_-lowerByte/upperByte properties.

Getter method for the argument property.

This is a virtual property which is created on-demand based on the upper and lower bytes.

Definition at line 70 of file SyncEvent.h.

5.40.3.2 - (uint8_t) argument_lowerByte [read, write, assign]

The lower byte of the 16bit argument.

Definition at line 61 of file SyncEvent.h.

5.40.3.3 - (uint8_t) argument_upperByte [read, write, assign]

The upper byte of the 16bit argument.

Definition at line 56 of file SyncEvent.h.

5.40.3.4 - (int) bytesLength [read, assign]

Length of the bytes array returned by bytes.

Definition at line 80 of file SyncEvent.h.

5.40.3.5 - (SyncEventCommand) command [read, write, assign]

The command of this sync event.

Definition at line 51 of file SyncEvent.h.

5.40.3.6 - (SynchronizerConnection*) connection [read, write, retain]

Definition at line 82 of file SyncEvent.h.

5.40.3.7 - (SyncEventSwipeDirection) direction [read, write, assign]

The swipe direction of the command, only relevant for JUMP commands.

Definition at line 75 of file SyncEvent.h.

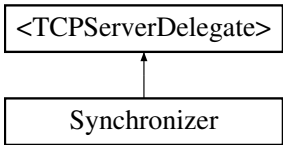
The documentation for this class was generated from the following files:

- ZigPad/SyncEvent.h
- ZigPad/SyncEvent.m

5.41 Synchronizer Class Reference

`#import <Synchronizer.h>`

Inheritance diagram for Synchronizer:



Public Member Functions

- (void) - lookForDevice
- (void) - stopCurrentResolve:
- (void) - fireSyncEvent:
- (void) - registerNotificationCenter
- (void) - unregisterNotificationCenter

Properties

- NSString * ownName

5.41.1 Detailed Description

This class manages connections to synchronized devices.

Each device is represented by a SynchronizerConnection instance, which can either be an inbound or an outbound connection.

Definition at line 18 of file Synchronizer.h.

5.41.2 Member Function Documentation

5.41.2.1 - (void) fireSyncEvent: dummy(NSNotification*) *notification*

Callback when recieving a sync event notification.

Definition at line 45 of file Synchronizer.m.

5.41.2.2 - (void) lookForDevice

Start looking for devices for synchronization.

Definition at line 90 of file Synchronizer.m.

5.41.2.3 - (void) registerNotificationCenter

Register for sync event notifications to be sent.

Definition at line 33 of file Synchronizer.m.

5.41.2.4 - (void) stopCurrentResolve: dummy(NSNetService *) *service*

Stop resolving a service.

Parameters

service	The service for which resolving should be stopped. If nil, resolving is stopped for services

Definition at line 104 of file Synchronizer.m.

5.41.2.5 - (void) unregisterNotificationCenter

Unregister from the notification center.

Definition at line 40 of file Synchronizer.m.

5.41.3 Property Documentation

5.41.3.1 - (NSString*) ownName [read, write, assign]

Name of the service propagated by this device.

Definition at line 51 of file Synchronizer.h.

The documentation for this class was generated from the following files:

- ZigPad/Synchronizer.h
- ZigPad/Synchronizer.m

5.42 SynchronizerConnection Class Reference

```
#import <SynchronizerConnection.h>
```

Public Member Functions

- (id) - initWithService:
- (id) - initWithStreams::
- (void) - send:
- (void) - sendTimed:

Properties

- NSInputStream * inStream
- NSOutputStream * outStream
- BOOL inReady
- BOOL outReady
- NSString * name

5.42.1 Detailed Description

This class represents a connection to another device.

Todo

: The class is currently only used either as a outbound or a inbound connection, but both a input and output stream is opened for both. This should be improved somehow.

Definition at line 20 of file SynchronizerConnection.h.

5.42.2 Member Function Documentation

5.42.2.1 - (id) initWithService: dummy(NSNetService ∗) *service*

Init with a NSNetService instance for an outbound connection.

Definition at line 37 of file SynchronizerConnection.m.

5.42.2.2 - (id) dummy(NSInputStream ∗) *istr* :(NSOutputStream ∗) *ostr*

Init with an input and output stream for an inbound connection.

Definition at line 70 of file SynchronizerConnection.m.

5.42.2.3 - (void) send: dummy(SyncEvent ∗) *event*

Send a sync event to the connected device.

Definition at line 97 of file SynchronizerConnection.m.

5.42.2.4 - (void) sendTimed: dummy(NSTimer ∗) *timer*

NSTimer callback used for automatically re-sending data when the connection is not yet ready.

Definition at line 113 of file SynchronizerConnection.m.

5.42.3 Property Documentation

5.42.3.1 - (BOOL) inReady [read, write, assign]

Indicates if the input stream is ready to be used.

Definition at line 37 of file SynchronizerConnection.h.

5.42.3.2 - (NSInputStream∗) inStream [read, write, retain]

Input stream.

Definition at line 27 of file SynchronizerConnection.h.

5.42.3.3 - (NSString∗) name [read, write, retain]

Name of service related to this connection.

This is only set for outbound connections.

Definition at line 49 of file SynchronizerConnection.h.

5.42.3.4 - (BOOL) outReady [read, write, assign]

Indicates if the output stream is ready to be used

Definition at line 42 of file SynchronizerConnection.h.

5.42.3.5 - (NSOutputStream*) outStream [read, write, retain]

Output stream.

Definition at line 32 of file SynchronizerConnection.h.

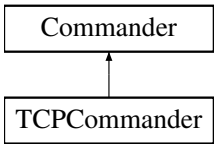
The documentation for this class was generated from the following files:

- ZigPad/SynchronizerConnection.h
- ZigPad/SynchronizerConnection.m

5.43 TCPCommander Class Reference

`#import <TCPCommander.h>`

Inheritance diagram for TCPCommander:



5.43.1 Detailed Description

TCP implementation of the abstract Commander class.

