Instant Message Framework Reference

Cocoa > Apple Applications

2007-07-08

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

Simultaneously published in the United States and Canada.

Contents

Introduction

Framework	/System/Library/Frameworks/InstantMessage.framework
Header file directories	InstantMessage.framework/Headers
Declared in	IMAVManager.h IMService.h

You can use the Instant Message framework to access iChat information and provide an auxiliary video source to iChat Theater.

Instant Message Framework Overview

The `IMService` class provides a way to integrate a variety of data about a user's iChat connections into your application. It provides information on which services the user is connected to (for example, AIM or Bonjour) their online screen names, their buddies, their current status on a given service (away, idle, available), idle times, and other presence-specific details. The API also provides notifications to update your applications when a user's status, information, status images, or service connections have changed. A variety of status notifications related to the user's status and preferences are posted by the IMService custom notification center. See the "Notifications" section in *IMService Class Reference* for more information.

The `IMAVManager` class allows you to create auxiliary video and audio sources that are played back through iChat AV during active chats. This is a mechanism for users to share other video sources with buddies. The `IMAVManager` class uses a delegation model in which you implement a video data source that provides each video frame via a callback message. You can implement your video source using either Core Video or OpenGL. You use Core Audio to handle audio channels. After setting up the audio and video sources, you begin playback by simply sending a `start` message to the shared `IMAVManager` object.

Introduction

Classes

IMAVManager Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/InstantMessage.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	InstantMessage/IMAVManager.h
Companion guide	Instant Message Programming Guide

Overview

The `IMAVManager` class is used to manage the state and configuration of auxiliary audio/video input to iChat AV—a feature that is called iChat Theater. There is only one shared instance of the `IMAVManager` class.

The `IMAVManager` shared object allows clients to provide audio and video to a running conference in iChat AV. Video is provided by supplying a data source object to receive periodic callbacks for individual frames, and audio is provided through an audio device and channel. The state of the shared `IMAVManager` object allows clients to configure the user interface appropriately.

Tasks

Creating an IMAVManager Object

`+ sharedAVManager` (page 10)
Returns the shared instance of the `IMAVManager` object, creating it if the object doesn't exist yet.

Getting and Setting Properties

`- state` (page 14)
Returns the current state of the receiver.

`- videoDataSource` (page 15)
Returns the receiver's video data source object.

`- setVideoDataSource:` (page 12)
Sets the receiver's video data source object that provides video data to iChat AV.

Starting and Stopping Audio/Video Content

- `start` (page 13)
  Starts sending audio and video to iChat AV.
- `stop` (page 14)
  Stops sending audio and video to iChat AV.

Optimizing Audio/Video Performance

- `setVideoOptimizationOptions:` (page 13)
  Sets the video optimization options.
- `videoOptimizationOptions` (page 15)
  Returns the video optimization options.

Managing Audio Channels

- `setNumberOfAudioChannels:` (page 12)
  Sets the number of audio channels.
- `numberOfAudioChannels` (page 12)
  Returns the number of audio channels.
- `audioDeviceUID` (page 11)
  Returns the audio device UID.
- `audioDeviceChannels` (page 11)
  Returns an array of audio device channel numbers used by the receiver.

Sharing Files

- `URLToShare` (page 15)
  Returns the file URL of the document that the user chose to share over iChat Theater.

Class Methods

sharedAVManager

Returns the shared instance of the `IMAVManager` object, creating it if the object doesn't exist yet.

```
+ (IMAVManager *)sharedAVManager
```

Return Value
The shared `IMAVManager` object.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMAVManager.h`

Instance Methods

audioDeviceChannels

Returns an array of audio device channel numbers used by the receiver.

```
- (NSArray *)audioDeviceChannels
```

Return Value
An array of audio device channel numbers. If the number of audio channels is set to `2`, then the first number in the array is the left channel and the second number is the right channel. Returns `nil` if the receiver is not in the `IMAVRunning` state. Also returns `nil` if the `setNumberOfAudioChannels:` (page 12) method is not invoked prior to invoking this method with `1` or `2` as the argument.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- audioDeviceUID` (page 11)

Declared In
`IMAVManager.h`

audioDeviceUID

Returns the audio device UID.

```
- (NSString *)audioDeviceUID
```

Return Value
A valid UID when the receiver is in the `IMAVRunning` state; otherwise, `nil`. Also returns `nil` if the `setNumberOfAudioChannels:` (page 12) method is not invoked prior to invoking this method with `1` or `2` as the argument.

