Game Kit Framework Reference

Networking & Internet

2009-05-26

Contents

Introduction

Framework	/System/Library/Frameworks/GameKit.framework
Header file directories	/System/Library/Frameworks/GameKit.framework/Headers
Companion guide	Game Kit Programming Guide
Declared in	GKPeerPickerController.h GKPublicConstants.h GKPublicProtocols.h GKSession.h GKSessionError.h GKVoiceChatService.h

Game Kit offers your applications the ability to create Bluetooth connections between two devices. It also offers the ability to host voice chat services over any network.

Classes

GKPeerPickerController Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/GameKit.framework
Availability	Available in iOS 3.0 and later.
Declared in	GKPeerPickerController.h
Companion guide	Game Kit Programming Guide
Related sample code	GKTank

Overview

The `GKPeerPickerController` class provides a standard user interface to allow an iPhone to discover and connect to another iPhone. The result is a configured `GKSession` object connecting the two devices. To use a `GKPeerPickerController` object, your application creates the controller, adds a delegate, configures the allowed connection types, and then shows the peer picker. The delegate is called as the user makes selections within the peer picker interface.

In iOS 3.0, the peer picker can be configured to select between Bluetooth and Internet connections.

Important: Although users can select internet connections in the peer picker, the `GKPeerPickerController` does not provide an user interface to configure them. If your application configures the peer picker to allow Internet connections, your application must also dismiss the peer picker and present its own interface to configure an internet connection.

On iOS 3.0, your application should release the peer picker object after it dismisses the peer picker dialog. On iOS 3.1 or later, your application may release the peer picker after it is shown to the user. If you do this, the peer picker controller is automatically deallocated after the dialog is dismissed.

Tasks

Setting and Getting the Delegate

`delegate` (page 11) *property*
The delegate of the peer picker controller.

Displaying the Picker Dialog

– `show` (page 12)
Displays the peer picker dialog to the user.

– `dismiss` (page 11)
Hides the peer picker dialog.

`visible` (page 11) *property*
A Boolean value that indicates whether the picker dialog is visible. (read-only)

Configuring Connectivity Options

`connectionTypesMask` (page 10) *property*
A mask that determines the types of connections a dialog presents to the user.

Properties

For more about Objective-C properties, see "Properties" in *The Objective-C Programming Language*.

connectionTypesMask

A mask that determines the types of connections a dialog presents to the user.

```
@property(nonatomic, assign) GKPeerPickerConnectionType connectionTypesMask
```

Discussion
Your application configures the connection types it allows before showing the peer picker. If your application allows more than one connection type, the peer picker offers the user a choice of which type of connection to use. The default value for the mask is `GKPeerPickerConnectionTypeNearby` (page 12).

Important: In iOS 3.0, `GKPeerPickerConnectionTypeNearby` (page 12) is required to be one of the allowed connection types. An exception is thrown if your application does not include it.

Availability
Available in iOS 3.0 and later.

Declared In
`GKPeerPickerController.h`

delegate

The delegate of the peer picker controller.

```
@property(nonatomic, assign) id<GKPeerPickerControllerDelegate> delegate
```

Discussion
The delegate must adopt the `GKPeerPickerControllerDelegate` formal protocol.

Availability
Available in iOS 3.0 and later.

Related Sample Code
GKTank

Declared In
`GKPeerPickerController.h`

visible

A Boolean value that indicates whether the picker dialog is visible. (read-only)

```
@property(readonly, getter=isVisible) BOOL visible
```

Availability
Available in iOS 3.0 and later.

Declared In
`GKPeerPickerController.h`

Instance Methods

dismiss

Hides the peer picker dialog.

```
- (void)dismiss
```

Discussion
The controller's delegate is responsible for dismissing the peer picker when it is no longer needed.

On iOS 3.1 or later, the peer picker is retained when it is shown, and autoreleased when it is dismissed.

Availability
Available in iOS 3.0 and later.

Related Sample Code
GKTank

Declared In
`GKPeerPickerController.h`

show

Displays the peer picker dialog to the user.

```
- (void)show
```

Discussion
On iOS 3.1 or later, the peer picker is retained when it is shown, and autoreleased when it is dismissed.

Availability
Available in iOS 3.0 and later.

Related Sample Code
GKTank

Declared In
`GKPeerPickerController.h`

Constants

GKPeerPickerConnectionType

Network connections available to the peer picker dialog.

```
enum {
    GKPeerPickerConnectionTypeOnline = 1 << 0,
    GKPeerPickerConnectionTypeNearby = 1 << 1
};
typedef NSUInteger GKPeerPickerConnectionType;
```

Constants

`GKPeerPickerConnectionTypeOnline`

An Internet-based connection.

Available in iOS 3.0 and later.

Declared in `GKPeerPickerController.h`.

`GKPeerPickerConnectionTypeNearby`

A Bluetooth connection to a device.

Available in iOS 3.0 and later.

Declared in `GKPeerPickerController.h`.

GKSession Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/GameKit.framework
Availability	Available in iOS 3.0 and later.
Declared in	GKSession.h
Companion guide	Game Kit Programming Guide
Related sample code	GKRocket GKTank

Overview

A `GKSession` object provides the ability to discover and connect to nearby iPhones using Bluetooth.

Sessions primarily work with **peers**. A peer is any iPhone made visible by creating and configuring a `GKSession` object. Each peer is identified by a unique identifier, called a peer id (`peerID`) string. Your application can use a `peerID` string to obtain a user-readable name for a remote peer and to attempt to connect to that peer. Similarly, your session's peer ID is visible to other nearby peers. Once a connection is established, your application uses the remote peer's ID to address data packets that it wishes to send.

Peers discover other peers by using a unique string to identify the service they implement, called a session ID (`sessionID`). Sessions can be configured to either broadcast a session ID (as a **server**), to search for other peers advertising with that session ID (as a **client**) or to act as both a server and a client simultaneously (as a **peer**.

Your application controls the behavior of a session through a delegate that implements the `GKSessionDelegate` protocol. The delegate is called when remote peers are discovered, when those peers attempt to connect to the session, and when the state of a remote peer changes. Your application also provides a data handler to the session so that the session can forward data it receives from remote peers. The data handler can be a separate object or the same object as the delegate.

GKSession methods are thread-safe and may be called from any thread. However, the session always calls its delegate on the main thread.

Tasks

Creating a Session

– `initWithSessionID:displayName:sessionMode:` (page 22)
Initializes and returns a newly allocated session.

Setting and Getting the Delegate

`delegate` (page 16) *property*
The delegate of the session object.

Searching for Other Peers

`available` (page 15) *property*
A Boolean value that determines whether or not the session wants to connect to new peers.

Obtaining Information About Other Peers

– `peersWithConnectionState:` (page 22)
Returns a list of peers in the specified connection state.

– `displayNameForPeer:` (page 21)
Returns a user-readable name for a peer.

Connecting to a Remote Peer

– `connectToPeer:withTimeout:` (page 19)
Attempts to create a connection to another iPhone.

– `cancelConnectToPeer:` (page 19)
Cancels a pending request to connect to another iPhone.

