MultipeerConnectivityFrameworkReference

Contents

Contents

About Multipeer Connectivity

Framework	/System/Library/Frameworks/MultipeerConnectivity.framework
Header file directories	/System/Library/Frameworks/MultipeerConnectivity.framework/Headers
Declared in	MCAdvertiserAssistant.h MCBrowserViewController.h MCError.h MCNearbyServiceAdvertiser.h MCNearbyServiceBrowser.h MCPeerID.h MCSession.h

The Multipeer Connectivity framework provides support for discovering services provided by nearby iOS devices using infrastructure Wi-Fi networks, peer-to-peer Wi-Fi, and Bluetooth personal area networks and subsequently communicating with those services by sending message-based data, streaming data, and resources (such as files).

Architecture

When working with the Multipeer Connectivity framework, your app must interact with several types of objects, as described below.

- Session objects (MCSession) provide support for communication between connected peer devices. If your app creates a session, it can invite other peers to join it. Otherwise, your app can join a session when invited by another peer.
- Advertiser objects (MCNearbyServiceAdvertiser) tell nearby peers that your app is willing to join sessions of a specified type.
- Advertiser assistant objects (MCAdvertiserAssistant) provide the same functionality as advertiser objects, but also provide a standard user interface that allows the user to accept invitations. If you wish to provide your own user interface, or if you wish to exercise additional programmatic control over which invitations are displayed, use an advertiser object directly.

- Browser objects (MCNearbyServiceBrowser) let your app search programmatically for nearby devices with apps that support sessions of a particular type.
- Browser view controller objects (MCBrowserViewController) provide a standard user interface that allows the user to choose nearby peers to add to a session.
- Peer IDs (MCPeerID) uniquely identify an app running on a device to nearby peers.

Session objects maintain a set of peer ID objects that represent the peers connected to the session. Advertiser objects also use a single local peer object to provide information that identifies the device and its user to other nearby devices.

Using the Framework

This framework is used in two phases: the discovery phase, and the session phase.

In the discovery phase, your app uses a browser object (described in *MCNearbyServiceBrowser Class Reference*) tobrowsefornearbypeers,optionallyusingtheprovidedviewcontroller(describedin*MCBrowserViewController Class Reference*) to display a user interface.

Theappalsousesanadvertiserobject(describedin*MCNearbyServiceAdvertiserClassReference*)oranadvertiser assistant object (described in *MCAdvertiserAssistant Class Reference*) to tell nearby peers that it is available so that apps on other nearby devices can invite it to a session.

During the discovery phase, your app has limited communication with and knowledge of other peers; it has access to the discoveryInfo data that other nearby clients provide, and any context data that other peers provide when inviting it to join a session.

After the user chooses which peers to add to a session, the app invites those peers to join the session. Apps running on the nearby devices can choose whether to accept or reject the invitation, and can ask their users for permission.

If the peer accepts the invitation, the browser establishes a connection with the advertiser and the session phase begins. In this phase, your app can perform direct communication to one or more peers within the session. The framework notifies your app through delegate callbacks when peers join the session and when they leave the session.

Classes

MCAdvertiserAssistant Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/MultipeerConnectivity.framework
Availability	Available in iOS 7.0 and later.
Declared in	MCAdvertiserAssistant.h

Overview

The MCAdvertiserAssistant is a convenience class that handles advertising, presents incoming invitations to the user and handles users' responses. This class should be used to provide a user interface for handling invitations when your app does not require programmatic control over the invitation process.

Before you can advertise a service, you must create an MCPeerID object that identifies your app and the user to nearby devices.

Tasks

Configuring and Initialization

delegate (page 8) *property*

The delegate object that handles advertising-assistant-related events.

discoveryInfo (page 8) *property*

The info dictionary that was passed when this object was initialized. (read-only)

serviceType (page 9) *property*

The service type that your app is advertising. (read-only)

Starting and Stopping the Assistant

– start (page 11)

Begins advertising the service provided by a local peer and starts the assistant.

– stop (page 11)

Stops advertising the service provided by a local peer and stops the assistant.

New Methods

– initWithServiceType:discoveryInfo:session: (page 9)

Initializes an advertiser assistant object.

session (page 9) *property*

The session into which new peers are added after accepting an invitation. (read-only)

Properties

delegate

The delegate object that handles advertising-assistant-related events.

@property(assign, nonatomic) id<MCAdvertiserAssistantDelegate> delegate

Availability

Available in iOS 7.0 and later.

Declared in

MCAdvertiserAssistant.h

discoveryInfo

The info dictionary that was passed when this object was initialized. (read-only)

@property(readonly, nonatomic) NSDictionary *discoveryInfo

Discussion

This property's value is set when you initialize the object, and cannot be changed later.

Availability

Available in iOS 7.0 and later.

Declared in

MCAdvertiserAssistant.h

serviceType

The service type that your app is advertising. (read-only)

@property(readonly, nonatomic) NSString *serviceType

Discussion

This property's value is set when you initialize the object, and cannot be changed later.

Availability

Available in iOS 7.0 and later.

Declared in

MCAdvertiserAssistant.h

session

The session into which new peers are added after accepting an invitation. (read-only)

@property(readonly, nonatomic) MCSession *session

Discussion

This property's value is set when you initialize the object, and cannot be changed later.

Availability

Available in iOS 7.0 and later.

Declared in

MCAdvertiserAssistant.h

Instance Methods

initWithServiceType:discoveryInfo:session:

Initializes an advertiser assistant object.

```
- (instancetype)initWithServiceType:(NSString *)serviceType
discoveryInfo:(NSDictionary *)info session:(MCSession *)session
```

Parameters

serviceType

The type of service to advertise. This should be a *short* text string that describes the app's networking protocol, in the same format as a Bonjour service type (without the transport protocol):

- Must be 1–15 characters long
- Can contain only ASCII lowercase letters, numbers, and hyphens.

This name should be easily distinguished from unrelated services. For example, a text chat app made by ABC company could use the service type `abc-txtchat`.

For more details, read "Domain Naming Conventions".

info

A dictionary of key-value pairs that are made available to browsers. Each key and value must be an `NSString` object.

This data is advertised using a Bonjour TXT record, encoded according to RFC 6763 (section 6). As a result:

- The key-value pair must be no longer than 255 bytes (total) when encoded in UTF-8 format with an equals sign (=) between the key and the value.
- Keys cannot contain an equals sign.

For optimal performance, the total size of the keys and values in this dictionary should be no more than about 400 bytes so that the entire advertisement can fit within a single Bluetooth data packet. For details on the maximum allowable length, read "Monitoring a Bonjour Service" in *NSNetServices and CFNetServices Programming Guide*.

If the data you need to provide is too large to fit within these constraints, you should create a custom discovery class using Bonjour for discovery and your choice of networking protocols for exchanging the information.

session

The session into which new peers should be added after they accept the invitation.

Return Value

Returns an initialized instance, or `nil` if an error occurred.

Discussion

This method throws an exception if a valid `peerID` object is not provided or if `serviceType` is not a legal Bonjour service type.

Availability

Available in iOS 7.0 and later.

Declared in

MCAdvertiserAssistant.h

start

Begins advertising the service provided by a local peer and starts the assistant.