Discussion
You can obtain the device by calling the `AudioHardwareGetProperty` function with the returned UID and the `kAudioHardwarePropertyDeviceForUID` constant as arguments.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- audioDeviceChannels` (page 11)
`- state` (page 14)

Declared In
`IMAVManager.h`

numberOfAudioChannels

Returns the number of audio channels.

```
- (NSInteger)numberOfAudioChannels
```

Return Value
The number of audio channels.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setNumberOfAudioChannels:` (page 12)

Declared In
`IMAVManager.h`

setNumberOfAudioChannels:

Sets the number of audio channels.

```
- (void)setNumberOfAudioChannels:(NSInteger)count
```

Parameters

count

The number of audio channels to configure. The allowed values are `0`, `1`, and `2`. If `0`, audio is disabled. If `1`, audio is set to mono, and if `2`, audio is stereo.

Discussion
Sets the number of audio channels that are configured after invoking `start` (page 13).

Availability
Available in Mac OS X v10.5 and later.

See Also
- `state` (page 14)
- `numberOfAudioChannels` (page 12)

Declared In
`IMAVManager.h`

setVideoDataSource:

Sets the receiver's video data source object that provides video data to iChat AV.

```
- (void)setVideoDataSource:(id)dataSource
```

Parameters

dataSource

An object that conforms to the `IMVideoDataSource` informal protocol. The object needs to respond to either the `renderIntoPixelBuffer:forTime:` and `getPixelBufferPixelFormat:` methods, or the`renderIntoOpenGLBuffer:onScreen:forTime:` and `getOpenGLBufferContext:pixelFormat:` methods for OpenGL content. Any NSView object can also be a video data source. The Instant Message framework adds video rendering capabilities to NSView and all its subclasses. Pass `nil` to remove the receiver's video data source. The data source is not retained by the receiver.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `videoDataSource` (page 15)

Declared In
`IMAVManager.h`

setVideoOptimizationOptions:

Sets the video optimization options.

```
- (void)setVideoOptimizationOptions:(IMVideoOptimizationOptions)options
```

Parameters

options

Indicates the characteristics of the video content. Possible values are described in "`IMVideoOptimizationOptions`" (page 17).

Discussion
Use this method to give hints to the receiver about the type of video content so it can optimize the CPU and bandwidth usage.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `state` (page 14)
- `videoOptimizationOptions` (page 15)

Declared In
`IMAVManager.h`

start

Starts sending audio and video to iChat AV.

```
- (void)start
```

Discussion
This method should be called when the receiver's state changes to `IMAVRequested` (page 16). If this method is successful, the receiver's state changes to `IMAVRunning` (page 17), after possibly changing momentarily to `IMAVStartingUp` (page 16) and `IMAVPending` (page 16).

Before invoking this method, you need to set the video source using the `setVideoDataSource:` (page 12) method to provide video content, or set the number of audio channels to greater than `0` using the `setNumberOfAudioChannels:` (page 12) method to provide audio content; otherwise, this method raises an exception.

This method has no effect if invoked when the receiver is not in the `IMAVRequested` (page 16) state.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- stop` (page 14)

Declared In
`IMAVManager.h`

state

Returns the current state of the receiver.

```
- (IMAVManagerState)state
```

Return Value
The current state of the receiver set by iChat AV. See "`IMAVManagerState`" (page 16) for a description of the possible return values.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMAVManager.h`

stop

Stops sending audio and video to iChat AV.

```
- (void)stop
```

Discussion
After this method is invoked the state changes to `IMAVRequested` (page 16), after possibly changing momentarily to `IMAVShuttingDown` (page 17).

Availability
Available in Mac OS X v10.5 and later.

See Also
`- start` (page 13)

Declared In
IMAVManager.h

URLToShare

Returns the file URL of the document that the user chose to share over iChat Theater.

```
- (NSURL *)URLToShare
```

Return Value
Returns the file URL of the document or `nil` if the receiver's state is `IMAVInactive` (page 16). Also returns `nil` if the user chose this application to share audio/video without a document.

Availability
Available in Mac OS X v10.5 and later.

Declared In
IMAVManager.h

videoDataSource

Returns the receiver's video data source object.

```
- (id)videoDataSource
```

Return Value
The receiver's video data source, or `nil` if it is not set.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setVideoDataSource:` (page 12)

Declared In
IMAVManager.h

videoOptimizationOptions

Returns the video optimization options.

```
- (IMVideoOptimizationOptions)videoOptimizationOptions
```

Return Value
Video optimization options. Possible values are described in "`IMVideoOptimizationOptions`" (page 17).