Receiving Connections from a Remote Peer

– `acceptConnectionFromPeer:error:` (page 18)
Called by the delegate to accept a connection request received from a remote peer.

– `denyConnectionFromPeer:` (page 20)
Called by the delegate to reject a connection request received from a remote peer.

Working With Connected Peers

– `setDataReceiveHandler:withContext:` (page 24)
    Sets the object that handles data received from other peers connected to the session.

– `sendData:toPeers:withDataMode:error:` (page 23)
    Transmits a collection of bytes to a list of connected peers.

– `sendDataToAllPeers:withDataMode:error:` (page 23)
    Transmits a collection of bytes to all connected peers.

`disconnectTimeout` (page 16) *property*
    A time interval that expresses how long the session waits before it disconnects a non responsive peer.

– `disconnectFromAllPeers` (page 20)
    Disconnects the session from all connected peers.

– `disconnectPeerFromAllPeers:` (page 21)
    Disconnects a connected peer from all peers connected to the session.

Information About the Session

`displayName` (page 17) *property*
    The name of the user. (read-only)

`peerID` (page 17) *property*
    A string that identifies your session to other peers. (read-only)

`sessionID` (page 17) *property*
    A string used to filter the list of peers who are allowed to see your session. (read-only)

`sessionMode` (page 18) *property*
    The mode the session uses to find other peers. (read-only)

Properties

For more about Objective-C properties, see “Properties” in *The Objective-C Programming Language*.

available

A Boolean value that determines whether or not the session wants to connect to new peers.

```
@property(getter=isAvailable) BOOL available
```

Discussion
When `available` is `YES`, the session is visible to other peers based on its `sessionMode` (page 18) property. When `available` is set to `NO`, it remains connected to peers, but is no longer visible to non connected peers. The default is `NO`.

Typically, your application configures the session object with a delegate and data receiver, and then sets `available` (page 15) to `YES`. When the delegate finishes connecting to peers, it should set the session’s `available` (page 15) property to `NO`.

Availability
Available in iOS 3.0 and later.

Related Sample Code
GKRocket
GKTank

Declared In
`GKSession.h`

delegate

The delegate of the session object.

```
@property(assign) id<GKSessionDelegate> delegate
```

Discussion
A session's delegate is responsible for observing changes to other peers running with the same session ID. Your application must set a delegate before making your session known to other peers.

Availability
Available in iOS 3.0 and later.

See Also
`GKSessionDelegate`

Related Sample Code
GKRocket
GKTank

Declared In
`GKSession.h`

disconnectTimeout

A time interval that expresses how long the session waits before it disconnects a non responsive peer.

```
@property(assign) NSTimeInterval disconnectTimeout
```

Discussion
The timeout is the waiting period before disconnecting a peer from the session. If a peer is disconnected, the delegate's `session:peer:didChangeState:` (page 51) method is called.

Availability
Available in iOS 3.0 and later.

Declared In
`GKSession.h`

displayName

The name of the user. (read-only)

```
@property(readonly) NSString *displayName
```

Discussion
The display name is transmitted to visible peers so that they can present a human-readable name for your session.

Availability
Available in iOS 3.0 and later.

See Also
`– displayNameForPeer:` (page 21)

Declared In
`GKSession.h`

peerID

A string that identifies your session to other peers. (read-only)

```
@property(readonly) NSString *peerID
```

Availability
Available in iOS 3.0 and later.

Related Sample Code
GKRocket

Declared In
`GKSession.h`

sessionID

A string used to filter the list of peers who are allowed to see your session. (read-only)

```
@property(readonly) NSString *sessionID
```

Discussion
The session ID is used by sessions configured as servers to advertise itself to other peers and by sessions configured as clients to search for compatible servers. The session ID should be the short name of an approved Bonjour service type.

Availability
Available in iOS 3.0 and later.

Declared In
`GKSession.h`

sessionMode

The mode the session uses to find other peers. (read-only)

```
@property(readonly) GKSessionMode sessionMode
```

Discussion
The session mode changes the behavior of the session when `available` (page 15) is set to `YES`.

- A `GKSessionModeServer` (page 25) session advertises itself to local devices using its session ID.
- A `GKSessionModeClient` (page 25) session searches for local devices advertising the same session ID. As it discovers available and compatible peers, it calls the delegate's `session:peer:didChangeState:` (page 51) method.
- A `GKSessionModePeer` (page 26) session both advertises as a server and searches as a client.

Availability
Available in iOS 3.0 and later.

See Also
`@property available` (page 15)

Declared In
`GKSession.h`

Instance Methods

acceptConnectionFromPeer:error:

Called by the delegate to accept a connection request received from a remote peer.

```
- (BOOL)acceptConnectionFromPeer:(NSString *)peerID error:(NSError **)error
```

Parameters

peerID
The string identifying the peer that initiated the connection to the session.

error
If an error occurred when connecting the peer, upon return contains an `NSError` object that explains the problem.

Return Value
`YES` if a connection was established to the remote peer; `NO` if an error occurred.

Discussion
When your session acts as a server, client peers can discover your session and attempt to connect to it. When a client attempts to connect to the session, the delegate's `session:didReceiveConnectionRequestFromPeer:` (page 51) method is called to decide whether the peer should be connected. Your application calls this method to accept the request, or `denyConnectionFromPeer:` (page 20) to reject it.

Availability
Available in iOS 3.0 and later.

See Also
`- denyConnectionFromPeer:` (page 20)

Declared In
`GKSession.h`

cancelConnectToPeer:

Cancels a pending request to connect to another iPhone.

```
- (void)cancelConnectToPeer:(NSString *)peerID
```

Parameters

peerID
The string identifying the peer you previously requested to connect to.

Discussion
Your application previously called `connectToPeer:withTimeout:` (page 19) to create a connection to another iPhone. When your application cancels the connection attempt, both delegates' `session:connectionWithPeerFailed:withError:` (page 50) methods are called.

If your application already connected to the peer, your application should call `disconnectFromAllPeers` (page 20) instead.

Availability
Available in iOS 3.0 and later.

See Also
`- connectToPeer:withTimeout:` (page 19)

Related Sample Code
GKRocket

Declared In
`GKSession.h`

connectToPeer:withTimeout:

Attempts to create a connection to another iPhone.

```
- (void)connectToPeer:(NSString *)peerID withTimeout:(NSTimeInterval)timeout
```

Parameters

peerID
The string that identifies the peer to connect to.

timeout
The amount of time to wait before canceling the connection attempt.

Discussion
When your application is acting as a client, it calls this method to connect to an available peer it discovered. When your application calls this method, a request is transmitted to the remote peer, who chooses whether to accept or reject the connection request.

If the connection to the remote peer is successful, the delegate's `session:peer:didChangeState:` (page 51) method is called for each peer it successfully connected to. If the connection fails or your application cancels the connection attempt, the session calls the delegate's `session:connectionWithPeerFailed:withError:` (page 50) method.

Availability
Available in iOS 3.0 and later.

See Also
`- cancelConnectToPeer:` (page 19)

Related Sample Code
GKRocket

Declared In
`GKSession.h`

denyConnectionFromPeer:

Called by the delegate to reject a connection request received from a remote peer.

```
- (void)denyConnectionFromPeer:(NSString *)peerID
```

Parameters

peerID
The string identifying the peer that initiated the connection to the session.