Definition at line 16 of file TCPCommander.h.

The documentation for this class was generated from the following file:

- ZigPad/TCPCommander.h

5.44 TCPServer Class Reference

`#import <TCPServer.h>`

Public Member Functions

- (BOOL) - start:

- (BOOL) - stop
- (BOOL) - enableBonjourWithDomain:applicationProtocol:name:
- (void) - disableBonjour

Static Public Member Functions

- (NSString ∗) + bonjourTypeFromIdentifier:

Properties

- id< TCPServerDelegate > delegate
- NSNetService ∗ netService
- uint16_t port

5.44.1 Detailed Description

Definition at line 69 of file TCPServer.h.

5.44.2 Member Function Documentation

5.44.2.1 + (NSString ∗) bonjourTypeFromIdentifier: dummy(NSString∗) *identifier*

Definition at line 271 of file TCPServer.m.

5.44.2.2 - (void) disableBonjour

Definition at line 256 of file TCPServer.m.

5.44.2.3 - (BOOL) enableBonjourWithDomain: dummy(NSString∗) *domain* applicationProtocol:(NSString∗) *protocol* name:(NSString∗) *name*

Definition at line 215 of file TCPServer.m.

5.44.2.4 - (BOOL) start: dummy(NSError ∗∗) *error*

Definition at line 113 of file TCPServer.m.

5.44.2.5 - (BOOL) stop

Definition at line 202 of file TCPServer.m.

5.44.3 Property Documentation

5.44.3.1 - (id<TCPServerDelegate>) delegate [read, write, assign]

Definition at line 83 of file TCPServer.h.

5.44.3.2 - (NSNetService*) netService [read, write, retain]

Definition at line 58 of file TCPServer.m.

5.44.3.3 - (uint16_t) port [read, write, assign]

Definition at line 59 of file TCPServer.m.

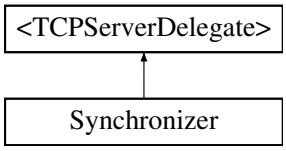
The documentation for this class was generated from the following files:

- ZigPad/TCPServer.h
- ZigPad/TCPServer.m

5.45 <TCPServerDelegate> Protocol Reference

`#import <TCPServer.h>`

Inheritance diagram for <TCPServerDelegate>:



Public Member Functions

- (void) - serverDidEnableBonjour:withName:
- (void) - server:didNotEnableBonjour:
- (void) - didAcceptConnectionForServer:inputStream:outputStream:

5.45.1 Detailed Description

Definition at line 61 of file TCPServer.h.

5.45.2 Member Function Documentation

5.45.2.1 - (void) didAcceptConnectionForServer: dummy(TCPServer ∗) *server* inputStream:(NSInputStream ∗) *istr* outputStream:(NSOutputStream ∗) *ostr* [optional]

5.45.2.2 - (void) server: dummy(TCPServer ∗) *server* didNotEnableBonjour:(NSDictionary ∗) *errorDict* [optional]

5.45.2.3 - (void) serverDidEnableBonjour: dummy(TCPServer ∗) *server* withName:(NSString ∗) *name* [optional]

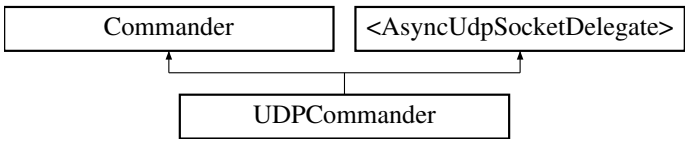
The documentation for this protocol was generated from the following file:

- ZigPad/TCPServer.h

5.46 UDPCommander Class Reference

`#import <UDPCommander.h>`

Inheritance diagram for UDPCommander:



5.46.1 Detailed Description

UDP implementation of the abstract Commander class.

Definition at line 15 of file UDPCommander.h.

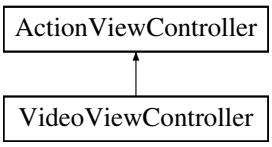
The documentation for this class was generated from the following file:

- ZigPad/UDPCommander.h

5.47 VideoViewController Class Reference

`#import <VideoViewController.h>`

Inheritance diagram for VideoViewController:



Protected Attributes

- MPMoviePlayerController * moviePlayer

5.47.1 Detailed Description

Definition at line 14 of file VideoViewController.h.

5.47.2 Field Documentation

5.47.2.1 - (MPMoviePlayerController*) moviePlayer [protected]

Definition at line 15 of file VideoViewController.h.

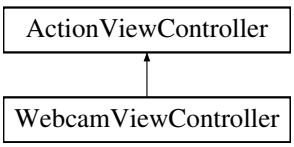
The documentation for this class was generated from the following file:

- ZigPad/VideoViewController.h

5.48 WebcamViewController Class Reference

`#import <WebCamViewController.h>`

Inheritance diagram for WebcamViewController:



Properties

- IBOutlet UIWebView * myWebView

5.48.1 Detailed Description

Definition at line 13 of file WebCamViewController.h.

5.48.2 Property Documentation

5.48.2.1 - (IBOutlet UIWebView*) myWebView [read, write, retain]

Definition at line 17 of file WebCamViewController.h.

The documentation for this class was generated from the following file:

- ZigPad/WebCamViewController.h

5.49 ZigPadAppDelegate Class Reference

```
#import <ZigPadAppDelegate.h>
```

Public Member Functions

- (void) - saveContext
- (NSURL *) - applicationDocumentsDirectory

Properties

- IBOutlet UIWindow * window
- IBOutlet UINavigationController * navigationController

5.49.1 Detailed Description

Definition at line 11 of file ZigPadAppDelegate.h.

5.49.2 Member Function Documentation

5.49.2.1 - (NSURL *) applicationDocumentsDirectory

Returns the URL to the application's Documents directory.

Definition at line 97 of file ZigPadAppDelegate.m.

5.49.2.2 - (void) saveContext

Definition at line 73 of file ZigPadAppDelegate.m.