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setVideoOptimizationOptions:` (page 13)

Declared In
`IMAVManager.h`

Constants

IMAVManagerState

The state of an `IMAVManager` object.

```
enum {
    IMAVInactive        = 0,
    IMAVRequested       = 1,
    IMAVShuttingDown    = 2,
    IMAVStartingUp      = 3,
    IMAVPending         = 4,
    IMAVRunning         = 5
};
typedef NSUInteger IMAVManagerState;
```

Constants

`IMAVInactive`
An `IMAVManager` object is not available to send audio/video to iChat AV because the user has not started a session.

Available in Mac OS X v10.5 and later.

Declared in `IMAVManager.h`.

`IMAVRequested`
The user selected this client to begin iChat Theater. The client should send `start` (page 13) to the `IMAVManager` object to begin an iChat Theater session.

Available in Mac OS X v10.5 and later.

Declared in `IMAVManager.h`.

`IMAVStartingUp`
An `IMAVManager` object is starting up and will soon change to the `IMAVPending` or `IMAVRunning` state.

Available in Mac OS X v10.5 and later.

Declared in `IMAVManager.h`.

`IMAVPending`
iChat AV is not ready to receive content from an `IMAVManager` object.

An `IMAVManager` object may enter this state after the `start` (page 13) method is invoked when iChat AV is not ready to receive audio/video content. This state may be followed by `IMAVRunning` at any point.

Typically, this state is entered if either the user does not yet have a video chat active or some internal processing or negotiation needs to take place before auxiliary audio/video input can begin. If the user does not have a video chat active, the state changes to `IMAVRunning` when a chat starts.

Available in Mac OS X v10.5 and later.

Declared in `IMAVManager.h`.

`IMAVRunning`
: An `IMAVManager` object is actively sending audio/video content to iChat AV.

  You should not send audio/video content to an `IMAVManager` object until it reaches this state. For example, do not send audio/video content to an `IMAVManager` object immediately after sending `start` to the manager unless the manager is in this state.

  Available in Mac OS X v10.5 and later.

  Declared in `IMAVManager.h`.

`IMAVShuttingDown`
: An `IMAVManager` object is shutting down and will soon change to the `IMAVInactive` state.

  Available in Mac OS X v10.5 and later.

  Declared in `IMAVManager.h`.

Declared In
`InstantMessage/IMAVManager.h`

IMVideoOptimizationOptions

The characteristics of the video source to allow for optimization of CPU and bandwidth usage.

```
enum {
    IMVideoOptimizationDefault = 0,
    IMVideoOptimizationStills = 1 << 0,
    IMVideoOptimizationReplacement = 1 << 1,
};
typedef NSUInteger IMVideoOptimizationOptions;
```

Constants
`IMVideoOptimizationDefault`
: Shared video is played alongside the user's local video, and the video is full-motion. This is the default.

  Available in Mac OS X v10.5 and later.

  Declared in `IMAVManager.h`.

`IMVideoOptimizationStills`
: Shared video remains unchanged for many sequential frames (such as a photo slideshow). This is a hint that the required bandwidth is lower than that of full-motion video. Incorrectly setting this option may result in poor video quality.

  Available in Mac OS X v10.5 and later.

  Declared in `IMAVManager.h`.

`IMVideoOptimizationReplacement`
: Do not send the user's local video, instead devote full CPU and bandwidth resources to the shared video.

  Available in Mac OS X v10.5 and later.

  Declared in `IMAVManager.h`.

Declared In
`InstantMessage/IMAVManager.h`

Notifications

IMAVManagerStateChangedNotification

Posted by the `IMService` class custom notification center when the iChat AV input state changes.

The notification object is the shared `IMAVManager` object. This notification does not have a user information dictionary. Observers of this notification should send `state` (page 14) to the shared `IMAVManager` object to get the new state.

When the user selects this application or one of its documents to share over iChat Theater, the state of the shared `IMAVManager` object changes to `IMAVRequested` and this notification is sent.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMAVManager.h`

IMAVManagerURLToShareChangedNotification

Posted by the `IMService` class custom notification center when a new document is selected by the user to share over iChat Theater during a running session.

The notification object is the shared `IMAVManager` object. This notification does not have a user information dictionary. Observers of this notification should send `URLToShare` (page 15) to the shared `IMAVManager` object to get the new document. This notification is not sent the first time the state of the shared `IMAVManager` object changes to `IMAVRequested` to begin the session.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMAVManager.h`

IMService Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/InstantMessage.framework
Availability	Available in Mac OS X v10.4 and later.
Declared in	InstantMessage/IMService.h
Companion guide	Instant Message Programming Guide
Related sample code	ABPresence

Overview

The `IMService` class provides methods for getting information about an instant message service. Each `IMService` object represents one service available through iChat. Class methods such as `allServices` and `serviceWithName:` return these objects. Each object represents a single instant messaging service, allowing you to access the iChat status of the user, the user's list of buddies, and other information that can be integrated into your application. A variety of status notifications related to the user's status and preferences are posted by the IMService custom notification center.