Discussion
When your session acts as a server, client peers can discover your session and attempt to connect to it. When a client attempts to connect to the session, the delegate's `session:didReceiveConnectionRequestFromPeer:` (page 51) method is called to decide whether the peer should be connected. Your application calls this method to reject the request or `acceptConnectionFromPeer:error:` (page 18) to accept it.

Availability
Available in iOS 3.0 and later.

See Also
`- acceptConnectionFromPeer:error:` (page 18)

Related Sample Code
GKRocket

Declared In
`GKSession.h`

disconnectFromAllPeers

Disconnects the session from all connected peers.

```
- (void)disconnectFromAllPeers
```

Availability
Available in iOS 3.0 and later.

Related Sample Code
GKRocket
GKTank

Declared In
`GKSession.h`

disconnectPeerFromAllPeers:

Disconnects a connected peer from all peers connected to the session.

```
- (void)disconnectPeerFromAllPeers:(NSString *)peerID
```

Parameters

peerID
A string identifying the peer to disconnect.

Availability
Available in iOS 3.0 and later.

Declared In
`GKSession.h`

displayNameForPeer:

Returns a user-readable name for a peer.

```
- (NSString *)displayNameForPeer:(NSString *)peerID
```

Parameters

peerID
A string that uniquely identifies a peer.

Return Value
The name for the peer or `nil` if *peerID* is not associated with a visible peer.

Discussion
The display name is used to populate your user interface with the names of other peers visible to the session.

Availability
Available in iOS 3.0 and later.

See Also
`@property displayName` (page 17)

Related Sample Code
GKRocket
GKTank

Declared In
`GKSession.h`

initWithSessionID:displayName:sessionMode:

Initializes and returns a newly allocated session.

```
- (id)initWithSessionID:(NSString *)sessionID displayName:(NSString *)name
    sessionMode:(GKSessionMode)mode
```

Parameters

`sessionID`
: A unique string that identifies your application. Your *`sessionID`* should be the short name of an approved Bonjour service type. If `nil`, the session uses the application's bundle identifier to create a *`sessionID`* string.

`name`
: A string identifying the user to display to other peers. If `nil`, the session uses the device name.

`mode`
: The mode the session should run in. See "Session Modes" (page 25) for possible values.

Return Value
An initialized session object or `nil` if an error occurred.

Discussion
Only sessions running with the same *`sessionID`* are visible to your session.

Availability
Available in iOS 3.0 and later.

Related Sample Code
GKRocket

GKTank

Declared In
`GKSession.h`

peersWithConnectionState:

Returns a list of peers in the specified connection state.

```
- (NSArray *)peersWithConnectionState:(GKPeerConnectionState)state
```

Parameters

`state`
: The connection state to search for. See "Connection States" (page 26) for possible values.

Return Value
An array of `NSString` objects with a `peerID` string for each peer visible to the session that is currently in the specified connection state. If there are no peers in the specified connection state, this method returns `nil`.

Availability
Available in iOS 3.0 and later.

Declared In

`GKSession.h`

sendData:toPeers:withDataMode:error:

Transmits a collection of bytes to a list of connected peers.

```
- (BOOL)sendData:(NSData *)data toPeers:(NSArray *)peers
    withDataMode:(GKSendDataMode)mode error:(NSError **)error
```

Parameters

`data`

The bytes to send.

`peers`

An array of `NSString` objects identifying the peers that should receive the data.

`mode`

The mechanism used to send the data.

`error`

If the data could not be queued, on return, this holds an `NSError` object describing the error.

Return Value

`YES` if the data was successfully queued for transmission; `NO` if the session object was unable to queue the data.

Discussion

The session queues the data and transmits it in the order it was queued. Data transmitted unreliably may be received out of order by the other peers.

Availability

Available in iOS 3.0 and later.

See Also

`– sendDataToAllPeers:withDataMode:error:` (page 23)

Related Sample Code

GKTank

Declared In

`GKSession.h`

sendDataToAllPeers:withDataMode:error:

Transmits a collection of bytes to all connected peers.

```
- (BOOL)sendDataToAllPeers:(NSData *)data withDataMode:(GKSendDataMode)mode
    error:(NSError **)error
```

Parameters

`data`

The bytes to send.

`mode`
: The mechanism used to send the data.

`error`
: If the data could not be queued, on return, this holds an `NSError` object describing the error.

Return Value
`YES` if the data was successfully queued for transmission; `NO` if the session object was unable to queue the data.

Discussion
The session queues the data and transmits it when the network is free.

Availability
Available in iOS 3.0 and later.

See Also
`– sendData:toPeers:withDataMode:error:` (page 23)

Declared In
`GKSession.h`

setDataReceiveHandler:withContext:

Sets the object that handles data received from other peers connected to the session.

```
- (void)setDataReceiveHandler:(id)handlerwithContext:(void *)context
```

Parameters

`handler`
: The object you want the session to call when it receives data from other peers.

`context`
: Arbitrary data to be passed to each invocation of the handler.

Discussion
The handler must implement a method with the following signature:

```
- (void) receiveData:(NSData *)data fromPeer:(NSString *)peer inSession:
(GKSession *)session context:(void *)context;
```

where *data* contains the bytes received from a remote peer, *peer* is a string that identifies the peer, *session* is the session that received the data, and *context* is the same context that was passed into the original call to `setDataReceiveHandler:withContext:` (page 24).

Important: Data received from other peers should be treated as *untrusted* data. Be sure to validate the data you receive from the session and write your code carefully to avoid security vulnerabilities. See the *Secure Coding Guide* for more information.

Availability
Available in iOS 3.0 and later.

Related Sample Code
GKRocket
GKTank

Declared In
`GKSession.h`

Constants

Data Transmission Modes

The mechanism used to transmit data to other peers.

```
typedef enum {
    GKSendDataReliable,
    GKSendDataUnreliable,
} GKSendDataMode;
```

Constants

`GKSendDataReliable`
: Continues to send the data until it is successfully transmitted or the connection times out. Reliable transmissions are delivered in the order they were sent. Use this when you need to guarantee delivery.

  Available in iOS 3.0 and later.

  Declared in `GKPublicConstants.h`.

`GKSendDataUnreliable`
: Sends the data once and does not retry if an error occurred. Data transmitted unreliably can be received out of order by recipients. Use this for small packets of data that must arrive quickly to be of any use to the recipient.

  Available in iOS 3.0 and later.

  Declared in `GKPublicConstants.h`.

Session Modes

Modes that determine how a session interacts with other peers.

```
typedef enum {
    GKSessionModeServer,
    GKSessionModeClient,
    GKSessionModePeer,
} GKSessionMode;
```

Constants

`GKSessionModeServer`
: A server advertises itself to local devices using its `sessionID` (page 17) property.

  Available in iOS 3.0 and later.

  Declared in `GKPublicConstants.h`.

`GKSessionModeClient`
: A client searches for servers advertising the same `sessionID` (page 17) property.

  Available in iOS 3.0 and later.

  Declared in `GKPublicConstants.h`.