5.49.3 Property Documentation

5.49.3.1 - (IBOutlet UINavigationController*) navigationController [read, write, retain]

Definition at line 20 of file ZigPadAppDelegate.h.

5.49.3.2 - (IBOutlet UIWindow*) window [read, write, retain]

Definition at line 15 of file ZigPadAppDelegate.h.

The documentation for this class was generated from the following files:

- ZigPad/ZigPadAppDelegate.h
- ZigPad/ZigPadAppDelegate.m

5.50 ZigPadSettings Class Reference

```
#import <ZigPadSettings.h>
```

Public Member Functions

- (void) - setIP:simulationMode:
- (void) - setPort:simulationMode:

Static Public Member Functions

- (ZigPadSettings *) + sharedInstance

Properties

- BOOL simulationMode
- NSString * configuratorURL
- UIColor * modeColor
- NSString * ip
- int port
- BOOL vibrationMode
- BOOL synchronizationMode

5.50.1 Detailed Description

Allows to access and store application settings.

Uses NSUserDefaults to store these permanently.

Definition at line 16 of file ZigPadSettings.h.

5.50.2 Member Function Documentation

5.50.2.1 - (void) setIP: dummy(NSString ∗) *ip* simulationMode:(BOOL) *simulationMode*

Allows to set the IP for a specific simulation mode.

Parameters

ip	The IP to be used for this simulation mode.
simulation-Mode	YES if this IP is for the simulation mode.

Setter method for the ip property.

Definition at line 128 of file ZigPadSettings.m.

5.50.2.2 - (void) setPort: dummy(int) *port* simulationMode:(BOOL) *simulationMode*

Allows to set the Port for a specific simulation mode.

Parameters

ip	The Port to be used for this simulation mode.
simulation-Mode	YES if this Port is for the simulation mode.

Setter method for the port property.

Definition at line 137 of file ZigPadSettings.m.

5.50.2.3 + (ZigPadSettings ∗) sharedInstance

Returns the shared instance of this class.

Definition at line 26 of file ZigPadSettings.m.

5.50.3 Property Documentation

5.50.3.1 - (NSString ∗) configuratorURL [read, write, assign]

Setting for the configurator URL.

Get method for the configurationURL property.

Definition at line 37 of file ZigPadSettings.h.

5.50.3.2 - (NSString ∗) ip [read, assign]

Setting for the HomeServer IP, depends on the simulation mode.

Getter method for the IP property.

Definition at line 47 of file ZigPadSettings.h.

5.50.3.3 - (UIColor *) modeColor [read, assign]

Returns the UIColor instance for the navigationBar.

Getter method for the modeColor property.

Definition at line 42 of file ZigPadSettings.h.

5.50.3.4 - (int) port [read, assign]

Setting for the HomeServer Port, depends on the simulation mode.

Getter method for the port property.

Definition at line 52 of file ZigPadSettings.h.

5.50.3.5 - (BOOL) simulationMode [read, write, assign]

Setting for the SimulationMode, YES if on.

Getter method for the simulationMode property.

Definition at line 32 of file ZigPadSettings.h.

5.50.3.6 - (BOOL) synchronizationMode [read, write, assign]

Setting for synchronization mode, YES if enabled.

Getter method for the simulationMode property.

Definition at line 62 of file ZigPadSettings.h.

5.50.3.7 - (BOOL) vibrationMode [read, write, assign]

Setting for the vibration mode, YES if enabled.

Getter method for the vibrationMode property.

Definition at line 57 of file ZigPadSettings.h.

The documentation for this class was generated from the following files:

- ZigPad/ZigPadSettings.h
- ZigPad/ZigPadSettings.m

Chapter 6

File Documentation

6.1 ZigPad/ActionViewController.h File Reference

```
#import <Foundation/Foundation.h>
#import <UIKit/UIKit.h>
#import <AudioToolbox/AudioServices.h>
#import "SequenceChoiceViewController.h"
#import "FavoritesViewController.h"
#import "Action.h"
#import "Presentation.h"
#import "Param.h"
#import "SyncEvent.h"
```

Data Structures

- class ActionViewController

6.2 ZigPad/ActionViewController.m File Reference

```
#import "ActionViewController.h"
#import "AnimatorHelper.h"
```

6.3 ZigPad/AnimatorHelper.h File Reference

```
#import <Foundation/Foundation.h>
```

```
#import <QuartzCore/QuartzCore.h>
```

Data Structures

- class AnimatorHelper

6.4 ZigPad/AnimatorHelper.m File Reference

```
#import "AnimatorHelper.h"
```

6.5 ZigPad/AsyncTCPSocket.h File Reference

```
#import <Foundation/Foundation.h>
```

Data Structures

- class NSObject(AsyncSocketDelegate)
- class AsyncTCPSocket

Typedefs

- typedef enum AsyncSocketError AsyncSocketError

Enumerations

- enum AsyncSocketError {

  AsyncSocketCFSocketError = kCFSocketError, AsyncSocketNoError = 0, AsyncSocketCanceledError, AsyncSocketConnectTimeoutError,

  AsyncSocketReadMaxedOutError, AsyncSocketReadTimeoutError, AsyncSocketWriteTimeoutError }

Variables

- NSString *const AsyncSocketException
- NSString *const AsyncSocketErrorDomain

6.5.1 Typedef Documentation

6.5.1.1 typedef enum AsyncSocketError AsyncSocketError

Definition at line 30 of file AsyncTCPSocket.h.

6.5.2 Enumeration Type Documentation

6.5.2.1 enum AsyncSocketError

Enumerator:

AsyncSocketCFSocketError

AsyncSocketNoError

AsyncSocketCanceledError

AsyncSocketConnectTimeoutError

AsyncSocketReadMaxedOutError

AsyncSocketReadTimeoutError

AsyncSocketWriteTimeoutError

Definition at line 20 of file AsyncTCPSocket.h.