Tasks

Accessing Instant Messaging Services

`+ allServices` (page 20)
Returns an array of the currently available services.

`+ serviceWithName:` (page 23)
Returns the specified service.

Accessing Service Attributes

`+ imageNameForStatus:` (page 21)
Returns the name of the image for the specified status of a person.

+ myIdleTime (page 22)
Returns the number of seconds that the current user is idle.

+ myStatus (page 22)
Returns the status of the current user.

+ notificationCenter (page 22)
Returns the custom notification center for the service.

- localizedName (page 24)
Returns the user-visible localized name of the service.

- localizedShortName (page 25)
Returns a short version, if available, of the user-visible localized name of the service.

- name (page 25)
Returns the fixed canonical name of the service.

- status (page 26)
Returns the login status of the service.

+ imageURLForStatus: (page 21) Deprecated in Mac OS X v10.5
Returns the URL of the image for the specified status of a person.

Accessing Buddies

- peopleWithScreenName: (page 26)
Returns Address Book entries that match the specified screen name of a buddy.

- screenNamesForPerson: (page 26)
Returns an array of strings that are valid screen names for the specified person.

- infoForAllScreenNames (page 23)
Returns information about all buddies for the service.

- infoForPreferredScreenNames (page 24)
Returns information about just the preferred accounts for all buddies.

- infoForScreenName: (page 24)
Returns information about a buddy with the specified screen name.

Class Methods

allServices

Returns an array of the currently available services.

```
+ (NSArray *)allServices
```

Return Value
Returns an `NSArray` of `IMService` objects corresponding to the current available services (AIM, Bonjour, and so on.).

Availability
Available in Mac OS X v10.4 and later.

See Also
`+ serviceWithName:` (page 23)

Declared In
`IMService.h`

imageNameForStatus:

Returns the name of the image for the specified status of a person.

```
+ (NSString *)imageNameForStatus:(IMPersonStatus)status
```

Parameters

status

The status of a person. See "IMPersonStatus" (page 30) for possible values.

Return Value
The name of an image that reflects the current online status of a person; it is usually a colored bubble or triangle.

Availability
Available in Mac OS X v10.5 and later.

See Also
`+ myStatus` (page 22)

Related Sample Code
ABPresence

Declared In
`IMService.h`

imageURLForStatus:

Returns the URL of the image for the specified status of a person. (Deprecated in Mac OS X v10.5.)

```
+ (NSURL *)imageURLForStatus:(IMPersonStatus)status
```

Parameters

status

The status of a person. See "IMPersonStatus" (page 30) for possible values.

Return Value
An image that reflects the current online status of a person; the image is usually a colored bubble or triangle.

Availability
Available in Mac OS X v10.4 and later.
Deprecated in Mac OS X v10.5.

See Also
`+ imageNameForStatus:` (page 21)

Declared In
`IMService.h`

myIdleTime

Returns the number of seconds that the current user is idle.

```
+ (NSDate *)myIdleTime
```

Return Value
The number of seconds that the current user is idle.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`IMService.h`

myStatus

Returns the status of the current user.

```
+ (IMPersonStatus)myStatus
```

Return Value
A code representing the status of the current user. See "IMPersonStatus" (page 30) for possible values.

Discussion
This status is global across all services.

Availability
Available in Mac OS X v10.4 and later.

See Also
`+ imageNameForStatus:` (page 21)

Declared In
`IMService.h`

notificationCenter

Returns the custom notification center for the service.

```
+ (NSNotificationCenter *)notificationCenter
```

Return Value
A custom notification center that manages `IMService` notifications.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
ABPresence

Declared In
`IMService.h`

serviceWithName:

Returns the specified service.

```
+ (IMService *)serviceWithName:(NSString *)name
```

Parameters

name

A service name as returned by a previous call to the `name` (page 25) method. Hard-coding the service names is not recommended.

Return Value
The service specified by *name*.

Availability
Available in Mac OS X v10.4 and later.

See Also
`+ allServices` (page 20)
`- name` (page 25)

Declared In
`IMService.h`

Instance Methods

infoForAllScreenNames

Returns information about all buddies for the service.

```
- (NSArray *)infoForAllScreenNames
```

Return Value
The dictionaries returned by `infoForScreenName:` (page 24) for all buddies.