`GKSessionModePeer`
A peer advertises like a server and searches like a client.

Available in iOS 3.0 and later.

Declared in `GKPublicConstants.h`.

Connection States

The state of a peer known to the session. States are not mutually exclusive. For example, a peer can be both available for other peers to discover while it is attempting to connect to another peer.

```
typedef enum {
    GKPeerStateAvailable,
    GKPeerStateUnavailable,
    GKPeerStateConnected,
    GKPeerStateDisconnected,
    GKPeerStateConnecting
} GKPeerConnectionState;
```

Constants

`GKPeerStateAvailable`
A peer not connected to the session, but one that the session can connect to.

Available in iOS 3.0 and later.

Declared in `GKPublicConstants.h`.

`GKPeerStateUnavailable`
A peer that is no longer interested in receiving connections.

Available in iOS 3.0 and later.

Declared in `GKPublicConstants.h`.

`GKPeerStateConnected`
A peer connected to the session.

Available in iOS 3.0 and later.

Declared in `GKPublicConstants.h`.

`GKPeerStateDisconnected`
A peer that disconnected from the session.

Available in iOS 3.0 and later.

Declared in `GKPublicConstants.h`.

`GKPeerStateConnecting`
A peer attempting to connect to the session.

Available in iOS 3.0 and later.

Declared in `GKPublicConstants.h`.

The Session Error Domain

This constant defines the `GKSession` error domain.

```
NSString * const GKSessionErrorDomain;
```

Constants

`GKSessionErrorDomain`

Indicates an error occurred in `GKSession`.

Available in iOS 3.0 and later.

Declared in `GKSessionError.h`.

GKSession Errors

Error codes for the `GKSession` error domain.

```
typedef enum {
    GKSessionInvalidParameterError = 30500,
    GKSessionPeerNotFoundError = 30501,
    GKSessionDeclinedError = 30502,
    GKSessionTimedOutError = 30503,
    GKSessionCancelledError = 30504,
    GKSessionConnectionFailedError = 30505,
    GKSessionConnectionClosedError = 30506,
    GKSessionDataTooBigError = 30507,
    GKSessionNotConnectedError = 30508,
    GKSessionCannotEnableError = 30509,
    GKSessionInProgressError = 30510,
    GKSessionConnectivityError = 30201,
    GKSessionTransportError = 30202,
    GKSessionInternalError = 30203,
    GKSessionUnknownError = 30204,
    GKSessionSystemError = 30205
} GKSessionError;
```

Constants

`GKSessionInvalidParameterError`

A parameter had an unexpected value.

Available in iOS 3.0 and later.

Declared in `GKSessionError.h`.

`GKSessionPeerNotFoundError`

A peer with the specified `peerID` string could not be found.

Available in iOS 3.0 and later.

Declared in `GKSessionError.h`.

`GKSessionDeclinedError`

The peer your application tried to connect to refused the connection.

Available in iOS 3.0 and later.

Declared in `GKSessionError.h`.

`GKSessionTimedOutError`

The operation could not be completed in the specified timeout period.

Available in iOS 3.0 and later.

Declared in `GKSessionError.h`.

`GKSessionCancelledError`
A peer that invited the session to connect to them canceled the connection request.

Available in iOS 3.0 and later.

Declared in `GKSessionError.h`.

`GKSessionConnectionFailedError`
The attempt to establish a connection with another peer failed.

Available in iOS 3.0 and later.

Declared in `GKSessionError.h`.

`GKSessionConnectionClosedError`
The connection to another peer closed unexpectedly.

Available in iOS 3.0 and later.

Declared in `GKSessionError.h`.

`GKSessionDataTooBigError`
The data your application attempted to send was too large for the session to transmit in a single call.

Available in iOS 3.0 and later.

Declared in `GKSessionError.h`.

`GKSessionNotConnectedError`
Reserved for future use.

Available in iOS 3.0 and later.

Declared in `GKSessionError.h`.

`GKSessionCannotEnableError`
Bluetooth is not currently available.

Available in iOS 3.0 and later.

Declared in `GKSessionError.h`.

`GKSessionInProgressError`
The peer your application attempted to connect to has already requested a connection to your session.

Available in iOS 3.0 and later.

Declared in `GKSessionError.h`.

`GKSessionConnectivityError`
An error occurred in the `GKSession` object's connection code.

Available in iOS 3.0 and later.

Declared in `GKSessionError.h`.

`GKSessionTransportError`
An error occurred in the `GKSession` object's transport code.

Available in iOS 3.0 and later.

Declared in `GKSessionError.h`.

`GKSessionInternalError`
An serious error occurred inside `GKSession`.

Available in iOS 3.0 and later.

Declared in `GKSessionError.h`.

`GKSessionSystemError`
: An error occurred outside of the `GKSession` object's control, such as memory allocation.

  Available in iOS 3.0 and later.

  Declared in `GKSessionError.h`.

`GKSessionUnknownError`
: Reserved for when the error does not fit in another category above.

  Available in iOS 3.0 and later.

  Declared in `GKSessionError.h`.

GKVoiceChatService Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/GameKit.framework
Availability	Available in iOS 3.0 and later.
Declared in	GKVoiceChatService.h
Companion guide	Game Kit Programming Guide
Related sample code	GKRocket

Overview

The `GKVoiceChatService` class allows your application to connect two iOS devices into a voice chat.

The voice chat service uses a `client` implemented by your application to find and connect to other participants. Each participant in the chat is identified by a unique **participant identifier** string. The client provides a participant identifier for the local user and translates other participant identifiers into connections to other users. The format and mechanism used to translate participant identifiers into network connections is defined by the client. See the *Game Kit Programming Guide* for a more complete discussion.

Your application can configure the voice chat service to control the volume level of both local and remote participants and to detect when someone is speaking.

To use the voice chat service, your application retrieves the default service and attaches a client to it, then either connects to another participant or waits for them to start a connection.

Tasks

Getting the Shared Voice Chat Service

`+ defaultVoiceChatService` (page 35)
Retrieves the singleton chat service.

Setting the Client

`client` (page 33) *property*
: An object that the voice chat service uses to communicate with remote participants.

Establishing a Voice Chat

– `startVoiceChatWithParticipantID:error:` (page 38)
: Sends a request to another participant to join the voice chat.

Adjusting Audio Properties

`microphoneMuted` (page 34) *property*
: A boolean value that determines whether the user's microphone is muted.

`remoteParticipantVolume` (page 35) *property*
: A float that scales the volume of all remote participants.

Monitoring the Audio Level

`inputMeteringEnabled` (page 33) *property*
: A Boolean value that indicates whether the microphone's sound level is being monitored.

`inputMeterLevel` (page 33) *property*
: The volume in decibels (db) being received by the microphone. (read-only)

`outputMeteringEnabled` (page 34) *property*
: A Boolean value that indicates whether the voice level of remote participants is monitored.

`outputMeterLevel` (page 35) *property*
: The volume in decibels (db) being received from all other participants. (read-only)

Ending a Voice Chat

– `stopVoiceChatWithParticipantID:` (page 38)
: Ends a previously established voice chat.