6.5.3 Variable Documentation

6.5.3.1 NSString* const AsyncSocketErrorDomain

Definition at line 32 of file AsyncTCPSocket.m.

6.5.3.2 NSString* const AsyncSocketException

Definition at line 31 of file AsyncTCPSocket.m.

6.6 ZigPad/AsyncTCPSocket.m File Reference

```
#import "AsyncTCPSocket.h"
#import <sys/socket.h>
#import <netinet/in.h>
#import <arpa/inet.h>
#import <netdb.h>
```

Data Structures

- class **AsyncTCPSocket(Private)**
- class AsyncReadPacket
- class AsyncWritePacket
- class AsyncSpecialPacket

Defines

- #define DEFAULT_PREBUFFERING YES
- #define READQUEUE_CAPACITY 5
- #define WRITEQUEUE_CAPACITY 5
- #define READALL_CHUNKSIZE 256
- #define WRITE_CHUNKSIZE (1024 * 4)

Enumerations

- enum AsyncSocketFlags {

  kEnablePreBuffering = 1 << 0, kDidPassConnectMethod = 1 << 1, kDidCompleteOpenForRead = 1 << 2, kDidCompleteOpenForWrite = 1 << 3,

  kStartingTLS = 1 << 4, kForbidReadsWrites = 1 << 5, kDisconnectAfterReads = 1 << 6, kDisconnectAfterWrites = 1 << 7,

  kClosingWithError = 1 << 8 }

Variables

- NSString *const AsyncSocketException = @"AsyncSocketException"
- NSString *const AsyncSocketErrorDomain = @"AsyncSocketErrorDomain"

6.6.1 Define Documentation

6.6.1.1 #define DEFAULT_PREBUFFERING YES

Definition at line 24 of file AsyncTCPSocket.m.

6.6.1.2 #define READALL_CHUNKSIZE 256

Definition at line 28 of file AsyncTCPSocket.m.

6.6.1.3 #define READQUEUE_CAPACITY 5

Definition at line 26 of file AsyncTCPSocket.m.

6.6.1.4 #define WRITE_CHUNKSIZE (1024 * 4)

Definition at line 29 of file AsyncTCPSocket.m.

6.6.1.5 #define WRITEQUEUE_CAPACITY 5

Definition at line 27 of file AsyncTCPSocket.m.

6.6.2 Enumeration Type Documentation

6.6.2.1 enum AsyncSocketFlags

Enumerator:

kEnablePreBuffering

kDidPassConnectMethod

kDidCompleteOpenForRead

kDidCompleteOpenForWrite

kStartingTLS

kForbidReadsWrites

kDisconnectAfterReads

kDisconnectAfterWrites

kClosingWithError

Definition at line 38 of file AsyncTCPSocket.m.

6.6.3 Variable Documentation

6.6.3.1 NSString* const AsyncSocketErrorDomain = @"AsyncSocketErrorDomain"

Definition at line 32 of file AsyncTCPSocket.m.

6.6.3.2 NSString* const AsyncSocketException = @"AsyncSocketException"

Definition at line 31 of file AsyncTCPSocket.m.

6.7 ZigPad/AsyncUdpSocket.h File Reference

```
#import <Foundation/Foundation.h>
```

Data Structures

- class AsyncUdpSocket
- protocol <AsyncUdpSocketDelegate>

Typedefs

- typedef enum AsyncUdpSocketError AsyncUdpSocketError

Enumerations

- enum AsyncUdpSocketError {
  AsyncUdpSocketCFSocketError = kCFSocketError, AsyncUdpSocketNoError = 0, AsyncUdpSocketBadParameter, AsyncUdpSocketIPv4Unavailable,
  AsyncUdpSocketIPv6Unavailable, AsyncUdpSocketSendTimeoutError, AsyncUdpSocketReceiveTimeoutError }

Variables

- NSString *const AsyncUdpSocketException
- NSString *const AsyncUdpSocketErrorDomain

6.7.1 Typedef Documentation

6.7.1.1 typedef enum AsyncUdpSocketError AsyncUdpSocketError

Definition at line 29 of file AsyncUdpSocket.h.

6.7.2 Enumeration Type Documentation

6.7.2.1 enum AsyncUdpSocketError

Enumerator:

AsyncUdpSocketCFSocketError

AsyncUdpSocketNoError

AsyncUdpSocketBadParameter

AsyncUdpSocketIPv4Unavailable

AsyncUdpSocketIPv6Unavailable

AsyncUdpSocketSendTimeoutError

AsyncUdpSocketReceiveTimeoutError

Definition at line 19 of file AsyncUdpSocket.h.

6.7.3 Variable Documentation

6.7.3.1 NSString* const AsyncUdpSocketErrorDomain

Definition at line 31 of file AsyncUdpSocket.m.

6.7.3.2 NSString* const AsyncUdpSocketException

Definition at line 30 of file AsyncUdpSocket.m.

6.8 ZigPad/AsyncUdpSocket.m File Reference

```
#import "AsyncUdpSocket.h"
#import <sys/socket.h>
#import <netinet/in.h>
#import <arpa/inet.h>
#import <sys/ioctl.h>
#import <net/if.h>
#import <netdb.h>
```

Data Structures

- class **AsyncUdpSocket(Private)**
- class AsyncSendPacket
- class AsyncReceivePacket

Defines

- #define SENDQUEUE_CAPACITY 5
- #define RECEIVEQUEUE_CAPACITY 5
- #define DEFAULT_MAX_RECEIVE_BUFFER_SIZE 9216

Enumerations

- enum AsyncUdpSocketFlags {

  kDidBind = 1 << 0, kDidConnect = 1 << 1, kSock4CanAcceptBytes = 1 << 2, kSock6CanAcceptBytes = 1 << 3,

  kSock4HasBytesAvailable = 1 << 4, kSock6HasBytesAvailable = 1 << 5, kForbidSendReceive = 1 << 6, kCloseAfterSends = 1 << 7,

  kCloseAfterReceives = 1 << 8, kDidClose = 1 << 9, kDequeueSendScheduled = 1 << 10, kDequeueReceiveScheduled = 1 << 11,

  kFlipFlop = 1 << 12 }

Variables

- NSString *const AsyncUdpSocketException = @"AsyncUdpSocketException"
- NSString *const AsyncUdpSocketErrorDomain = @"AsyncUdpSocketErrorDomain"

6.8.1 Define Documentation

6.8.1.1 #define DEFAULT_MAX_RECEIVE_BUFFER_SIZE 9216

Definition at line 28 of file AsyncUdpSocket.m.