- (void)start

Availability

Available in iOS 7.0 and later.

Declared in

MCAdvertiserAssistant.h

stop

Stops advertising the service provided by a local peer and stops the assistant.

- (void)stop

Availability

Available in iOS 7.0 and later.

Declared in

MCAdvertiserAssistant.h

MCBrowserViewController Class Reference

Inherits from	UIViewController : UIResponder : NSObject
Conforms to	MCNearbyServiceBrowserDelegate NSCoding (UIViewController) UIAppearanceContainer (UIViewController) NSObject (NSObject)
Framework	/System/Library/Frameworks/MultipeerConnectivity.framework
Availability	Available in iOS 7.0 and later.
Declared in	MCBrowserViewController.h

Overview

The MCBrowserViewController class presents nearby devices to the user and enables the user to invite nearby devices to a session. To use this class, call methods from the underlying UIViewController class (prepareForSegue:sender: and performSegueWithIdentifier:sender: for storyboards or presentViewController:animated:completion: and dismissViewControllerAnimated:completion: for nib-based views) to present and dismiss the view controller.

Tasks

Initializing a Browser View Controller

– initWithServiceType:session: (page 16)

Initializes a browser view controller using the provided service type and session.

– initWithBrowser:session: (page 15)

Initializes a browser view controller with the provided browser and session.

delegate (page 13) *property*

The delegate object that handles browser-view-controller-related events.

browser (page 13) *property*

The browser object that is used for discovering peers. (read-only)

session (page 15) *property*

The multipeer session to which the invited peers are connected. (read-only)

Getting and Setting the Maximum and Minimum Number of Peers

maximumNumberOfPeers (page 14) *property*

The maximum number of peers allowed in a session, including the local peer.

minimumNumberOfPeers (page 14) *property*

The minimum number of peers that need to be in a session, including the local peer.

Properties

browser

The browser object that is used for discovering peers. (read-only)

@property(readonly, nonatomic) MCNearbyServiceBrowser *browser

Discussion

This value is set when you initialize the object, and cannot be changed later.

Availability

Available in iOS 7.0 and later.

Declared in

MCBrowserViewController.h

delegate

The delegate object that handles browser-view-controller-related events.

@property(assign, nonatomic) id<MCBrowserViewControllerDelegate> delegate

Discussion

A browser view controller notifies the delegate:

- When the user presses the “Done” button, which is enabled when the specified minimum number of peers are connected in a session.
- When the user cancels the view controller.

Also, as new peers are discovered, the delegate can choose whether to present them in the user interface.

Availability

Available in iOS 7.0 and later.

Declared in

MCBrowserViewController.h

maximumNumberOfPeers

The maximum number of peers allowed in a session, including the local peer.

@property(assign, nonatomic) NSUInteger maximumNumberOfPeers

Discussion

The largest allowable value (and the default) is 8.

Availability

Available in iOS 7.0 and later.

Declared in

MCBrowserViewController.h

minimumNumberOfPeers

The minimum number of peers that need to be in a session, including the local peer.

@property(assign, nonatomic) NSUInteger minimumNumberOfPeers

Discussion

The smallest allowable value (and the default) is 2.

Availability

Available in iOS 7.0 and later.

Declared in

MCBrowserViewController.h

session

The multipeer session to which the invited peers are connected. (read-only)

@property(readonly, nonatomic) MCSession *session

Discussion

This value is set when you initialize the object, and cannot be changed later.

Availability

Available in iOS 7.0 and later.

Declared in

MCBrowserViewController.h

Instance Methods

initWithBrowser:session:

Initializes a browser view controller with the provided browser and session.

- (instancetype)initWithBrowser:(MCNearbyServiceBrowser *)browser session:(MCSession *)session

Parameters

browser

An object that the browser view controller uses for browsing. This is usually an instance of MCNearbyServiceBrowser. However, if your app is using a custom discovery scheme, you can instead pass any custom subclass that calls the methods defined in the MCNearbyServiceBrowserDelegate protocol on its delegate when peers are found and lost.

Important: If you want the browser view controller to manage the browsing process, the browser object must not be actively browsing, and its delegate must be nil.

session

The multipeer session into which the invited peers are connected.

Return Value

Returns an initialized object, or nil if an error occurred.

Discussion

This method throws an exception if the browser or session parameters do not contain valid objects.

Availability

Available in iOS 7.0 and later.

Declared in

MCBrowserViewController.h

initWithServiceType:session:

Initializes a browser view controller using the provided service type and session.

- (instancetype)initWithServiceType:(NSString *)serviceType session:(MCSession *)session

Parameters

serviceType

The type of service to browse for. This should be a short text string that describes the app's networking protocol, in the same format as a Bonjour service type:

- Must be 1–15 characters long
- Can contain only ASCII lowercase letters, numbers, and hyphens.

This name should be easily distinguished from unrelated services. For example, a text chat app made by ABC company could use the service type abc-txtchat.

For more details, read “Domain Naming Conventions”.

session

The multipeer session that any user-chosen peers should be invited to join.

Return Value

Returns an initialized object, or nil if an error occurred.

Discussion

This method throws an exception if the session or serviceType parameters do not contain valid objects or the specified Bonjour service type is not valid.

Availability

Available in iOS 7.0 and later.

Declared in

MCBrowserViewController.h

MCNearbyServiceAdvertiser Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/MultipeerConnectivity.framework

Availability	Available in iOS 7.0 and later.
Declared in	MCNearbyServiceAdvertiser.h

Overview

The MCNearbyServiceAdvertiser class publishes an advertisement for a specific service that your app provides through the Multipeer Connectivity framework and notifies its delegate about invitations from nearby peers.

Before you can advertise a service, you must create an MCPeerID object that identifies your app and the user to nearby devices.

The serviceType parameter is a short text string used to describe the app's networking protocol. It should be in the same format as a Bonjour service type: 1–15 characters long and valid characters include ASCII lowercase letters, numbers, and the hyphen. A short name that distinguishes itself from unrelated services is recommended; for example, a text chat app made by ABC company could use the service type "abc-txtchat". For more information about service types, read “Domain Naming Conventions”.

The discoveryInfo parameter is a dictionary of string key/value pairs that will be advertised for browsers to see. The content of discoveryInfo will be advertised within Bonjour TXT records, so you should keep the dictionary small for better discovery performance.

For more information about TXT records, read “Bonjour Operations”.

Tasks

Configuring and Initialization

myPeerID (page 20) *property*

The local peer ID for this instance. (read-only)

serviceType (page 20) *property*

The service type that your app is advertising (read-only)

Starting and Stopping Advertisement

– startAdvertisingPeer (page 22)

Begins advertising the service provided by a local peer.

– stopAdvertisingPeer (page 22)

Stops advertising the service provided by a local peer.

Properties

delegate

The delegate object that handles advertising-related events.

@property(assign, nonatomic) id<MCNearbyServiceAdvertiserDelegate> delegate

Availability

Available in iOS 7.0 and later.

Declared in

MCNearbyServiceAdvertiser.h

discoveryInfo

The info dictionary passed when this object was initialized. (read-only)

@property(readonly, nonatomic) NSDictionary *discoveryInfo

Discussion

This value is set when you initialize the object, and cannot be changed later.

Availability

Available in iOS 7.0 and later.