Discussion
If the current user has multiple buddies for the same person (determined by the user's Address Book), this method returns the information for all of the accounts belonging to that person.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- infoForPreferredScreenNames` (page 24)
`- infoForScreenName:` (page 24)

Declared In
`IMService.h`

infoForPreferredScreenNames

Returns information about just the preferred accounts for all buddies.

```
- (NSArray *)infoForPreferredScreenNames
```

Return Value
An array of the dictionaries returned by `infoForScreenName:` (page 24) for all preferred accounts.

Discussion
If the current user has multiple buddies for the same person (determined by the user's Address Book), this method returns only the information for the preferred accounts belonging to that person. The preferred account is determined by iChat, using a combination of capabilities (video chat capability, audio chat capability, and so on), status (available, idle, away), and other user attributes.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `infoForAllScreenNames` (page 23)
- `infoForScreenName:` (page 24)

Declared In
`IMService.h`

infoForScreenName:

Returns information about a buddy with the specified screen name.

```
- (NSDictionary *)infoForScreenName:(NSString *)screenName
```

Parameters
screenName
A screen name for a buddy.

Return Value
Information about a buddy with the specified screen name. See "Screen Name Properties" (page 27) for the key-value pairs that appear in this dictionary.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `infoForAllScreenNames` (page 23)
- `infoForPreferredScreenNames` (page 24)

Declared In
`IMService.h`

localizedName

Returns the user-visible localized name of the service.

```
- (NSString *)localizedName
```

Return Value
The user-visible localized name of the service, such as "AOL Instant Messenger" or "Bonjour".

Availability
Available in Mac OS X v10.4 and later.

See Also
- `localizedShortName` (page 25)
- `name` (page 25)

Declared In
`IMService.h`

localizedShortName

Returns a short version, if available, of the user-visible localized name of the service.

```
- (NSString *)localizedShortName
```

Return Value
The user-visible short localized name of the service, such as "AOL".

Availability
Available in Mac OS X v10.4 and later.

See Also
- `localizedName` (page 24)
- `name` (page 25)

Declared In
`IMService.h`

name

Returns the fixed canonical name of the service.

```
- (NSString *)name
```

Return Value
The fixed canonical name of the service. This string is not intended to be visible to the user and therefore is not localized.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `localizedName` (page 24)
- `localizedShortName` (page 25)

Declared In
`IMService.h`

peopleWithScreenName:

Returns Address Book entries that match the specified screen name of a buddy.

```
- (NSArray *)peopleWithScreenName:(NSString *)screenName
```

Parameters

screenName

The screen name of a buddy.

Return Value

An array of Address Book entries that match the specified screen name of a buddy. Returns an empty array if there is no match.

Availability

Available in Mac OS X v10.4 and later.

See Also

- screenNamesForPerson: (page 26)

Related Sample Code

ABPresence

Declared In

IMService.h

screenNamesForPerson:

Returns an array of strings that are valid screen names for the specified person.

```
- (NSArray *)screenNamesForPerson:(ABPerson *)person
```

Parameters

person

An entry in the Address Book.

Return Value

An array of valid screen names for the specified person. Returns an empty array if there is no match.

Availability

Available in Mac OS X v10.4 and later.

See Also

- peopleWithScreenName: (page 26)

Declared In

IMService.h

status

Returns the login status of the service.

```
- (IMServiceStatus)status
```

Return Value
The login status of the service. One of the constants described in "IMServiceStatus" (page 29).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`IMService.h`

Constants

Screen Name Properties

Keys for information about a person logged in to an instant message service—specifically, a buddy that appears in the user's buddy list:

```
extern NSString *IMPersonAVBusyKey;
extern NSString *IMPersonCapabilitiesKey;
extern NSString *IMPersonEmailKey;
extern NSString *IMPersonFirstNameKey;
extern NSString *IMPersonIdleSinceKey;
extern NSString *IMPersonLastNameKey;
extern NSString *IMPersonPictureDataKey;
extern NSString *IMPersonScreenNameKey;
extern NSString *IMPersonServiceNameKey;
extern NSString *IMPersonStatusKey;
extern NSString *IMPersonStatusMessageKey;
```

Constants

`IMPersonAVBusyKey`
Used to obtain a person's busy status. The value is an `NSNumber` set to `0` if the person's audio/video capabilities are available, or `1` if they are busy.

Available in Mac OS X v10.4 and later.

Declared in `IMService.h`.

`IMPersonCapabilitiesKey`
Used to obtain a person's iChat capabilities. The value is an `NSArray` of capability properties. See `"Person Capability Values"` (page 29) for more information.

Available in Mac OS X v10.4 and later.

Declared in `IMService.h`.