6.8.1.2 #define RECEIVEQUEUE_CAPACITY 5

Definition at line 26 of file AsyncUdpSocket.m.

6.8.1.3 #define SENDQUEUE_CAPACITY 5

Definition at line 25 of file AsyncUdpSocket.m.

6.8.2 Enumeration Type Documentation

6.8.2.1 enum AsyncUdpSocketFlags

Enumerator:

kDidBind

kDidConnect

kSock4CanAcceptBytes

kSock6CanAcceptBytes

kSock4HasBytesAvailable

kSock6HasBytesAvailable

kForbidSendReceive

kCloseAfterSends

kCloseAfterReceives

kDidClose

kDequeueSendScheduled

kDequeueReceiveScheduled

kFlipFlop

Definition at line 39 of file AsyncUdpSocket.m.

6.8.3 Variable Documentation

6.8.3.1 NSString* const AsyncUdpSocketErrorDomain = @"AsyncUdpSocketErrorDomain"

Definition at line 31 of file AsyncUdpSocket.m.

6.8.3.2 NSString* const AsyncUdpSocketException = @"AsyncUdpSocketException"

Definition at line 30 of file AsyncUdpSocket.m.

6.9 ZigPad/Commander.h File Reference

```
#import <Foundation/Foundation.h>
#import "ZigPadSettings.h"
#import "Action.h"
#import "Param.h"
```

Data Structures

- class Commander

Variables

- NSString *const COMMANDER_IMPL

6.9.1 Variable Documentation

6.9.1.1 NSString* const COMMANDER_IMPL

Defines which commander implementation is used.

Definition at line 16 of file Commander.m.

6.10 ZigPad/Commander.m File Reference

```
#import "Commander.h"
```

Variables

- NSString *const COMMANDER_IMPL = @"TCPCommander"

6.10.1 Variable Documentation

6.10.1.1 NSString* const COMMANDER_IMPL = @"TCPCommander"

Defines which commander implementation is used.

Definition at line 16 of file Commander.m.

6.11 ZigPad/CommandViewController.h File Reference

```
#import "ActionViewController.h"
```

Data Structures

- class CommandViewController

6.12 ZigPad/CommandViewController.m File Reference

```
#import "CommandViewController.h"
#import <QuartzCore/QuartzCore.h>
#import "Param.h"
#import "LocalPicture.h"
#import "Commander.h"
```

6.13 ZigPad/Config.h File Reference

```
#import <Foundation/Foundation.h>
```

Data Structures

- class Config

6.14 ZigPad/Config.m File Reference

```
#import "Config.h"
#import "Action.h"
#import "Param.h"
#import "LocalPicture.h"
#import "Sequence.h"
#import "Presentation.h"
#import "Database.h"
#import "ZigPadSettings.h"
#import "ImageDownloader.h"
#import <CoreData/CoreData.h>
```

Variables

- NSString ∗ keyCache = @"000"
- NSMutableDictionary ∗ managedObjectIDs
- NSManagedObjectContext ∗ context

6.14.1 Variable Documentation

6.14.1.1 NSManagedObjectContext∗ context

Reference to the persistence context for CoreData.

Definition at line 36 of file Config.m.

6.14.1.2 NSString∗ keyCache = @"000"

The xml id of the actual parent-Tag at runtime when parser iterates top-down through the xml file.

Definition at line 26 of file Config.m.

6.14.1.3 NSMutableDictionary∗ managedObjectIDs

Maps all xml-ID's to ID's of Coredata-Entities.

Definition at line 31 of file Config.m.

6.15 ZigPad/coredata/Action.h File Reference

```
#import <Foundation/Foundation.h>
#import <CoreData/CoreData.h>
#import "Param.h"
#import "Sequence.h"
#import "BWOrderedManagedObject.h"
```

Data Structures

- class Action

6.16 ZigPad/coredata/Action.m File Reference

```
#import "Action.h"
```

```
#import "Sequence.h"
```

6.17 ZigPad/coredata/BWOrderedManagedObject.h File Reference

Data Structures

- class BWOrderedManagedObject

6.18 ZigPad/coredata/BWOrderedManagedObject.m File Reference

```
#import "BWOrderedManagedObject.h"
```

Data Structures

- class BWOrderedValueProxy

Variables

- NSString * BWOrderedChangeContext = "BWOrderedChangeContext"

6.18.1 Variable Documentation

6.18.1.1 NSString* BWOrderedChangeContext = "BWOrderedChangeContext"

Definition at line 82 of file BWOrderedManagedObject.m.

6.19 ZigPad/coredata/Database.h File Reference

```
#import <Foundation/Foundation.h>
#import <CoreData/CoreData.h>
```

Data Structures

- class Database

6.20 ZigPad/coredata/Database.m File Reference

```
#import "Database.h"
```

6.21 ZigPad/coredata/LocalPicture.h File Reference

```
#import <Foundation/Foundation.h>
#import <CoreData/CoreData.h>
#import "BWOrderedManagedObject.h"
```

Data Structures

- class LocalPicture

6.22 ZigPad/coredata/LocalPicture.m File Reference

```
#import "LocalPicture.h"
#import "Param.h"
#import "Sequence.h"
```

6.23 ZigPad/coredata/Param.h File Reference

```
#import <Foundation/Foundation.h>
#import <CoreData/CoreData.h>
#import "Action.h"
#import "LocalPicture.h"
#import "BWOrderedManagedObject.h"
```

Data Structures

- class Param

6.24 ZigPad/coredata/Param.m File Reference

```
#import "Param.h"
#import "Action.h"
```

6.25 ZigPad/coredata/Presentation.h File Reference

```
#import <Foundation/Foundation.h>
```

```
#import <CoreData/CoreData.h>
#import "BWOrderedManagedObject.h"
#import "Action.h"
#import "Sequence.h"
```

Data Structures

- class Presentation

6.26 ZigPad/coredata/Presentation.m File Reference

```
#import "Presentation.h"
#import "Sequence.h"
```

Variables

- int activeSequencesIndex = 0
- int activeActionsIndex = 0
- bool isFirstCallOfGetNextMethod = true

6.26.1 Variable Documentation

6.26.1.1 int activeActionsIndex = 0

Definition at line 27 of file Presentation.m.