Declared in

MCNearbyServiceAdvertiser.h

myPeerID

The local peer ID for this instance. (read-only)

@property(readonly, nonatomic) MCPeerID *myPeerID

Discussion

This value is set when you initialize the object, and cannot be changed later.

Availability

Available in iOS 7.0 and later.

Declared in

MCNearbyServiceAdvertiser.h

serviceType

The service type that your app is advertising (read-only)

@property(readonly, nonatomic) NSString *serviceType

Discussion

This value is set when you initialize the object, and cannot be changed later.

Availability

Available in iOS 7.0 and later.

Declared in

MCNearbyServiceAdvertiser.h

Instance Methods

initWithPeer:discoveryInfo:serviceType:

Initializes an advertiser object.

- (instancetype)initWithPeer:(MCPeerID *)myPeerID discoveryInfo:(NSDictionary *)info
  serviceType:(NSString *)serviceType

Parameters

myPeerID

Your app's local peer ID.

info

A dictionary of key-value pairs that are made available to browsers. Each key and value must be an NSString object.

This data is advertised using a Bonjour TXT record, encoded according to RFC 6763 (section 6). As a result:

- The key-value pair must be no longer than 255 bytes (total) when encoded in UTF-8 format with an equals sign (=) between the key and the value.
- Keys cannot contain an equals sign.

For optimal performance, the total size of the keys and values in this dictionary should be no more than about 400 bytes so that the entire advertisement can fit within a single Bluetooth data packet. For details onthemaximumallowablelength,read“MonitoringaBonjourService”in*NSNetServicesandCFNetServices Programming Guide*.

If the data you need to provide is too large to fit within these constraints, you should create a custom discovery class using Bonjour for discovery and your choice of networking protocols for exchanging the information.

serviceType

The type of service to advertise. This should be a *short* text string that describes the app's networking protocol, in the same format as a Bonjour service type:

- Must be 1–15 characters long
- Can contain only ASCII lowercase letters, numbers, and hyphens.

This name should be easily distinguished from unrelated services. For example, a text chat app made by ABC company could use the service type abc-txtchat.

For more details, read “Domain Naming Conventions”.

Return Value

Returns an initialized instance, or nil if an error occurred.

Discussion

This method throws an exception if a valid peerID object is not provided or if the value of serviceType is not a legal Bonjour service type.

Availability

Available in iOS 7.0 and later.

Declared in

MCNearbyServiceAdvertiser.h

startAdvertisingPeer

Begins advertising the service provided by a local peer.

- (void)startAdvertisingPeer

Availability

Available in iOS 7.0 and later.

Declared in

MCNearbyServiceAdvertiser.h

stopAdvertisingPeer

Stops advertising the service provided by a local peer.

- (void)stopAdvertisingPeer

Availability

Available in iOS 7.0 and later.

Declared in

MCNearbyServiceAdvertiser.h

MCNearbyServiceBrowser Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/MultipeerConnectivity.framework
Availability	Available in iOS 7.0 and later.
Declared in	MCNearbyServiceBrowser.h

Overview

Searches (by service type) for services offered by nearby devices using infrastructure Wi-Fi, peer-to-peer Wi-Fi, and Bluetooth, and provides the ability to easily invite those devices to a Multipeer Connectivity session (MCSession).

Tasks

Initializing the Browser

– initWithPeer:serviceType: (page 25)

Initializes the nearby service browser object.

delegate (page 24) *property*

The delegate object that handles browser-related events.

myPeerID (page 24) *property*

The local peer ID for this instance. (read-only)

serviceType (page 25) *property*

The service type to browse for. (read-only)

Browsing for Peers

– startBrowsingForPeers (page 27)

Starts browsing for peers.

– stopBrowsingForPeers (page 27)

Stops browsing for peers.

Inviting Peers

– invitePeer:toSession:withContext:timeout: (page 26)

Invites a discovered peer to join a Multipeer Connectivity session.

Properties

delegate

The delegate object that handles browser-related events.

@property(assign, nonatomic) id<MCNearbyServiceBrowserDelegate> delegate

Availability

Available in iOS 7.0 and later.

Declared in

MCNearbyServiceBrowser.h

myPeerID

The local peer ID for this instance. (read-only)

@property(readonly, nonatomic) MCPeerID *myPeerID

Discussion

This value is set when you initialize the object, and cannot be changed later.

Availability

Available in iOS 7.0 and later.

Declared in

MCNearbyServiceBrowser.h

serviceType

The service type to browse for. (read-only)

@property(readonly, nonatomic) NSString *serviceType

Discussion

This value is set when you initialize the object, and cannot be changed later.

Availability

Available in iOS 7.0 and later.

Declared in

MCNearbyServiceBrowser.h

Instance Methods

initWithPeer:serviceType:

Initializes the nearby service browser object.

```
- (instancetype)initWithPeer:(MCPeerID *)myPeerID serviceType:(NSString *)serviceType
```

Parameters

myPeerID

The local peer ID for this instance.

serviceType

- Must be 1–15 characters long
- Can contain only ASCII lowercase letters, numbers, and hyphens.

This name should be easily distinguished from unrelated services. For example, a text chat app made by ABC company could use the service type `abc-txtchat`.

For more details, read "Domain Naming Conventions".

Return Value

Returns an initialized nearby service browser object, or `nil` if an error occurs.

Discussion

This method throws an exception if the `session` or `serviceType` parameters do not contain valid objects or the specified Bonjour service type is not valid.

Availability

Available in iOS 7.0 and later.

Declared in

MCNearbyServiceBrowser.h

invitePeer:toSession:withContext:timeout:

Invites a discovered peer to join a Multipeer Connectivity session.

```
- (void)invitePeer:(MCPeerID *)peer toSession:(MCSession *)session withContext:(NSData
  *)context timeout:(NSTimeInterval)timeout
```

Parameters

peer

The ID of the peer to invite.

session

The session you wish the invited peer to join.

context

An arbitrary piece of data that is passed to the nearby peer. This can be used to provide further information to the user about the nature of the invitation.

Important: The nearby peer should treat any data it receives as potentially untrusted. To learn more about working with untrusted data, read *Secure Coding Guide*.

timeout

The amount of time to wait for the peer to respond to the invitation.

This timeout is measured in seconds, and must be a positive value. If a negative value or zero is specified, the default timeout (30 seconds) is used.

Availability

Available in iOS 7.0 and later.

Declared in

MCNearbyServiceBrowser.h

startBrowsingForPeers

Starts browsing for peers.

- (void)startBrowsingForPeers

Discussion

After this method is called (until you call stopBrowsingForPeers (page 27)), the framework calls your delegate's browser:foundPeer:withDiscoveryInfo: (page 58) and browser:lostPeer: (page 59) methods as new peers are found and lost.

Availability

Available in iOS 7.0 and later.

Declared in

MCNearbyServiceBrowser.h

stopBrowsingForPeers

Stops browsing for peers.

- (void)stopBrowsingForPeers

Availability

Available in iOS 7.0 and later.

Declared in

MCNearbyServiceBrowser.h

MCPeerID Class Reference

Inherits from	NSObject
Conforms to	NSCopying NSSecureCoding NSObject (NSObject)
Framework	/System/Library/Frameworks/MultipeerConnectivity.framework
Availability	Available in iOS 7.0 and later.
Declared in	MCPeerID.h

Overview

The MCPeerID class represents a peer in a multipeer session.