`IMPersonEmailKey`
Used to obtain a person's email address. The value is an `NSString` containing the person's email address. This is a key used directly by Bonjour; however, if a person has an Address Book entry associated with a relevant AIM account, this key reflects the first email address of that person.

Available in Mac OS X v10.4 and later.

Declared in `IMService.h`.

`IMPersonFirstNameKey`
Used to obtain a person's first name. The value is an `NSString` containing the person's first name. This is a key used directly by Bonjour; however, if a person has an Address Book entry associated with a relevant AIM account, this key reflects the first name of that person.

Available in Mac OS X v10.4 and later.

Declared in `IMService.h`.

`IMPersonIdleSinceKey`
Used to obtain a person's idle status. The value is an `NSDate` containing the time, in seconds, since the last user activity. Available if the person's status is idle.

Available in Mac OS X v10.4 and later.

Declared in `IMService.h`.

`IMPersonLastNameKey`
Used to obtain a person's last name. The value is an `NSString` containing the person's last name. This is a key used directly by Bonjour; however, if a person has an Address Book entry associated with a relevant AIM account, this key reflects the last name of that person.

Available in Mac OS X v10.4 and later.

Declared in `IMService.h`.

`IMPersonPictureDataKey`
Used to obtain a person's image. The value is an `NSData` containing the image for the person's icon.

Available in Mac OS X v10.4 and later.

Declared in `IMService.h`.

`IMPersonScreenNameKey`
Used to obtain a person's screen name. The value is an `NSString` containing the service-specific identifier for a person. For example, "User123" or "steve@mac.com" for AIM, and "John Doe" for Bonjour.

Available in Mac OS X v10.4 and later.

Declared in `IMService.h`.

`IMPersonServiceNameKey`
Used to obtain a person's service name. The value is an `NSString` containing the name of the service this person belongs to.

Available in Mac OS X v10.4 and later.

Declared in `IMService.h`.

`IMPersonStatusKey`
Used to obtain a person's online status. The value is an `NSNumber` representing the current online status of the person, if known. See "`IMPersonStatus`" (page 30) for more information.

Available in Mac OS X v10.4 and later.

Declared in `IMService.h`.

`IMPersonStatusMessageKey`
Used to obtain a person's status message. The value is an `NSString` containing the person's current status message.

Available in Mac OS X v10.4 and later.

Declared in `IMService.h`.

Discussion
These keys appear in the dictionary returned by the `infoForScreenName:` (page 24) method.

Declared In
`InstantMessage/IMService.h`

Person Capability Values

A person's iChat capabilities accessed using the `IMPersonCapabilitiesKey` (page 27) key.

```
extern NSString *IMCapabilityAudioConference;
extern NSString *IMCapabilityDirectIM;
extern NSString *IMCapabilityFileSharing;
extern NSString *IMCapabilityFileTransfer;
extern NSString *IMCapabilityText;
extern NSString *IMCapabilityVideoConference;
```

Constants

`IMCapabilityAudioConference`
A person has audio chat capability.
Available in Mac OS X v10.4 and later.
Declared in `IMService.h`.

`IMCapabilityDirectIM`
A person has direct connect capability.
Available in Mac OS X v10.4 and later.
Declared in `IMService.h`.

`IMCapabilityFileSharing`
A person has file sharing capability.
Available in Mac OS X v10.4 and later.
Declared in `IMService.h`.

`IMCapabilityFileTransfer`
A person has file transfer capability.
Available in Mac OS X v10.4 and later.
Declared in `IMService.h`.

`IMCapabilityText`
A person has text capability.
Available in Mac OS X v10.4 and later.
Declared in `IMService.h`.

`IMCapabilityVideoConference`
A person has video chat capability.
Available in Mac OS X v10.4 and later.
Declared in `IMService.h`.

Declared In
`InstantMessage/IMService.h`

IMServiceStatus

The states of a service.

```
enum {
    IMServiceStatusLoggedOut,
    IMServiceStatusDisconnected,
    IMServiceStatusLoggingOut,
    IMServiceStatusLoggingIn,
    IMServiceStatusLoggedIn
};
typedef NSUInteger IMServiceStatus;
```

Constants

`IMServiceStatusLoggedOut`

A service is currently logged out.

Available in Mac OS X v10.4 and later.

Declared in `IMService.h`.

`IMServiceStatusDisconnected`

A service was disconnected, not by the user but by the system or because of an error.

Available in Mac OS X v10.4 and later.

Declared in `IMService.h`.

`IMServiceStatusLoggingOut`

A service is in the process of logging out.

Available in Mac OS X v10.4 and later.

Declared in `IMService.h`.