6.26.1.2 int activeSequencesIndex = 0

Definition at line 26 of file Presentation.m.

6.26.1.3 bool isFirstCallOfGetNextMethod = true

Definition at line 28 of file Presentation.m.

6.27 ZigPad/coredata/Sequence.h File Reference

```
#import <Foundation/Foundation.h>
#import <CoreData/CoreData.h>
#import "BWOrderedManagedObject.h"
```

```
#import "LocalPicture.h"
```

Data Structures

- class Sequence

6.28 ZigPad/coredata/Sequence.m File Reference

```
#import "Sequence.h"
```

6.29 ZigPad/DataCache.h File Reference

```
#import <Foundation/Foundation.h>
```

Data Structures

- class DataCache

6.30 ZigPad/DataCache.m File Reference

```
#import "DataCache.h"
```

6.31 ZigPad/FavoritesViewController.h File Reference

```
#import <UIKit/UIKit.h>
#import "Action.h"
#import "Param.h"
#import "LocalPicture.h"
#import "Database.h"
#import <CoreData/CoreData.h>
#import "QuartzCore/QuartzCore.h"
#import "Commander.h"
#import "AnimatorHelper.h"
```

Data Structures

- class FavoritesViewController

6.32 ZigPad/FavoritesViewController.m File Reference

```
#import "FavoritesViewController.h"
```

Variables

- int favoritesCount

6.32.1 Variable Documentation

6.32.1.1 int favoritesCount

Definition at line 18 of file FavoritesViewController.m.

6.33 ZigPad/FlowCoverView.h File Reference

```
#import <UIKit/UIKit.h>
#import <OpenGLES/EAGLDrawable.h>
#import <OpenGLES/EAGL.h>
#import <OpenGLES/ES1/gl.h>
#import <OpenGLES/ES1/glext.h>
#import "DataCache.h"
```

Data Structures

- class FlowCoverView
- protocol <FlowCoverViewDelegate>

6.34 ZigPad/FlowCoverView.m File Reference

```
#import "FlowCoverView.h"
#import <QuartzCore/QuartzCore.h>
```

Data Structures

- class FlowCoverRecord

Defines

- #define TEXTURESIZE 256
- #define MAXTILES 48
- #define VISTILES 6
- #define SPREADIMAGE 0.1
- #define FLANKSPREAD 0.4
- #define FRICTION 10.0
- #define MAXSPEED 10.0

Variables

- const GLfloat GVertices []
- const GLshort GTextures []

6.34.1 Define Documentation

6.34.1.1 #define FLANKSPREAD 0.4

Definition at line 58 of file FlowCoverView.m.

6.34.1.2 #define FRICTION 10.0

Definition at line 59 of file FlowCoverView.m.

6.34.1.3 #define MAXSPEED 10.0

Definition at line 60 of file FlowCoverView.m.

6.34.1.4 #define MAXTILES 48

Definition at line 50 of file FlowCoverView.m.

6.34.1.5 #define SPREADIMAGE 0.1

Definition at line 57 of file FlowCoverView.m.

6.34.1.6 #define TEXTURESIZE 256

Definition at line 49 of file FlowCoverView.m.

6.34.1.7 #define VISTILES 6

Definition at line 51 of file FlowCoverView.m.

6.34.2 Variable Documentation

6.34.2.1 const GLshort GTextures[]

Initial value:

```
 {
        0, 0,
        1, 0,
        0, 1,
        1, 1,
}
```

Definition at line 75 of file FlowCoverView.m.

6.34.2.2 const GLfloat GVertices[]

Initial value:

```
 {
        -1.0f, -1.0f, 0.0f,
         1.0f, -1.0f, 0.0f,
        -1.0f,  1.0f, 0.0f,
         1.0f,  1.0f, 0.0f,
}
```

Definition at line 68 of file FlowCoverView.m.

6.35 ZigPad/ImageDownloader.h File Reference

```
#import <Foundation/Foundation.h>
```

Data Structures

- class ImageDownloader

6.36 ZigPad/ImageDownloader.m File Reference

```
#import "ImageDownloader.h"
```

6.37 ZigPad/ImageViewController.h File Reference

```
#import <UIKit/UIKit.h>
#import "ActionViewController.h"
#import "ImageDownloader.h"
```

Data Structures

- class ImageViewController

6.38 ZigPad/ImageViewController.m File Reference

```
#import "ImageViewController.h"
```

6.39 ZigPad/Importer.h File Reference

```
#import <Foundation/Foundation.h>
```

Data Structures

- class Importer

6.40 ZigPad/Importer.m File Reference

```
#import "Importer.h"
#import "Config.h"
#import <CoreData/CoreData.h>
#import "Action.h"
#import "Param.h"
#import "LocalPicture.h"
#import "Sequence.h"
#import "Presentation.h"
#import "Database.h"
```

Variables

- Config * configTag

- bool importSuccess

6.40.1 Variable Documentation

6.40.1.1 Config* configTag

Definition at line 23 of file Importer.m.

6.40.1.2 bool importSuccess

Definition at line 24 of file Importer.m.