The Multipeer Connectivity framework is responsible for creating peer objects that represent other devices. Your app is responsible for creating a single peer object that represents the instance of your app that is running on the local device.

To create a new peer ID for the local app and associate a display name with that ID, call initWithDisplayName: (page 29). The peer's name must be no longer than 63 bytes in UTF-8 encoding.

Tasks

Peer Methods

– initWithDisplayName: (page 29)

Initializes a peer.

displayName (page 29) *property*

The display name for this peer. (read-only)

Properties

displayName

The display name for this peer. (read-only)

@property(readonly, nonatomic) NSString *displayName

Discussion

For the local peer, this property is set when the object is initialized and cannot be changed.

For other peer objects provided to you by the framework, this property is provided by the peer and cannot be changed.

Availability

Available in iOS 7.0 and later.

Declared in

MCPeerID.h

Instance Methods

initWithDisplayName:

Initializes a peer.

- (instancetype)initWithDisplayName:(NSString *)myDisplayName

Parameters

myDisplayName

The display name for the local peer. If you use the multipeer browser view controller, this name is shown.

The display name is intended for use in UI elements, and should be short and descriptive of the local peer. The maximum allowable length is 63 bytes in UTF-8 encoding. The displayName parameter may not be nil or an empty string.

Return Value

Returns an initialized object.

Discussion

This method should be called *only* when creating the local peer, not for creating objects that represent other devices.

This method throws an exception if the displayName value is too long, empty, or nil.

Availability

Available in iOS 7.0 and later.

Declared in

MCPeerID.h

MCSession Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/MultipeerConnectivity.framework
Availability	Available in iOS 7.0 and later.
Declared in	MCSession.h MCError.h

Overview

An MCSession object enables and manages communication among all peers in a Multipeer Connectivity session.

Initiating a Session

To set up a session, your app must do the following:

1. Create an MCPeerID object that represents the local peer, and use it to initialize the session object.
2. Add peers to the session using a browser object, a browser view controller, or manually. (Sessions currently support up to 8 peers, including the local peer.)
3. Wait until the session calls your delegate object's session:peer:didChangeState: (page 64) method

with MCSessionStateConnected (page 44) as the new state, along with an object that tells you which peer became connected.

You should also set up an advertiser or advertiser assistant to allow other devices to ask your app to join a session that they create.

Relevant Method Group: "Creating a Session" (page 34)

Communicating With Peers

Once you have set up the session, your app can send data to other peers by calling one of the following methods:

- sendData:toPeers:withMode:error: (page 40) sends an NSData object to the specified peers.

  On each recipient device, the delegate object's session:didReceiveData:fromPeer: (page 62) method is called with the data object when the data has been fully received.

- sendResourceAtURL:withName:toPeer:withCompletionHandler: (page 41) sends the contents from an NSURL object to the specified peer. The URL can be either a local file URL or a web URL. The completionHandler block is called when the resource is fully received by the recipient peer or when an error occurs during transmission.

  This method returns an NSProgress object that you can use to cancel the transfer or check the current status of the transfer.

  On the recipient device, the session calls its delegate object's session:didStartReceivingResourceWithName:fromPeer:withProgress: (page 64) method when the device begins receiving the resource, and calls its session:didFinishReceivingResourceWithName:fromPeer:atURL:withError: (page 61)methodwhen the resource has been fully received or when an error occurs.

- startStreamWithName:toPeer:error: (page 42) creates a connected byte stream (NSOutputStream) that you can use to send data to the specified peer.

  On the recipient device, the session calls its delegate object's session:didReceiveStream:withName:fromPeer: (page 63) method with an NSInputStream object that represents the other endpoint of communication.

  On both sides, your code must set the stream's delegate, schedule the stream on a run loop, and open the stream. Your code must also implement stream delegate methods to manage sending and receiving stream data.

Important: Delegate calls occur on a private operation queue. If your app needs to perform an action on a particular run loop or operation queue, its delegate method should explicitly dispatch or schedule that work.

Relevant Method Group: "Sending Data and Resources" (page 35)

Managing Peers Manually

If you decide to write your own peer discovery code (with NSNetService or the Bonjour C API, for example), you can also manually connect nearby peers into a session. To do this, your app must do the following:

1. Establish a connection to nearby peers and exchange peer IDs with those peers. Each peer should serialize its own local MCPeerID object with NSKeyedArchiver, and the receiving peer should unserialize it with NSKeyedUnarchiver.

   Important: Do not attempt to construct a peer ID object for a nonlocal peer using initWithDisplayName: (page 29). A peer ID object must be constructed *on the device that it represents*.

2. Exchange connection data. After you have obtained the nearby peer's ID object, call nearbyConnectionDataForPeer:withCompletionHandler: (page 40) to obtain a connection data object specific to that nearby peer.

   When the completion handler block is called, send the resulting connection data object to that peer.

   Note: Each device in the session must perform this step for each nonlocal peer in the session. So if there are four devices in the session, each device must generate a connection data object for each of the other three devices.

3. When your app receives connection data from another peer, it must call connectPeer:withNearbyConnectionData: (page 38) to add that peer to the session.

   Note: Each of the nonlocal peers must also call connectPeer:withNearbyConnectionData: (page 38) with the connection data that it received from your app and other nonlocal peers.

You can also cancel an outstanding connection attempt by calling cancelConnectPeer: (page 37).

Tasks

Relevant Method Group: "Managing Peers Manually" (page 34)

Disconnecting

To leave a session, your app must call disconnect (page 38).

Relevant Method Group: "Leaving a Session" (page 35)

Tasks

Creating a Session

– initWithPeer: (page 38)

Creates a Multipeer Connectivity session.

– initWithPeer:securityIdentity:encryptionPreference: (page 39)

Creates a Multipeer Connectivity session, providing security information.

delegate (page 35) *property*

The delegate object that handles session-related events.

encryptionPreference (page 36) *property*

Indicates whether the connection prefers encrypted connections, unencrypted connections, or has no preference. (read-only)

myPeerID (page 36) *property*

A local identifier that represents the device on which your app is currently running. (read-only)

securityIdentity (page 36) *property*

The security identity of the local peer. (read-only)

Managing Peers Manually

– connectPeer:withNearbyConnectionData: (page 38)

Connect a peer to the session manually.

– cancelConnectPeer: (page 37)

Cancels an attempt to connect to a peer.

Properties

connectedPeers (page 35) *property*

An array of all peers that are currently connected to this session. (read-only)

– nearbyConnectionDataForPeer:withCompletionHandler: (page 40)

Obtains connection data for the specified peer.

Sending Data and Resources

– sendData:toPeers:withMode:error: (page 40)

Sends a message encapsulated in an NSData object to nearby peers.

– sendResourceAtURL:withName:toPeer:withCompletionHandler: (page 41)

Sends the contents of a URL to a peer.

– startStreamWithName:toPeer:error: (page 42)

Opens a byte stream to a nearby peer.

Leaving a Session

– disconnect (page 38)

Disconnects the local peer from the session.