`IMServiceStatusLoggingIn`

A service is in the process of logging in.

Available in Mac OS X v10.4 and later.

Declared in `IMService.h`.

`IMServiceStatusLoggedIn`

A service is currently logged in.

Available in Mac OS X v10.4 and later.

Declared in `IMService.h`.

Declared In

`InstantMessage/IMService.h`

IMPersonStatus

The state of a person across all services.

```
enum {
    IMPersonStatusUnknown,
    IMPersonStatusOffline,
    IMPersonStatusIdle,
    IMPersonStatusAway,
    IMPersonStatusAvailable,
#if MAC_OS_X_VERSION_MAX_ALLOWED >= MAC_OS_X_VERSION_10_5
    IMPersonStatusNoStatus
#endif
};
typedef NSUInteger IMPersonStatus;
```

Constants

`IMPersonStatusUnknown`
The person's status is unknown.
Available in Mac OS X v10.4 and later.
Declared in `IMService.h`.

`IMPersonStatusOffline`
The person is currently offline.
Available in Mac OS X v10.4 and later.
Declared in `IMService.h`.

`IMPersonStatusIdle`
The person is currently idle.
Available in Mac OS X v10.4 and later.
Declared in `IMService.h`.

`IMPersonStatusAway`
The person is currently away.
Available in Mac OS X v10.4 and later.
Declared in `IMService.h`.

`IMPersonStatusAvailable`
The person is currently available.
Available in Mac OS X v10.4 and later.
Declared in `IMService.h`.

`IMPersonStatusNoStatus`
No status is available.
Available in Mac OS X v10.5 and later.
Declared in `IMService.h`.

Discussion

This is accessed using the `IMPersonStatusKey` (page 28) key for a buddy or returned by the `myStatus` (page 22) method for the current user.

Declared In

`InstantMessage/IMService.h`

Notifications

IMMyStatusChangedNotification

Posted by the `IMService` custom notification center when the local user changes their online status. The notification object is an `IMService` object. The user information dictionary does not contain keys. The receiver should send `myStatus` (page 22) to the notification object to get the new online status.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMService.h`

IMPersonInfoChangedNotification

Posted by the `IMService` custom notification center when a screen name changes some aspect of its published information. The notification object is an `IMService` object. The user information dictionary always contains the `IMPersonServiceNameKey` (page 28) key and may contain any of the other keys as described in `"Screen Name Properties"` (page 27). If a particular attribute is removed, the value for the relevant key is `NSNull`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`IMService.h`

IMPersonStatusChangedNotification

Posted by the `IMService` custom notification center when a different buddy (screen name) logs in, logs off, goes away, and so on. The notification object is an `IMService` object. The user information dictionary always contain the `IMPersonScreenNameKey` (page 28) and `IMPersonStatusKey` (page 28) keys, and no others.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`IMService.h`

IMServiceStatusChangedNotification

Posted by the `IMService` custom notification center when the status of a service changes—the current user logs in, logs off, goes away, and so on. The notification object is an `IMService` object. The user information dictionary does not contain keys. The receiver should send `status` (page 26) to the notification object to get the new service status.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`IMService.h`

IMStatusImagesChangedAppearanceNotification

Posted by the `IMService` custom notification center when the current user changes his or her preferred images for displaying status. The notification object is `nil`. This notification does not contain a user information dictionary. Use the `imageNameForStatus:` (page 21) method to get the new images.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`IMService.h`

Protocols

IMVideoDataSource Protocol Reference

(informal protocol)

Framework	/System/Library/Frameworks/InstantMessage.framework
Declared in	InstantMessage/IMAVManager.h
Companion guide	Instant Message Programming Guide

Overview

`IMVideoDataSource` is an informal protocol that an `IMAVManager` data source must conform to in order to provide video data to iChat AV.

To provide video when the `CVPixelBuffer` representation is preferred, the data source must implement both the `getPixelBufferPixelFormat:` (page 38) and `renderIntoPixelBuffer:forTime:` (page 39) methods. Otherwise, to provide video when the `CVOpenGLBuffers` representation is preferred, the data source must implement both the `getOpenGLBufferContext:pixelFormat:` (page 38) and `renderIntoOpenGLBuffer:onScreen:forTime:` (page 39) methods.

Tasks

Providing Pixel Buffered Video

- `getPixelBufferPixelFormat:` (page 38)
  Returns the pixel buffer format.
- `renderIntoPixelBuffer:forTime:` (page 39)
  Provides data for the next video frame using pixel buffering.