Methods Called by the Client

– `acceptCallID:error:` (page 36)
: Accepts a request from a remote user to establish a voice chat.

– `denyCallID:` (page 36)
: Rejects a request to establish a voice chat.

– `receivedData:fromParticipantID:` (page 37)
: Called by the client to deliver new data received from a remote participant.

`- receivedRealTimeData:fromParticipantID:` (page 37)
Called by the client to deliver voice data received from a remote participant..

Properties

For more about Objective-C properties, see “Properties” in *The Objective-C Programming Language*.

client

An object that the voice chat service uses to communicate with remote participants.

```
@property(assign) id<GKVoiceChatClient> client
```

Discussion
The client’s chief responsibility is to provide a network connection that the voice chat service can use to connect to another participant.

Availability
Available in iOS 3.0 and later.

Declared In
`GKVoiceChatService.h`

inputMeteringEnabled

A Boolean value that indicates whether the microphone’s sound level is being monitored.

```
@property(nonatomic, getter=isInputMeteringEnabled) BOOL inputMeteringEnabled
```

Discussion
If `YES`, your application can read the `inputMeterLevel` (page 33) property to monitor the sound level of the microphone. If `NO`, the value of the `inputMeterLevel` (page 33) property is undefined. Default is `NO`. When your application doesn’t need to monitor the microphone, it should set this property to `NO` to improve performance.

Availability
Available in iOS 3.0 and later.

See Also
`@property inputMeterLevel` (page 33)

Declared In
`GKVoiceChatService.h`

inputMeterLevel

The volume in decibels (db) being received by the microphone. (read-only)

```
@property(readonly) float inputMeterLevel
```

Discussion
The value of this property is undefined if `inputMeteringEnabled` (page 33) is set to `NO`.

Availability
Available in iOS 3.0 and later.

See Also
`@property inputMeteringEnabled` (page 33)

Related Sample Code
GKRocket

Declared In
`GKVoiceChatService.h`

microphoneMuted

A boolean value that determines whether the user's microphone is muted.

```
@property(nonatomic, getter=isMicrophoneMuted) BOOL microphoneMuted
```

Discussion
`YES` if the user's microphone is turned off, `NO` if the user's speech is being transmitted to remote participants. The default is `NO`.

Availability
Available in iOS 3.0 and later.

Declared In
`GKVoiceChatService.h`

outputMeteringEnabled

A Boolean value that indicates whether the voice level of remote participants is monitored.

```
@property(nonatomic, getter=isOutputMeteringEnabled) BOOL outputMeteringEnabled
```

Discussion
If `YES`, your application can read the `outputMeterLevel` (page 35) property to monitor sound level of remote participants. If `NO`, the value of the `outputMeterLevel` (page 35) property is undefined. Default is `NO`. When your application doesn't need to monitor remote participants, it should set this property to `NO` to improve performance.

Availability
Available in iOS 3.0 and later.

Declared In
`GKVoiceChatService.h`

outputMeterLevel

The volume in decibels (db) being received from all other participants. (read-only)

```
@property(readonly) float outputMeterLevel
```

Discussion
The value of this property is undefined if `outputMeteringEnabled` (page 34) is set to `NO`.

The volume level is the aggregate volume of all remote participants, modified by the `remoteParticipantVolume` (page 35) property.

Availability
Available in iOS 3.0 and later.

Related Sample Code
GKRocket

Declared In
`GKVoiceChatService.h`

remoteParticipantVolume

A float that scales the volume of all remote participants.

```
@property(nonatomic) float remoteParticipantVolume
```

Discussion
The value should be between `0.0` (muted) and `1.0` (full volume). The default is `1.0`.

Availability
Available in iOS 3.0 and later.

Declared In
`GKVoiceChatService.h`

Class Methods

defaultVoiceChatService

Retrieves the singleton chat service.

```
+ (GKVoiceChatService *)defaultVoiceChatService
```

Return Value
The chat service.

Availability
Available in iOS 3.0 and later.

Related Sample Code
GKRocket

Declared In
`GKVoiceChatService.h`

Instance Methods

acceptCallID:error:

Accepts a request from a remote user to establish a voice chat.

```
- (BOOL)acceptCallID:(NSInteger)callID error:(NSError **)error
```

Parameters

callID
An integer that identifies the connection request.

error
If an problem occurred, this holds the error.

Return Value
`YES` if the connection was established; otherwise `NO`.

Discussion
When a remote user requests a voice chat, the voice chat service calls the client's `voiceChatService:didReceiveInvitationFromParticipantID:callID:` (page 55) method. The client calls this method to accept the request or `denyCallID:` (page 36) to reject it.

Availability
Available in iOS 3.0 and later.

Declared In
`GKVoiceChatService.h`

denyCallID:

Rejects a request to establish a voice chat.

```
- (void)denyCallID:(NSInteger)callID
```

Parameters

callID
An integer that identifies the connection request.

Discussion
When a remote user requests a voice chat, the voice chat service calls the client's `voiceChatService:didReceiveInvitationFromParticipantID:callID:` (page 55) method. The client calls this method to reject the request or `acceptCallID:error:` (page 36) to accept it.

Availability
Available in iOS 3.0 and later.

Related Sample Code
GKRocket

Declared In
`GKVoiceChatService.h`

receivedData:fromParticipantID:

Called by the client to deliver new data received from a remote participant.

```
- (void)receivedData:(NSData *)arbitraryData fromParticipantID:(NSString
    *)participantID
```

Parameters

`arbitraryData`
The data received from a participant.

`participantID`
A string that uniquely identifies the participant that sent the data.

Discussion
The voice chat service uses a network connection provided by the client to exchange information between the participants. When the client receives information intended for the voice chat service, it should call this method to transfer it.

Availability
Available in iOS 3.0 and later.

See Also
`- voiceChatService:sendData:toParticipantID:` (page 56)

Related Sample Code
GKRocket

Declared In
`GKVoiceChatService.h`

receivedRealTimeData:fromParticipantID:

Called by the client to deliver voice data received from a remote participant..

```
- (void)receivedRealTimeData:(NSData *)audio fromParticipantID:(NSString
    *)participantID
```

Parameters

`audio`
The audio data that was received from the other participant.

`participantID`
A string that uniquely identifies the speaking participant.

Discussion
The voice chat service uses a network connection provided by the client to exchange information between the participants. When the client receives information intended for the voice chat service, it should call this method to transfer it.

Availability
Available in iOS 3.0 and later.

See Also
– `voiceChatService:sendRealTimeData:toParticipantID:` (page 57)

Declared In
`GKVoiceChatService.h`

startVoiceChatWithParticipantID:error:

Sends a request to another participant to join the voice chat.

```
- (BOOL)startVoiceChatWithParticipantID:(NSString *)participantID error:(NSError
    **)error
```

Parameters

participantID
A string that uniquely identifies the participant to connect to.

error
If a problem occurred establishing the voice chat, on return this holds the error.

Return Value
`YES` if the connection was successfully created.

Discussion
The voice chat service calls the client's `voiceChatService:sendData:toParticipantID:` (page 56) method to send the connection request to the remote participant.

Availability
Available in iOS 3.0 and later.