Properties

connectedPeers

An array of all peers that are currently connected to this session. (read-only)

```
@property(readonly, nonatomic) NSArray *connectedPeers
```

Availability

Available in iOS 7.0 and later.

Declared in

MCSession.h

delegate

The delegate object that handles session-related events.

@property(assign, nonatomic) id<MCSessionDelegate> delegate

Availability

Available in iOS 7.0 and later.

Declared in

MCSession.h

encryptionPreference

Indicates whether the connection prefers encrypted connections, unencrypted connections, or has no preference. (read-only)

@property(readonly, nonatomic) MCEncryptionPreference encryptionPreference

Discussion

This value is set when you initialize the object, and cannot be changed later.

Availability

Available in iOS 7.0 and later.

Declared in

MCSession.h

myPeerID

A local identifier that represents the device on which your app is currently running. (read-only)

@property(readonly, nonatomic) MCPeerID *myPeerID

Discussion

This value is set when you initialize the object, and cannot be changed later.

Availability

Available in iOS 7.0 and later.

Declared in

MCSession.h

securityIdentity

The security identity of the local peer. (read-only)

@property(readonly, nonatomic) NSArray *securityIdentity

Discussion

This value is set when you initialize the object, and cannot be changed later. For details on this value, see the documentation for initWithPeer:securityIdentity:encryptionPreference: (page 39).

Availability

Available in iOS 7.0 and later.

Declared in

MCSession.h

Instance Methods

cancelConnectPeer:

Cancels an attempt to connect to a peer.

- (void)cancelConnectPeer:(MCPeerID *)peerID

Parameters

peerID

The ID of the nearby peer.

Discussion

This method is used for canceling connections to peers when you are using your own service discovery code. It should be called in two situations:

- If your app calls connectPeer:withNearbyConnectionData: (page 38) and later decides to cancel the connection attempt
- If your app has obtained nearby connection data for a peer and later decides not to connect to it

For more information, see "Managing Peers Manually" (page 33).

Availability

Available in iOS 7.0 and later.

Declared in

MCSession.h

connectPeer:withNearbyConnectionData:

Connect a peer to the session manually.

- (void)connectPeer:(MCPeerID *)peerID withNearbyConnectionData:(NSData *)data

Parameters

peerID

The peer ID object obtained from the nearby peer.

data

The connection data object obtained from the nearby peer.

Discussion

This method is used for connecting to peers when you are using your own service discovery code. For more information, see “Managing Peers Manually” (page 33).

Availability

Available in iOS 7.0 and later.

Declared in

MCSession.h

disconnect

Disconnects the local peer from the session.

- (void)disconnect

Availability

Available in iOS 7.0 and later.

Declared in

MCSession.h

initWithPeer:

Creates a Multipeer Connectivity session.

- (instancetype)initWithPeer:(MCPeerID *)myPeerID

Parameters

myPeerID

A local identifier that represents the device on which your app is currently running.

Return Value

Returns the initialized session object, or nil if an error occurs.

Discussion

This method is equivalent to calling initWithPeer:securityIdentity:encryptionPreference: (page 39) with a nil identity and MCEncryptionOptional as the encryption preference.

This method throws an exception if the provided peer ID object is invalid or nil.

Availability

Available in iOS 7.0 and later.

Declared in

MCSession.h

initWithPeer:securityIdentity:encryptionPreference:

Creates a Multipeer Connectivity session, providing security information.

```
- (instancetype)initWithPeer:(MCPeerID *)myPeerID securityIdentity:(NSArray *)identity
  encryptionPreference:(MCEncryptionPreference)encryptionPreference
```

Parameters

myPeerID

A local identifier that represents the device on which your app is currently running.

identity

An array containing information that can be used to identify the local peer to other nearby peers.

The first object in this array should be a SecIdentityRef object that provides the local peer's identity.

The remainder of the array should contain zero or more additional SecCertificateRef objects that provide any intermediate certificates that nearby peers might require when verifying the local peer's identity. These certificates should be sent in certificate chain order.

When you add other peers to the session, those peers receive your local peer's certificate (extracted from the provided identity) and any additional certificates that you provided. It is the receiving peer's responsibility to validate that certificate, if desired.

encryptionPreference

An integer value that indicates whether encryption is required, preferred, or undesirable.

Return Value

Returns the initialized session object, or nil if an error occurs.

Discussion

This method throws an exception if the provided peer ID object is invalid or nil.

Availability

Available in iOS 7.0 and later.

Declared in

MCSession.h

nearbyConnectionDataForPeer:withCompletionHandler:

Obtains connection data for the specified peer.

```
- (void)nearbyConnectionDataForPeer:(MCPeerID *)peerID withCompletionHandler:(void
(^)(NSData *connectionData, NSError *error))completionHandler
```

Parameters

peerID

A peer ID object obtained from the nearby peer that you want to add to a session.

completionHandler

A handler that is called when connection data is available or when an error occurs.

Discussion

This method provides connection data that is required when adding a specific nearby peer to a session if you are using your own service discovery code. For more information, see “Managing Peers Manually” (page 33).

Availability

Available in iOS 7.0 and later.

Declared in

MCSession.h

sendData:toPeers:withMode:error:

Sends a message encapsulated in an NSData object to nearby peers.

```
- (BOOL)sendData:(NSData *)data toPeers:(NSArray *)peerIDs
withMode:(MCSessionSendDataMode)mode error:(NSError **)error
```

Parameters

data

An object containing the message to send.

peerIDs

An array of peer ID objects representing the peers that should receive the message.

mode

The transmission mode to use (reliable or unreliable delivery).

error

The address of an NSError pointer where an error object should be stored upon error.

Return Value

Returns YES if the message was successfully enqueued for delivery, or NO if an error occurred.

Discussion

This method is asynchronous (non-blocking).

Ontherecipientdevice,thesessionobjectcallsitsdelegateobject'ssession:didReceiveData:fromPeer: (page 62) method with the message after it has been fully received.

Availability

Available in iOS 7.0 and later.

Declared in

MCSession.h

sendResourceAtURL:withName:toPeer:withCompletionHandler:

Sends the contents of a URL to a peer.

```
- (NSProgress *)sendResourceAtURL:(NSURL *)resourceURL withName:(NSString
*)resourceName toPeer:(MCPeerID *)peerID withCompletionHandler:(void (^)(NSError
*error))completionHandler
```

Parameters

resourceURL

A file or HTTP URL.

resourceName

A name for the resource.

peerID

The peer that should receive this resource.

completionHandler

A block that gets called when delivery succeeds or fails. Upon success, the handler is called with an error value of nil. Upon failure, the handle is called with an error object that indicates what went wrong.

Return Value

Returns an NSProgress object that can be used to query the status of the transfer or cancel the transfer.

Discussion

This method is asynchronous (non-blocking).

On the local device, the completion handler block is called when delivery succeeds or when an error occurs.

On the recipient device, the session calls its delegate's session:didStartReceivingResourceWithName:fromPeer:withProgress: (page 64) method as soon as it begins receiving the resource. This method provides an NSProgress object that your app can use to cancel the transfer or check its status.

Upon successful delivery, on the recipient device, the session calls its delegate's session:didFinishReceivingResourceWithName:fromPeer:atURL:withError: (page 61) method. The received resource is written to a file in a temporary location with the same base name; the app is responsible for opening the file or moving it to a permanent location before that delegate method returns.