Providing OpenGL Buffered Video

- `getOpenGLBufferContext:pixelFormat:` (page 38)
  Returns the pixel OpenGL buffer context and pixel format.
- `renderIntoOpenGLBuffer:onScreen:forTime:` (page 39)
  Provides data for the next video frame using OpenGL buffering.

Instance Methods

getOpenGLBufferContext:pixelFormat:

Returns the pixel OpenGL buffer context and pixel format.

```
- (void)getOpenGLBufferContext:(CGLContextObj *)contextOut
    pixelFormat:(CGLPixelFormatObj *)pixelFormatOut
```

Parameters

contextOut

The OpenGL context to be used for the `CVOpenGLBufferRef` instances passed to the `renderIntoOpenGLBuffer:onScreen:forTime:` method.

pixelFormatOut

The OpenGL pixel format to be used for the `CVOpenGLBufferRef` instances passed to the `renderIntoOpenGLBuffer:onScreen:forTime:` method.

Discussion

This method is invoked once after `setVideoDataSource:` (page 12) is sent to an `IMAVManager` object.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- renderIntoOpenGLBuffer:onScreen:forTime:` (page 39)

Declared In

`IMAVManager.h`

getPixelBufferPixelFormat:

Returns the pixel buffer format.

```
- (void)getPixelBufferPixelFormat:(OSType *)pixelFormatOut
```

Parameters

pixelFormatOut

The pixel format to be used for the `CVPixelBufferRef` instances passed to the `renderIntoPixelBuffer:forTime:` method.

Discussion

This method is invoked once after `setVideoDataSource:` (page 12) is sent to an `IMAVManager` object.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- renderIntoPixelBuffer:forTime:` (page 39)

Declared In

`IMAVManager.h`

renderIntoOpenGLBuffer:onScreen:forTime:

Provides data for the next video frame using OpenGL buffering.

```
- (BOOL)renderIntoOpenGLBuffer:(CVOpenGLBufferRef)buffer onScreen:(int *)screenInOut
    forTime:(CVTimeStamp *)timeStamp
```

Parameters

buffer

The OpenGL buffer to fill. The receiver should call the `CVOpenGLBufferAttach` function and then fill the buffer with video data.

screenInOut

The recommended virtual screen number to pass to the `CVOpenGLBufferAttach` function for maximum efficiency. The receiver may use a different screen number, but it must write that value back into *screenInOut* before returning.

timeStamp

The frame time for which the buffer should be rendered.

You should render a video frame that corresponds to the supplied host time, `timeStamp->hostTime`, and before returning from this method, change the host time to the earliest time for which the rendered video is valid. For example, if the content is a movie, then set the host time to correspond to the rendered frame—typically, slightly earlier than the original host time. If the content is a photo slideshow, then set the host time to the time the image first appeared which can be several seconds before the original host time. Adjusting the time this way helps synchronize the audio with the video track.

Return Value

Returns `YES` if the buffer is successfully filled with new frame data. Returns `NO` if nothing changed or an error was encountered.

Discussion

This method is invoked each time a frame is sent to iChat AV. This method is not invoked on the main thread.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- getOpenGLBufferContext:pixelFormat:` (page 38)

Declared In

`IMAVManager.h`

renderIntoPixelBuffer:forTime:

Provides data for the next video frame using pixel buffering.

```
- (BOOL)renderIntoPixelBuffer:(CVPixelBufferRef)buffer forTime:(CVTimeStamp
    *)timeStamp
```

Parameters

buffer

The pixel buffer to fill with video data. The dimensions can vary. Use the `CVPixelBufferGetWidth` and `CVPixelBufferGetHeight` functions to get the dimensions each time this method is invoked.

timeStamp

The frame time for which the buffer should be rendered.

You should render a video frame that corresponds to the supplied host time, `timeStamp->hostTime`, and before returning from this method, change the host time to the earliest time for which the rendered video is valid. For example, if the content is a movie, then set the host time to correspond to the rendered frame—typically, slightly earlier than the original host time. If the content is a photo slideshow, then set the host time to the time the image first appeared which can be several seconds before the original host time. Adjusting the time this way helps synchronize the audio with the video track.

Return Value
Returns `YES` if the buffer is successfully filled with new frame data. Returns `NO` if nothing changed or an error was encountered.

Discussion
This method is invoked each time a frame is sent to iChat AV. This method is not invoked on the main thread.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- getPixelBufferPixelFormat:` (page 38)

Declared In
`IMAVManager.h`

Document Revision History

This table describes the changes to *Instant Message Framework Reference*.

Date	Notes
2007-07-08	Updated for Mac OS X v10.5.
2006-05-23	First publication of this content as a collection of separate documents.

Document Revision History

Index

M

N

P

R

S

U

V