Declared In
`GKVoiceChatService.h`

stopVoiceChatWithParticipantID:

Ends a previously established voice chat.

```
- (void)stopVoiceChatWithParticipantID:(NSString *)participantID
```

Parameters

participantID
A string that uniquely identifies the participant in the chat.

Discussion
When this method is called, the client's `voiceChatService:didStopWithParticipantID:error:` (page 56) method is called.

Availability
Available in iOS 3.0 and later.

Related Sample Code
GKRocket

Declared In
`GKVoiceChatService.h`

Constants

Voice Chat Service Error Domain

This constant defines the `GKVoiceChatService` error domain.

```
NSString * const GKVoiceChatServiceErrorDomain;
```

Constants

`GKVoiceChatServiceErrorDomain`
Indicates an error occurred in `GKVoiceChatService`.

Available in iOS 3.0 and later.

Declared in `GKVoiceChatService.h`.

Voice Chat Service Errors

Error codes for the `GKVoiceChatService` error domain.

```
typedef enum {
    GKVoiceChatServiceInternalError = 32000,
    GKVoiceChatServiceNoRemotePacketsError = 32001,
    GKVoiceChatServiceUnableToConnectError = 32002,
    GKVoiceChatServiceRemoteParticipantHangupError = 32003,
    GKVoiceChatServiceInvalidCallIDError = 32004,
    GKVoiceChatServiceAudioUnavailableError = 32005,
    GKVoiceChatServiceUninitializedClientError = 32006,
    GKVoiceChatServiceClientMissingRequiredMethodsError = 32007,
    GKVoiceChatServiceRemoteParticipantBusyError = 32008,
    GKVoiceChatServiceRemoteParticipantCancelledError = 32009,
    GKVoiceChatServiceRemoteParticipantResponseInvalidError = 32010,
    GKVoiceChatServiceRemoteParticipantDeclinedInviteError = 32011,
    GKVoiceChatServiceMethodCurrentlyInvalidError = 32012,
    GKVoiceChatServiceNetworkConfigurationError = 32013,
    GKVoiceChatServiceUnsupportedRemoteVersionError = 32014,
    GKVoiceChatServiceOutOfMemoryError = 32015,
    GKVoiceChatServiceInvalidParameterError = 32016
} GKVoiceChatServiceError;
```

Constants

`GKVoiceChatServiceInternalError`
A serious error occurred inside the voice chat service.

Available in iOS 3.0 and later.

Declared in `GKPublicConstants.h`.

`GKVoiceChatServiceNoRemotePacketsError`
: The voice chat service stopped receiving packets from the remote participant.

  Available in iOS 3.0 and later.

  Declared in `GKPublicConstants.h`.

`GKVoiceChatServiceUnableToConnectError`
: The voice chat service was unable to establish a connection with another user.

  Available in iOS 3.0 and later.

  Declared in `GKPublicConstants.h`.

`GKVoiceChatServiceRemoteParticipantHangupError`
: The remote participant in a voice chat stopped the chat.

  Available in iOS 3.0 and later.

  Declared in `GKPublicConstants.h`.

`GKVoiceChatServiceInvalidCallIDError`
: The voice chat service didn't recognize the call identifier.

  Available in iOS 3.0 and later.

  Declared in `GKPublicConstants.h`.

`GKVoiceChatServiceAudioUnavailableError`
: The audio hardware is unavailable to the voice chat service.

  Available in iOS 3.0 and later.

  Declared in `GKPublicConstants.h`.

`GKVoiceChatServiceUninitializedClientError`
: The application did not set a client before calling voice chat service methods.

  Available in iOS 3.0 and later.

  Declared in `GKPublicConstants.h`.

`GKVoiceChatServiceClientMissingRequiredMethodsError`
: The voice chat service did not find an expected method defined by the client.

  Available in iOS 3.0 and later.

  Declared in `GKPublicConstants.h`.

`GKVoiceChatServiceRemoteParticipantBusyError`
: The remote participant is already connected to a voice chat.

  Available in iOS 3.0 and later.

  Declared in `GKPublicConstants.h`.

`GKVoiceChatServiceRemoteParticipantCancelledError`
: A remote participant attempted to start a voice chat, then canceled.

  Available in iOS 3.0 and later.

  Declared in `GKPublicConstants.h`.

`GKVoiceChatServiceRemoteParticipantResponseInvalidError`
: Invalid data was received from a remote participant.

  Available in iOS 3.0 and later.

  Declared in `GKPublicConstants.h`.

`GKVoiceChatServiceRemoteParticipantDeclinedInviteError`
: A remote participant declined an invitation.

  Available in iOS 3.0 and later.

  Declared in `GKPublicConstants.h`.

`GKVoiceChatServiceMethodCurrentlyInvalidError`
: A method on the voice chat service was called when it was not allowed to be called (for example, attempting to connect when the voice chat service was already connected).

  Available in iOS 3.0 and later.

  Declared in `GKPublicConstants.h`.

`GKVoiceChatServiceNetworkConfigurationError`
: The voice chat service had problems accessing the network.

  Available in iOS 3.0 and later.

  Declared in `GKPublicConstants.h`.

`GKVoiceChatServiceUnsupportedRemoteVersionError`
: The other participant is running a different version of the voice chat service.

  Available in iOS 3.0 and later.

  Declared in `GKPublicConstants.h`.

`GKVoiceChatServiceOutOfMemoryError`
: The voice chat service was unable to allocate memory required to operate.

  Available in iOS 3.0 and later.

  Declared in `GKPublicConstants.h`.

`GKVoiceChatServiceInvalidParameterError`
: A parameter had an unrecognized value.

  Available in iOS 3.0 and later.

  Declared in `GKPublicConstants.h`.

Protocols

GKPeerPickerControllerDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/GameKit.framework
Availability	Available in iOS 3.0 and later.
Declared in	GKPeerPickerController.h
Companion guide	Game Kit Programming Guide
Related sample code	GKTank

Overview

The GKPeerPickerControllerDelegate (page 45) protocol is implemented on an object to customize the behavior of a `GKPeerPickerController` object. The delegate is called by the peer picker to create a session object and to respond as the session is configured by the controller.

Tasks

Creating a Session for the Peer Picker

`– peerPickerController:didSelectConnectionType:` (page 46) *required method*
    Tells the delegate that the user selected a connection type. This method is optional. (required)

`– peerPickerController:sessionForConnectionType:` (page 47) *required method*
    Asks the delegate to return a session for the specified connection type. This method is optional. (required)

Responding to Connection Messages

`– peerPickerController:didConnectPeer:toSession:` (page 46) *required method*
    Tells the delegate that the controller connected a peer to the session. This method is optional but expected. (required)

Responding When the User Cancels the Connection Attempt

`– peerPickerControllerDidCancel:` (page 48) *required method*
Tells the delegate that the user cancelled the connection attempt. This method is optional but expected. (required)

Instance Methods

peerPickerController:didConnectPeer:toSession:

Tells the delegate that the controller connected a peer to the session. This method is optional but expected. (required)

```
- (void)peerPickerController:(GKPeerPickerController *)picker
    didConnectPeer:(NSString *)peerIDtoSession:(GKSession *)session
```

Parameters

picker
The controller that connected the peer.

peerID
The identification string for the peer that connected to the session.

session
The session that the peer is connected to.