Availability

Available in iOS 7.0 and later.

Declared in

MCSession.h

startStreamWithName:toPeer:error:

Opens a byte stream to a nearby peer.

```
- (NSOutputStream *)startStreamWithName:(NSString *)streamName toPeer:(MCPeerID
*)peerID error:(NSError **)error
```

Parameters

streamName

A name for the stream. This name is provided to the nearby peer.

peerID

The ID of the nearby peer.

error

The address of an NSError pointer where an error object should be stored if something goes wrong.

Return Value

Returns an output stream object upon success or nil if a stream could not be established.

Discussion

This method is non-blocking.

For more information about performing networking with input and output streams, read *Networking Programming Topics*.

Availability

Available in iOS 7.0 and later.

Declared in

MCSession.h

Constants

MCSessionSendDataMode

Indicates whether delivery of data should be guaranteed.

```
typedef NS_ENUM(NSInteger,
    MCSessionSendDataMode) {
    MCSessionSendDataReliable,
    MCSessionSendDataUnreliable
};
```

Constants

MCSessionSendDataReliable

The framework should guarantee delivery of each message, enqueueing and retransmitting data as needed, and ensuring in-order delivery.

This message type should be used for application-critical data.

Available in iOS 7.0 and later.

Declared in MCSession.h.

MCSessionSendDataUnreliable

Messages to peers should be sent immediately without socket-level queueing. If a message cannot be sent immediately, it should be dropped. The order of messages is not guaranteed.

This message type should be used for data that ceases to be relevant if delayed, such as real-time gaming data.

Available in iOS 7.0 and later.

Declared in MCSession.h.

MCSessionState

Indicates the current state of a given peer within a session.

```
typedef NS_ENUM(NSInteger,
    MCSessionState) {
    MCSessionStateNotConnected,
    MCSessionStateConnecting,
    MCSessionStateConnected
};
```

Constants

MCSessionStateNotConnected

The peer is not (or is no longer) in this session.

Available in iOS 7.0 and later.

Declared in MCSession.h.

MCSessionStateConnecting

A connection to the peer is currently being established.

Available in iOS 7.0 and later.

Declared in MCSession.h.

MCSessionStateConnected

The peer is connected to this session.

Available in iOS 7.0 and later.

Declared in MCSession.h.

MCEncryptionPreference

Indicates whether a session should use encryption when communicating with nearby peers.

typedef NS_ENUM(NSInteger,

```
        MCEncryptionPreference) {
        MCEncryptionOptional                    = 0,
        MCEncryptionRequired                    = 1,
        MCEncryptionNone                        = 2,
};
```

Constants

MCEncryptionOptional

The session prefers to use encryption, but will accept unencrypted connections.

Available in iOS 7.0 and later.

Declared in MCSession.h.

MCEncryptionRequired

The session requires encryption.

Available in iOS 7.0 and later.

Declared in MCSession.h.

MCEncryptionNone

The session should not be encrypted.

Available in iOS 7.0 and later.

Declared in MCSession.h.

Error codes

Error codes found in MCErrorDomain (page 47) error domain NSError objects returned by methods in the Multipeer Connectivity framework.

```
enum MCErrorCode {
        MCErrorUnknown                          = 0,
        MCErrorNotConnected                     = 1,
        MCErrorInvalidParameter                 = 2,
        MCErrorUnsupported                      = 3,
        MCErrorTimedOut                         = 4,
        MCErrorCancelled                        = 5,
        MCErrorUnavailable                      = 6,
};
typedef NSInteger MCErrorCode;
```


Constants

MCErrorUnknown

An unknown error occurred.
Available in iOS 7.0 and later.

Declared in MCError.h.

MCErrorNotConnected

Your app attempted to send data to a peer that is not connected.

Available in iOS 7.0 and later.

Declared in MCError.h.

MCErrorInvalidParameter

Your app passed an invalid value as a parameter.

Available in iOS 7.0 and later.

Declared in MCError.h.

MCErrorUnsupported

The operation is unsupported. For example, this error is returned if you call sendResourceAtURL:withName:toPeer:withCompletionHandler: (page 41) with a URL that is neither a local file nor a web URL.

Available in iOS 7.0 and later.

Declared in MCError.h.

MCErrorTimedOut

The connection attempt timed out.

Available in iOS 7.0 and later.

Declared in MCError.h.

MCErrorCancelled

The operation was cancelled by the user.

Available in iOS 7.0 and later.

Declared in MCError.h.

MCErrorUnavailable

Multipeer connectivity is currently unavailable.

Available in iOS 7.0 and later.

Declared in MCError.h.

Multipeer Connectivity Error Domain

The error domain for errors specific to Multipeer Connectivity.

NSString * const MCErrorDomain;

Constants

MCErrorDomain

The NSError domain constant. If the domain value for an NSError object is equal to MCErrorDomain, then the error was produced by the Multipeer Connectivity framework itself, as opposed to a lower-level framework on which it depends.

Available in iOS 7.0 and later.

Declared in MCError.h.

Minimum and Maximum Supported Peers

Constants that define the minimum and maximum number of peers supported in a session.

```
NSUInteger const kMCSessionMaximumNumberOfPeers;
NSUInteger const kMCSessionMinimumNumberOfPeers;
```

Constants

kMCSessionMaximumNumberOfPeers

The maximum number of peers that a session can support, including the local peer.

Available in iOS 7.0 and later.

Declared in MCSession.h.

kMCSessionMinimumNumberOfPeers

The minimum number of peers that a session can support, including the local peer.

Available in iOS 7.0 and later.

Declared in MCSession.h.

Protocols

MCAdvertiserAssistantDelegate Class Reference

Conforms to NSObject

Framework	/System/Library/Frameworks/MultipeerConnectivity.framework
Availability	Available in iOS 7.0 and later.
Declared in	MCAdvertiserAssistant.h

Overview

The MCAdvertiserAssistantDelegate protocol describes the methods that the delegate object for an MCAdvertiserAssistant instance can implement to handle advertising-related events.

Tasks

Advertiser Assistant Delegate Methods

– advertiserAssitantWillPresentInvitation: (page 50)

Indicates that the advertiser assistant is about to present an invitation to the user.

– advertiserAssistantDidDismissInvitation: (page 49)

Indicates that the advertiser assistant finished showing the invitation to the user.

Instance Methods

advertiserAssistantDidDismissInvitation:

Indicates that the advertiser assistant finished showing the invitation to the user.

- (void)advertiserAssistantDidDismissInvitation:(MCAdvertiserAssistant *)advertiserAssistant

Parameters

advertiserAssistant

The advertiser assistant that finished showing an invitation.

Discussion

This call is intended to tell your app to resume any activity that it stopped doing while the invitation was onscreen. For example, it might resume computationally intensive UI updates for views that are no longer hidden by the invitation.

Availability
Available in iOS 7.0 and later.

Declared in
MCAdvertiserAssistant.h

advertiserAssitantWillPresentInvitation:

Indicates that the advertiser assistant is about to present an invitation to the user.

- (void)advertiserAssitantWillPresentInvitation:(MCAdvertiserAssistant *)advertiserAssistant

Parameters
advertiserAssistant

The advertiser assistant that is about to present an invitation to the user.