6.41 ZigPad/main.m File Reference

```
#import <UIKit/UIKit.h>
#import "Commander.h"
#import "Database.h"
#import "ZigPadSettings.h"
```

Functions

- int main (int argc, char *argv[])

6.41.1 Function Documentation

6.41.1.1 int main (int *argc*, char * *argv[]*)

Definition at line 14 of file main.m.

6.42 ZigPad/MBProgressHUD/Demo/main.m File Reference

```
#import <UIKit/UIKit.h>
```

Functions

- int main (int argc, char *argv[])

6.42.1 Function Documentation

6.42.1.1 int main (int *argc*, char * *argv[]*)

Definition at line 11 of file main.m.

6.43 ZigPad/MBProgressHUD/Demo/Classes/HudDemoAppDelegate.h File Reference

```
#import <UIKit/UIKit.h>
```

Data Structures

- class HudDemoAppDelegate

6.44 ZigPad/MBProgressHUD/Demo/Classes/HudDemoAppDelegate.m File Reference

```
#import "HudDemoAppDelegate.h"
#import "HudDemoViewController.h"
```

6.45 ZigPad/MBProgressHUD/Demo/Classes/HudDemoViewController.h File Reference

```
#import <UIKit/UIKit.h>
#import "MBProgressHUD.h"
```

Data Structures

- class HudDemoViewController

6.46 ZigPad/MBProgressHUD/Demo/Classes/HudDemoViewController.m File Reference

```
#import "HudDemoViewController.h"
#import <unistd.h>
```

6.47 ZigPad/MBProgressHUD/MBProgressHUD.h File Reference

```
#import <UIKit/UIKit.h>
```

Data Structures

- protocol <MBProgressHUDDelegate>
- class MBRoundProgressView
- class MBProgressHUD

Enumerations

- enum MBProgressHUDMode { MBProgressHUDModeIndeterminate, MBProgressHUDModeDeterminate, MBProgressHUDModeCustomView }
- enum MBProgressHUDAnimation { MBProgressHUDAnimationFade, MBProgressHUDAnimationZoom }

6.47.1 Enumeration Type Documentation

6.47.1.1 enum MBProgressHUDAnimation

Enumerator:

MBProgressHUDAnimationFade Opacity animation

MBProgressHUDAnimationZoom Opacity + scale animation

Definition at line 42 of file MBProgressHUD.h.

6.47.1.2 enum MBProgressHUDMode

Enumerator:

MBProgressHUDModeIndeterminate Progress is shown using an UIActivityIndicatorView. This is the default.

MBProgressHUDModeDeterminate Progress is shown using a MBRoundProgressView.

MBProgressHUDModeCustomView Shows a custom view

Definition at line 33 of file MBProgressHUD.h.

6.48 ZigPad/MBProgressHUD/MBProgressHUD.m File Reference

```
#import "MBProgressHUD.h"
```

Data Structures

- class **MBProgressHUD()**

Defines

- #define PADDING 4.0
- #define LABELFONTSIZE 16.0
- #define LABELDETAILSFONTSIZE 12.0
- #define PI 3.14159265358979323846
- #define RADIANS(degrees) ((degrees * M_PI) / 180.0)

6.48.1 Define Documentation

6.48.1.1 #define LABELDETAILSFONTSIZE 12.0

Definition at line 185 of file MBProgressHUD.m.

6.48.1.2 #define LABELFONTSIZE 16.0

Definition at line 184 of file MBProgressHUD.m.

6.48.1.3 #define PADDING 4.0

Definition at line 182 of file MBProgressHUD.m.

6.48.1.4 #define PI 3.14159265358979323846

Definition at line 187 of file MBProgressHUD.m.

6.48.1.5 #define RADIANS(*degrees*) ((degrees * M_PI) / 180.0)

Definition at line 589 of file MBProgressHUD.m.

6.49 ZigPad/RootViewController.h File Reference

```
#import <UIKit/UIKit.h>
#import "MBProgressHUD.h"
#import "Synchronizer.h"
#import "Presentation.h"
#import "SettingsViewController.h"
```

Data Structures

- class RootViewController

6.50 ZigPad/RootViewController.m File Reference

```
#import "RootViewController.h"
#import "ActionViewController.h"
#import "MBProgressHUD.h"
#import "Importer.h"
#import "Database.h"
#import "Synchronizer.h"
#import "ZigPadSettings.h"
#import "SyncEvent.h"
#import "AnimatorHelper.h"
```

6.51 ZigPad/SequenceChoiceViewController.h File Reference

```
#import <UIKit/UIKit.h>
#import "FlowCoverView.h"
#import "Presentation.h"
```

Data Structures

- class SequenceChoiceViewController

6.52 ZigPad/SequenceChoiceViewController.m File Reference

```
#import "SequenceChoiceViewController.h"
#import <CoreData/CoreData.h>
#import "Database.h"
#import "Presentation.h"
#import "Sequence.h"
#import "LocalPicture.h"
#import "ActionViewController.h"
#import "AnimatorHelper.h"
```

```
#import "SyncEvent.h"
```

6.53 ZigPad/SettingsViewController.h File Reference

```
#import <UIKit/UIKit.h>
#import "ZigPadSettings.h"
```

Data Structures

- class SettingsViewController

6.54 ZigPad/SettingsViewController.m File Reference

```
#import "SettingsViewController.h"
#import "Commander.h"
```

6.55 ZigPad/SyncEvent.h File Reference

```
#import <Foundation/Foundation.h>
#import "SynchronizerConnection.h"
```

Data Structures

- class SyncEvent

Enumerations

- enum SyncEventCommand {
  SELECT = 83, JUMP = 74, CONNECTED = 67, LOST_CONNECTION = 76,
  REQUEST = 82, ANSWER = 65, EXIT = 69 }
- enum SyncEventSwipeDirection {
  RIGHT, LEFT, RIGHT_ANIMATED, LEFT_ANIMATED,
  UP }

6.55.1 Enumeration Type Documentation

6.55.1.1 enum SyncEventCommand

SyncEventCommand enumeration, list of the existing commands including the corresponding value as ASCI.