Discussion
Once a peer is connected to the session, your application should take ownership of the session, dismiss the peer picker, and then use the session to communicate with the other peer.

Availability
Available in iOS 3.0 and later.

Declared In
`GKPeerPickerController.h`

peerPickerController:didSelectConnectionType:

Tells the delegate that the user selected a connection type. This method is optional. (required)

```
- (void)peerPickerController:(GKPeerPickerController *)picker
    didSelectConnectionType:(GKPeerPickerConnectionType)type
```

Parameters

picker
The controller for the peer picker dialog.

type
The type of network connection chosen by the user.

Discussion

If the peer picker is configured to allow the user to choose between multiple connection types, this method is called when the user selects the connection type they want to use. Your delegate implements this method if you want to override the behavior for a particular connection type.

Important: In iOS 3.0, the peer picker can configure Bluetooth connections (`GKPeerPickerConnectionTypeNearby` (page 12)). If the user chooses an Internet connection (`GKPeerPickerConnectionTypeOnline` (page 12)), your delegate should dismiss the dialog and present its own user interface to configure the Internet connection:

```
- (void)peerPickerController:(GKPeerPickerController *)picker
didSelectConnectionType:(GKPeerPickerConnectionType)type {
    if(type == GKPeerPickerConnectionTypeOnline) {
        [picker dismiss];
        [picker autorelease];
// Display your own user interface here.
}
```

Availability

Available in iOS 3.0 and later.

Declared In

`GKPeerPickerController.h`

peerPickerController:sessionForConnectionType:

Asks the delegate to return a session for the specified connection type. This method is optional. (required)

```
- (GKSession *)peerPickerController:(GKPeerPickerController *)picker
    sessionForConnectionType:(GKPeerPickerConnectionType)type
```

Parameters

`picker`

The controller requesting the session.

`type`

The type of connection the controller wants to configure.

Discussion

Your delegate is responsible for providing a `GKSession` to use to find and connect to other devices. When the peer picker needs a session, it calls this method. Your application can either create a new session or return a previously created session to the peer picker. The session that your application returns to the peer picker must advertise itself as a peer (`GKSessionModePeer` (page 26)).

If your delegate does not implement this method and the user selected a network of type `GKPeerPickerConnectionTypeNearby` (page 12), the peer controller allocates a new session that advertises itself as a peer (`GKSessionModePeer`) with the default `sessionID` and `displayName` parameters.

Special Considerations

In iOS 3.0, your delegate only receive requests for network of type `GKPeerPickerConnectionTypeNearby` (page 12).

Availability

Available in iOS 3.0 and later.

Declared In
`GKPeerPickerController.h`

peerPickerControllerDidCancel:

Tells the delegate that the user cancelled the connection attempt. This method is optional but expected. (required)

```
- (void)peerPickerControllerDidCancel:(GKPeerPickerController *)picker
```

Parameters

picker
    The controller for the peer picker dialog.

Discussion
After this method returns, the controller dismisses the picker interface.

Availability
Available in iOS 3.0 and later.

Declared In
`GKPeerPickerController.h`

GKSessionDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/GameKit.framework
Availability	Available in iOS 3.0 and later.
Declared in	GKSession.h
Companion guide	Game Kit Programming Guide
Related sample code	GKRocket GKTank

Overview

An object implements the `GKSessionDelegate` protocol to control the behavior of a `GKSession` object. The delegate is called when other visible peers change their state relative to the session, and to determine if another peer is allowed to connect to the session.

Tasks

Observing Changes to Peers

– `session:peer:didChangeState:` (page 51) *required method*
Received by the delegate when a peer changes state. (required)

Connection Requests From Other Peers

– `session:didReceiveConnectionRequestFromPeer:` (page 51) *required method*
Received by the delegate when a remote peer wants to create a connection to the session. (required)

Connection Errors

– `session:connectionWithPeerFailed:withError:` (page 50) *required method*
Received by the delegate when an attempt to connect to another peer failed. (required)

`– session:didFailWithError:` (page 50) *required method*
    Sent to the delegate when an serious error has occurred in the session. (required)

Instance Methods

session:connectionWithPeerFailed:withError:

Received by the delegate when an attempt to connect to another peer failed. (required)

```
- (void)session:(GKSession *)session connectionWithPeerFailed:(NSString *)peerID
    withError:(NSError *)error
```

Parameters

session
    The session that received the message.

peerID
    A string that uniquely identifies the peer.

error
    The error that occurred.

Discussion
The *error* parameter can be used to inform the user of why the connection failed.

Important: If a `GKPeerPickerController` object is being used to configure the session, the controller handles this message automatically. Your delegate can ignore it if the peer picker dialog is in use.

Availability
Available in iOS 3.0 and later.

Declared In
`GKPublicProtocols.h`

session:didFailWithError:

Sent to the delegate when an serious error has occurred in the session. (required)

```
- (void)session:(GKSession *)session didFailWithError:(NSError *)error
```

Parameters

session
    The session that failed.

error
    The error that occurred.

Discussion
This method is called when an serious internal error occurred in the session. Your application should disconnect the session from other peers and release the session.

Availability
Available in iOS 3.0 and later.

Declared In
`GKPublicProtocols.h`

session:didReceiveConnectionRequestFromPeer:

Received by the delegate when a remote peer wants to create a connection to the session. (required)

```
- (void)session:(GKSession *)session didReceiveConnectionRequestFromPeer:(NSString
    *)peerID
```

Parameters

`session`
The session that received the request.

`peerID`
A string that uniquely identifies the peer.

Discussion
The delegate should call the session's `acceptConnectionFromPeer:error:` (page 18) method if it wishes to accept the connection or the `denyConnectionFromPeer:` (page 20) method if it wishes to refuse the connection.

Important: If a `GKPeerPickerController` object is being used to configure the session, the controller handles this message automatically. Your delegate can ignore it if the peer picker dialog is in use. If your application is not using a `GKPeerPickerController` object to configure the session, then your delegate must implement this method as described above.

Availability
Available in iOS 3.0 and later.

Declared In
`GKPublicProtocols.h`

session:peer:didChangeState:

Received by the delegate when a peer changes state. (required)

```
- (void)session:(GKSession *)session peer:(NSString *)peerID
    didChangeState:(GKPeerConnectionState)state
```

Parameters

`session`
The session that received the update.

`peerID`
A string that identifies the peer.

`state`
The state the peer changed to.

Discussion

A session calls this method whenever a visible peer changes it state relative to itself. The action your delegate should take depends on what state the peer moved to.

- When a peer first becomes visible to the session, it appears with a state of `GKPeerStateAvailable`. Your application should show this peer in its user interface. If the peer changes its state to `GKPeerStateUnavailable`, then it no longer accepts connection requests and your application should remove it from the user interface.
- The delegate should ignore `GKPeerStateConnecting` changes and implement the `session:didReceiveConnectionRequestFromPeer:` (page 51) method instead.
- When a peer is connected (`GKPeerStateConnected`), your application may send data to the peer and receive data from the peer. If a connection to a peer is lost or if the peer deliberately disconnects (`GKPeerStateDisconnected`), your application should stop sending messages to this peer.