Discussion
This call is intended to allow your app to prepare for an invitation that will be presented to the user. For example, your app might stop performing computationally intensive UI updates for views that will be hidden by the invitation.

Availability
Available in iOS 7.0 and later.

Declared in
MCAdvertiserAssistant.h

MCBrowserViewControllerDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/MultipeerConnectivity.framework
Availability	Available in iOS 7.0 and later.

Declared in MCBrowserViewController.h

Overview

The MCBrowserViewControllerDelegate protocol defines the methods that your delegate object can implement to handle events related to the MCBrowserViewController class.

Tasks

Peer Notifications

– browserViewController:shouldPresentNearbyPeer:withDiscoveryInfo: (page 52)

Called when a new peer is discovered to decide whether to show it in the user interface.

User Action Notifications

– browserViewControllerDidFinish: (page 52) *required method*

Called when the browser view controller is dismissed with peers connected in a session. (required)

– browserViewControllerWasCancelled: (page 53) *required method*

Called when the user cancels the browser view controller. (required)

Instance Methods

browserViewController:shouldPresentNearbyPeer:withDiscoveryInfo:

Called when a new peer is discovered to decide whether to show it in the user interface.

- (BOOL)browserViewController:(MCBrowserViewController *)browserViewController
shouldPresentNearbyPeer:(MCPeerID *)peerID withDiscoveryInfo:(NSDictionary *)info

Parameters

browserViewController

The browser view controller object that discovered the new peer.

peerID

The unique ID of the nearby peer.

info

The info dictionary advertised by the discovered peer. For more information on the contents of this dictionary, see the documentation for initWithPeer:discoveryInfo:serviceType: (page 21) in *MCNearbyServiceAdvertiser Class Reference*.

Return Value

This delegate method should return YES if the newly discovered peer should be shown in the user interface, or NO otherwise.

Discussion

If this method is not provided, all peers are shown.

Availability

Available in iOS 7.0 and later.

Declared in

MCBrowserViewController.h

browserViewControllerDidFinish:

Called when the browser view controller is dismissed with peers connected in a session. (required)

- (void)browserViewControllerDidFinish:(MCBrowserViewController *)browserViewController

Instance Methods

Parameters

browserViewController

The view controller that was dismissed.

Discussion

This call is intended to inform your app that the user has connected with nearby peers in a session and that the browser view controller has been dismissed. Upon receiving this delegate method call, your app must call dismissViewControllerAnimated:completion: to dismiss the view controller. Your app can also begin sendingdatatoanyconnectedpeers,andshouldresumeanyUIupdatesthatitmayhavetemporarilysuspended while the view controller was onscreen.

Availability

Available in iOS 7.0 and later.

Declared in

MCBrowserViewController.h

browserViewControllerWasCancelled:

Called when the user cancels the browser view controller. (required)

```
- (void)browserViewControllerWasCancelled:(MCBrowserViewController
*)browserViewController
```

Parameters

browserViewController

The browser view controller that was canceled.

Discussion

This call is intended to inform your app that the view controller has been dismissed because the user canceled the discovery process and is no longer interested in creating a communication session.

When your app receives this delegate method call, your app must call dismissViewControllerAnimated:completion: to dismiss the view controller. Then, your app should handle the cancelation in whatever way is appropriate for your app, and then resume any UI updates that it may have temporarily suspended while the view controller was onscreen.

Availability

Available in iOS 7.0 and later.

Declared in

MCBrowserViewController.h

MCNearbyServiceAdvertiserDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/MultipeerConnectivity.framework
Availability	Available in iOS 7.0 and later.
Declared in	MCNearbyServiceAdvertiser.h

Overview

The MCNearbyServiceAdvertiserDelegate protocol describes the methods that the delegate object for an MCNearbyServiceAdvertiser instance can implement for handling events from the MCNearbyServiceAdvertiser class.

Tasks

Error Handling Delegate Methods

– advertiser:didNotStartAdvertisingPeer: (page 55)

Called when advertisement fails.

Invitation Handling Delegate Methods

– advertiser:didReceiveInvitationFromPeer:withContext:invitationHandler: (page 55) *required method*

Called when an invitation to join a session is received from a nearby peer. (required)

Instance Methods

advertiser:didNotStartAdvertisingPeer:

Called when advertisement fails.

- (void)advertiser:(MCNearbyServiceAdvertiser *)advertiser
didNotStartAdvertisingPeer:(NSError *)error

Parameters

advertiser

The advertiser object that failed to begin advertising.

error

An error object that indicates what went wrong.

Availability

Available in iOS 7.0 and later.

Declared in
MCNearbyServiceAdvertiser.h

advertiser:didReceiveInvitationFromPeer:withContext:invitationHandler:

Called when an invitation to join a session is received from a nearby peer. (required)

- (void)advertiser:(MCNearbyServiceAdvertiser *)advertiser
didReceiveInvitationFromPeer:(MCPeerID *)peerID withContext:(NSData *)context
invitationHandler:(void (^)(BOOL accept, MCSession *session))invitationHandler

Parameters

advertiser

The advertiser object that was invited to join the session.

peerID

The peer ID of the nearby peer that invited your app to join the session.

context

An arbitrary piece of data received from the nearby peer. This can be used to provide further information to the user about the nature of the invitation.

Important: The nearby peer should treat any data it receives as potentially untrusted. To learn more about working with untrusted data, read *Secure Coding Guide*.

invitationHandler

A block that your code must call to indicate whether the advertiser should accept or decline the invitation, and to provide a session with which to associate the peer that sent the invitation.

Availability

Available in iOS 7.0 and later.

Declared in

MCNearbyServiceAdvertiser.h

MCNearbyServiceBrowserDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/MultipeerConnectivity.framework
Availability	Available in iOS 7.0 and later.
Declared in	MCNearbyServiceBrowser.h

Overview

The MCNearbyServiceBrowserDelegate protocol defines methods that a MCNearbyServiceBrowser object's delegate can implement to handle browser-related events.

Tasks

Error Handling Delegate Methods

– browser:didNotStartBrowsingForPeers: (page 58)

Called when a browser failed to start browsing for peers.

Peer Discovery Delegate Methods

– browser:foundPeer:withDiscoveryInfo: (page 58) *required method*

Called when a nearby peer is found. (required)

– browser:lostPeer: (page 59) *required method*

Called when a nearby peer is lost. (required)

Instance Methods

browser:didNotStartBrowsingForPeers:

Called when a browser failed to start browsing for peers.

```
- (void)browser:(MCNearbyServiceBrowser *)browser didNotStartBrowsingForPeers:(NSError
  *)error
```

Parameters

browser

The browser object that failed to start browsing.

error

An error object indicating what went wrong.

Availability

Available in iOS 7.0 and later.

Declared in

MCNearbyServiceBrowser.h

browser:foundPeer:withDiscoveryInfo:

Called when a nearby peer is found. (required)

- (void)browser:(MCNearbyServiceBrowser *)browser foundPeer:(MCPeerID *)peerID
withDiscoveryInfo:(NSDictionary *)info

Parameters

browser

The browser object that found the nearby peer.

peerID

The unique ID of the peer that was found.

info

The info dictionary advertised by the discovered peer. For more information on the contents of this dictionary, see the documentation for initWithPeer:discoveryInfo:serviceType: (page 21) in *MCNearbyServiceAdvertiser Class Reference*.