Enumerator:

SELECT

JUMP

CONNECTED

LOST_CONNECTION

REQUEST

ANSWER

EXIT

Definition at line 16 of file SyncEvent.h.

6.55.1.2 enum SyncEventSwipeDirection

Enumeration used to indicate the direction of the view.

Enumerator:

RIGHT

LEFT

RIGHT_ANIMATED

LEFT_ANIMATED

UP

Definition at line 37 of file SyncEvent.h.

6.56 ZigPad/SyncEvent.m File Reference

```
#import "SyncEvent.h"
```

6.57 ZigPad/Synchronizer.h File Reference

```
#import "TCPServer.h"
```

Data Structures

- class Synchronizer

6.58 ZigPad/Synchronizer.m File Reference

```
#import "Synchronizer.h"
#import "SyncEvent.h"
#import "SynchronizerConnection.h"
```

Variables

- NSString ∗const SERVICE_IDENTIFIER = @"zigpad"
- NSString ∗const SERVICE_DOMAIN = @"local"

6.58.1 Variable Documentation

6.58.1.1 NSString∗ const SERVICE_DOMAIN = @"local"

Domain for which the service should be propagated and searched.

Definition at line 27 of file Synchronizer.m.

6.58.1.2 NSString∗ const SERVICE_IDENTIFIER = @"zigpad"

The Bonjour application protocol, which must: 1) be no longer than 14 characters 2) contain only lower-case letters, digits, and hyphens 3) begin and end with lower-case letter or digit It should also be descriptive and human-readable See the following for more information: http://developer.apple.com/networking/bonjour/faq.html

Definition at line 22 of file Synchronizer.m.

6.59 ZigPad/SynchronizerConnection.h File Reference

```
#import <Foundation/Foundation.h>
```

Data Structures

- class SynchronizerConnection

6.60 ZigPad/SynchronizerConnection.m File Reference

```
#import "SynchronizerConnection.h"
#import "SyncEvent.h"
```

6.61 ZigPad/TCPCommander.h File Reference

```
#import <Foundation/Foundation.h>
#import "Commander.h"
#import "AsyncTCPSocket.h"
```

Data Structures

- class TCPCommander

6.62 ZigPad/TCPCommander.m File Reference

```
#import "TCPCommander.h"
```

6.63 ZigPad/TCPServer.h File Reference

```
#import <Foundation/Foundation.h>
```

Data Structures

- protocol <TCPServerDelegate>
- class TCPServer

Enumerations

- enum TCPServerErrorCode { kTCPServerCouldNotBindToIPv4Address = 1, kTCPServer-CouldNotBindToIPv6Address = 2, kTCPServerNoSocketsAvailable = 3 }

Variables

- NSString *const TCPServerErrorDomain

6.63.1 Enumeration Type Documentation

6.63.1.1 enum TCPServerErrorCode

Enumerator:

kTCPServerCouldNotBindToIPv4Address

kTCPServerCouldNotBindToIPv6Address

kTCPServerNoSocketsAvailable

Definition at line 54 of file TCPServer.h.

6.63.2 Variable Documentation

6.63.2.1 NSString* const TCPServerErrorDomain

Definition at line 50 of file TCPServer.h.

6.64 ZigPad/TCPServer.m File Reference

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>
#include <CFNetwork/CFSocketStream.h>
#import "TCPServer.h"
```

Data Structures

- class **TCPServer()**

Variables

- NSString *const TCPServerErrorDomain = @"TCPServerErrorDomain"

6.64.1 Variable Documentation

6.64.1.1 NSString* const TCPServerErrorDomain = @"TCPServerErrorDomain"

Definition at line 55 of file TCPServer.m.

6.65 ZigPad/UDPCommander.h File Reference

```
#import "Commander.h"
#import "AsyncUdpSocket.h"
```

Data Structures

- class UDPCommander

6.66 ZigPad/UDPCommander.m File Reference

```
#import "UDPCommander.h"
```

6.67 ZigPad/VideoViewController.h File Reference

```
#import <UIKit/UIKit.h>
#import "ActionViewController.h"
#import <MediaPlayer/MediaPlayer.h>
```

Data Structures

- class VideoViewController

6.68 ZigPad/VideoViewController.m File Reference

```
#import "VideoViewController.h"
```

Variables

- bool movieIsPlaying = false

6.68.1 Variable Documentation

6.68.1.1 bool movieIsPlaying = false

Definition at line 14 of file VideoViewController.m.

6.69 ZigPad/WebCamViewController.h File Reference

```
#import <UIKit/UIKit.h>
#import "ActionViewController.h"
```

Data Structures

- class WebcamViewController

6.70 ZigPad/WebCamViewController.m File Reference

```
#import "WebcamViewController.h"
#import "Database.h"
```

6.71 ZigPad/ZigPadAppDelegate.h File Reference

```
#import <UIKit/UIKit.h>
```

Data Structures

- class ZigPadAppDelegate

6.72 ZigPad/ZigPadAppDelegate.m File Reference

```
#import "ZigPadAppDelegate.h"
#import "RootViewController.h"
#import "Database.h"
```

6.73 ZigPad/ZigPadSettings.h File Reference

```
#import <Foundation/Foundation.h>
```

Data Structures

- class ZigPadSettings

6.74 ZigPad/ZigPadSettings.m File Reference

```
#import "ZigPadSettings.h"
```

Variables

- NSString ∗const SIMULATOR = @"Simulator"
- NSString ∗const CONFIGURATOR = @"KonfiguratorURL"

6.74.1 Variable Documentation

6.74.1.1 NSString* const CONFIGURATOR = @"KonfiguratorURL"

The key for the configuration URL.

Definition at line 19 of file ZigPadSettings.m.

6.74.1.2 NSString* const SIMULATOR = @"Simulator"

The key for the simulation mode.

Definition at line 14 of file ZigPadSettings.m.

Index