Important: If a `GKPeerPickerController` object is being used to configure the session, the controller handles updates for the `GKPeerStateAvailable`, `GKPeerStateUnavailable`, and `GKPeerStateConnected` states. Your delegate can ignore state changes if the peer picker dialog is in use.

Availability

Available in iOS 3.0 and later.

Declared In

`GKPublicProtocols.h`

GKVoiceChatClient Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/GameKit.framework
Availability	Available in iOS 3.0 and later.
Declared in	GKVoiceChatService.h
Companion guide	Game Kit Programming Guide

Overview

The `GKVoiceChatClient` protocol is implemented to control the behavior of the `GKVoiceChatService` object. The voice chat client has a number of responsibilities:

- Provides a network connection that the voice chat service uses to send and receive configuration data with other participants. If this network connection is shared with other application data, the client must also disambiguate between chat configuration data and application data.
- Provides a participant ID that identifies the user to remote participants in the chat.
- Defines how a remote user's participant ID translates into a network connection to that user.
- Accepts or rejects requests from remote participants to join the voice chat.

Tasks

Getting Information About the Participant

`– participantID` (page 54) *required method*
Returns a string that uniquely identifies the local user. (required)

Sending Data to Other Participants

`– voiceChatService:sendData:toParticipantID:` (page 56) *required method*
A request for the client to send data to a participant. (required)

– `voiceChatService:sendRealTimeData:toParticipantID:` (page 57) *required method*
  Asks the client to send data to a participant that must get there quickly. This method is optional. (required)

Accepting Invitations From Remote Participants

– `voiceChatService:didReceiveInvitationFromParticipantID:callID:` (page 55) *required method*
  Asks the client to accept or reject an invitation from a remote participant. This method is optional. (required)

Responding to Changes in Other Participants

– `voiceChatService:didStartWithParticipantID:` (page 56) *required method*
  Received by the client when a voice chat with another participant is established. This method is optional. (required)

– `voiceChatService:didNotStartWithParticipantID:error:` (page 55) *required method*
  Received by the client when an attempt to establish a voice chat with another participant failed. This method is optional. (required)

– `voiceChatService:didStopWithParticipantID:error:` (page 56) *required method*
  Received by the client when a previously established voice chat has ended. This method is optional. (required)

Instance Methods

participantID

Returns a string that uniquely identifies the local user. (required)

```
- (NSString *)participantID
```

Return Value
A string that can be used by other participants to connect to the local user.

Discussion
The client decides the format and meaning of the participant identifier. For more information, see the *Game Kit Programming Guide*.

Availability
Available in iOS 3.0 and later.

Declared In
`GKPublicProtocols.h`

voiceChatService:didNotStartWithParticipantID:error:

Received by the client when an attempt to establish a voice chat with another participant failed. This method is optional. (required)

```
- (void)voiceChatService:(GKVoiceChatService *)voiceChatService
    didNotStartWithParticipantID:(NSString *)participantID error:(NSError *)error
```

Parameters

voiceChatService

The voice chat service that was establishing the connection.

participantID

A string that uniquely identifies the other user.

error

The error that prevented the voice chat from being established.

Discussion

Your application can implement this method to notify the user that an error occurred when establishing a connection.

Availability

Available in iOS 3.0 and later.

Declared In

`GKPublicProtocols.h`

voiceChatService:didReceiveInvitationFromParticipantID:callID:

Asks the client to accept or reject an invitation from a remote participant. This method is optional. (required)

```
- (void)voiceChatService:(GKVoiceChatService *)voiceChatService
    didReceiveInvitationFromParticipantID:(NSString *)participantID
    callID:(NSInteger)callID
```

Parameters

voiceChatService

The service that received the request.

participantID

A string that uniquely identifies the other user.

callID

An integer that uniquely identifies the request.

Discussion

If this method is not implemented by the client, the voice chat service automatically accept requests from other participants.

This method should call the service's `acceptCallID:error:` (page 36) method if it wants to accept the request or the `denyCallID:` (page 36) to reject it.

Availability

Available in iOS 3.0 and later.

Declared In

`GKPublicProtocols.h`

voiceChatService:didStartWithParticipantID:

Received by the client when a voice chat with another participant is established. This method is optional. (required)

```
- (void)voiceChatService:(GKVoiceChatService *)voiceChatService
    didStartWithParticipantID:(NSString *)participantID
```

Parameters

voiceChatService

The voice chat service that initiated the connection.

participantID

A string that uniquely identifies the other user.

Discussion

Your client can use this method to update the user interface to show that a connection has been established.

Availability

Available in iOS 3.0 and later.

Declared In

`GKPublicProtocols.h`

voiceChatService:didStopWithParticipantID:error:

Received by the client when a previously established voice chat has ended. This method is optional. (required)

```
- (void)voiceChatService:(GKVoiceChatService *)voiceChatService
    didStopWithParticipantID:(NSString *)participantID error:(NSError *)error
```

Parameters

voiceChatService

The voice chat that maintained the connection.

participantID

A string that uniquely identifies the user who disconnected.

error

The error that caused the chat to end.

Discussion

Your application can implement this method to notify the user that an established voice connection has ended. This may occur when another participant ends the chat or if the network connection was lost.

Availability

Available in iOS 3.0 and later.

Declared In

`GKPublicProtocols.h`

voiceChatService:sendData:toParticipantID:

A request for the client to send data to a participant. (required)

```
- (void)voiceChatService:(GKVoiceChatService *)voiceChatService sendData:(NSData
    *)data toParticipantID:(NSString *)participantID
```

Parameters

voiceChatService

The service that requested the transmission.

data

The data to send.

participantID

A string that uniquely identifies the participant to send the data to.

Discussion

An implementation of this method must reliably transmit the data to the participant identified by *participantID*. When the client on the other end receives the data, it should forward it to the voice chat service by calling the service's `receivedData:fromParticipantID:` (page 37) method.

Availability

Available in iOS 3.0 and later.

Declared In

`GKPublicProtocols.h`

voiceChatService:sendRealTimeData:toParticipantID:

Asks the client to send data to a participant that must get there quickly. This method is optional. (required)

```
- (void)voiceChatService:(GKVoiceChatService *)voiceChatService
    sendRealTimeData:(NSData *)data toParticipantID:(NSString *)participantID
```

Parameters

voiceChatService

The service that requested the transmission.

data

The data to send.

participantID

A string that uniquely identifies the participant to send the data to.

Discussion

An implementation of this method maps the *participantID* string to a known participant and transmits the data to them. Data transmitted by this method can be sent unreliably. When the client on the other end receives this data, it should forward it to the voice chat service by calling the service's `receivedRealTimeData:fromParticipantID:` (page 37) method.

Availability

Available in iOS 3.0 and later.

Declared In

`GKPublicProtocols.h`

Document Revision History

This table describes the changes to *Game Kit Framework Reference*.

Date	Notes
2009-05-26	Minor edits.
2009-03-12	First version of a New document that describes the API for implementing Bluetooth networking and voice chat services.