Discussion

The peer ID provided to this delegate method can be used to invite the nearby peer to join a session.

Availability

Available in iOS 7.0 and later.

Declared in

MCNearbyServiceBrowser.h

browser:lostPeer:

Called when a nearby peer is lost. (required)

- (void)browser:(MCNearbyServiceBrowser *)browser lostPeer:(MCPeerID *)peerID

Parameters

browser

The browser object that lost the nearby peer.

peerID

The unique ID of the nearby peer that was lost.

Discussion

This callback informs your app that invitations can no longer be sent to a peer, and that your app should remove that peer from its user interface.

Important: Because there is a delay between when a host leaves a network and when the underlying Bonjour layer detects that it has left, the fact that your app has not yet received a disappearance callback

does not guarantee that it can communicate with the peer successfully.

Availability

Available in iOS 7.0 and later.

Declared in

MCNearbyServiceBrowser.h

MCSessionDelegate Protocol Reference

Adopted by	NSObject
Conforms to	NSObject
Framework	/System/Library/Frameworks/MultipeerConnectivity.framework
Availability	Available in iOS 7.0 and later.
Declared in	MCSession.h

Overview

The MCSessionDelegate protocol defines methods that a delegate of the MCSession class can implement to handle session-related events. For more information, see *MCSession Class Reference*.

Tasks

MCSession Delegate Methods

– session:didReceiveData:fromPeer: (page 62) *required method*

Indicates that an NSData object has been received from a nearby peer. (required)

– session:didStartReceivingResourceWithName:fromPeer:withProgress: (page 64) *required method*

Indicates that the local peer began receiving a resource from a nearby peer. (required)

– session:didFinishReceivingResourceWithName:fromPeer:atURL:withError: (page 61) *required method*

Indicates that the local peer finished receiving a resource from a nearby peer. (required)

– session:didReceiveStream:withName:fromPeer: (page 63) *required method*

Called when a nearby peer opens a byte stream connection to the local peer. (required)

– session:peer:didChangeState: (page 64) *required method*

Called when the state of a nearby peer changes. (required)

– session:didReceiveCertificate:fromPeer:certificateHandler: (page 61)

Called to validate the client certificate provided by a peer when the connection is first established.

Instance Methods

session:didFinishReceivingResourceWithName:fromPeer:atURL:withError:

Indicates that the local peer finished receiving a resource from a nearby peer. (required)

```
- (void)session:(MCSession *)session didFinishReceivingResourceWithName:(NSString
*)resourceName fromPeer:(MCPeerID *)peerID atURL:(NSURL *)localURL withError:(NSError
  *)error
```

Parameters

session

The session through which the data was received.

resourceName

The name of the resource, as provided by the sender.

peerID

The peer ID of the sender.

localURL

An NSURL object that provides the location of a temporary file containing the received data.

error

An error object indicating what went wrong if the file was not received successfully, or nil.

Discussion

The file referenced by resourceURL is a temporary file. Your app must either read the file or make a copy in a permanent location before this delegate method returns.

Availability

Available in iOS 7.0 and later.

Declared in

MCSession.h

session:didReceiveCertificate:fromPeer:certificateHandler:

Called to validate the client certificate provided by a peer when the connection is first established.

- (void)session:(MCSession *)session didReceiveCertificate:(NSArray *)certificate fromPeer:(MCPeerID *)peerID certificateHandler:(void (^)(BOOL accept))certificateHandler

Parameters

session

The session that the nearby peer wishes to join.

certificate

A certificate chain, presented as an array of SecCertificateRef certificate objects. The first certificate in this chain is the peer's certificate, which is derived from the identity that the peer provided when it called the initWithPeer:securityIdentity:encryptionPreference: (page 39) method. The other certificates are the (optional) additional chain certificates provided in that same array.

If the nearby peer did not provide a security identity, then this parameter's value is nil.

peerID

The peer ID of the sender.

certificateHandler

Your app should call this handler with a value of YES if the nearby peer should be allowed to join the session, or NO otherwise.

Discussion

Your app should inspect the nearby peer's certificate, and then should decide whether to trust that certificate. Upon making that determination, your app should call the provided certificateHandler block, passing either YES (to trust the nearby peer) or NO (to reject it).

For information about validating certificates, read *Cryptographic Services Guide*.

Important: The multipeer connectivity framework makes no attempt to validate the peer-provided identity

or certificates in any way. If your delegate does not implement this method, all certificates are accepted automatically.

Availability

Available in iOS 7.0 and later.

Declared in

MCSession.h

session:didReceiveData:fromPeer:

Indicates that an NSData object has been received from a nearby peer. (required)


```
- (void)session:(MCSession *)session didReceiveData:(NSData *)data fromPeer:(MCPeerID
  *)peerID
```

Parameters

session

The session through which the data was received.

data

An object containing the received data.

peerID

The peer ID of the sender.

Availability

Available in iOS 7.0 and later.

Declared in

MCSession.h

session:didReceiveStream:withName:fromPeer:

Called when a nearby peer opens a byte stream connection to the local peer. (required)

```
- (void)session:(MCSession *)session didReceiveStream:(NSInputStream *)stream
withName:(NSString *)streamName fromPeer:(MCPeerID *)peerID
```

Parameters

session

The session through which the byte stream was opened.

stream

An NSInputStream object that represents the local endpoint for the byte stream.

streamName

The name of the stream, as provided by the originator.

peerID

The peer ID of the originator of the stream.

Availability

Available in iOS 7.0 and later.

Declared in

MCSession.h

session:didStartReceivingResourceWithName:fromPeer:withProgress:

Indicates that the local peer began receiving a resource from a nearby peer. (required)

- (void)session:(MCSession *)session didStartReceivingResourceWithName:(NSString *)resourceName fromPeer:(MCPeerID *)peerID withProgress:(NSProgress *)progress

Parameters

session

The session that started receiving the resource.

resourceName

The name of the resource, as provided by the sender.

peerID

The sender’s peer ID.

progress

An NSProgress object that can be used to cancel the transfer or queried to determine how far the transfer has progressed.

Availability

Available in iOS 7.0 and later.

Declared in

MCSession.h

session:peer:didChangeState:

Called when the state of a nearby peer changes. (required)

- (void)session:(MCSession *)session peer:(MCPeerID *)peerID didChangeState:(MCSessionState)state

Parameters

session

The session that manages the nearby peer whose state changed.

peerID

The ID of the nearby peer whose state changed.

state

The new state of the nearby peer.

Discussion

This delegate method is called with the following state values when the nearby peer's state changes:

- MCSessionStateConnected (page 44)—the nearby peer accepted the invitation and is now connected to the session.
- MCSessionStateNotConnected (page 44)—the nearby peer declined the invitation, the connection could not be established, or a previously connected peer is no longer connected.

Availability

Available in iOS 7.0 and later.

Declared in

MCSession.h

Document Revision History

This table describes the changes to *Multipeer Connectivity Framework Reference*.

Date	Notes
2013-09-18	New document that describes an API for finding nearby iOS devices and adding them to a networking session.

Page 67

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010