Instant Message Framework Reference

Audio & Video

2009-07-30

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

Simultaneously published in the United States and Canada.

Contents

Introduction **Introduction 5**

Instant Message Framework Overview 5

Part I **Classes 7**

Chapter 1 **IMAVButton Class Reference 9**

Overview 9
Tasks 9
Class Methods 10
Instance Methods 12

Chapter 2 **IMAVControl Class Reference 13**

Overview 13
Tasks 13
Instance Methods 14

Chapter 3 **IMAVControlBar Class Reference 21**

Overview 21
Tasks 21
Instance Methods 22

Chapter 4 **IMAVManager Class Reference 25**

Overview 25
Tasks 25
Class Methods 26
Instance Methods 27
Constants 33
Notifications 35

Chapter 5 **IMAVSlider Class Reference 37**

Overview 37
Tasks 37
Class Methods 38
Instance Methods 38

Chapter 6 IMService Class Reference 41

Overview 41
Tasks 41
Class Methods 42
Instance Methods 45
Constants 49
Notifications 54

Part II Protocols 57

Chapter 7 IMVideoDataSource Protocol Reference 59

Overview 59
Tasks 59
Instance Methods 60

Part III Functions 63

Chapter 8 Instant Message Functions Reference 65

Overview 65
Functions 65

Document Revision History 67

Index 69

Introduction

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

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

Concurrency Note: The Instant Message framework is not thread safe. If you call functions or methods in this framework, you must do so exclusively on the main program thread. The only exception is the video frame callback methods for iChat Theater, which are invoked off the main thread. However, typically you do not need to use any Instant Message functions or methods inside those callbacks.

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

IMAVButton Class Reference

Inherits from	IMAVControl : NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/InstantMessage.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	InstantMessage/IMAVControl.h
Companion guide	Instant Message Programming Guide
Related sample code	iChatTheater

Overview

The `IMAVButton` class represents a type of control that you can add to the control bar used in iChat Theater, an instance of the `IMAVControlBar` class. Use the class methods defined in this class to add standard buttons to the control bar. You should not create instances of this class directly.

For example, this code fragment sets the target-action of the Forward button and adds it to the control bar:

```
[[IMAVButton forwardButton] setTarget:self];
[[IMAVButton forwardButton] setAction:@selector(forwardButtonAction:)];
[[[IMAVManager sharedAVManager] controlBar] addControl:[IMAVButton
forwardButton]];
```

Tasks

Getting Buttons

`+ backwardButton` (page 10)
Returns the backward button used in iChat Theater to control video playback.

`+ forwardButton` (page 10)
Returns the forward button used in iChat Theater to control video playback.

`+ muteButton` (page 11)
Returns the mute button used in iChat Theater to control video playback.

`+ playPauseButton` (page 11)
Returns the play button used in iChat Theater to control video playback.

Setting Properties

`- state` (page 12)
Returns the state of the button.

`- setState:` (page 12)
Sets the state of the button.

Class Methods

backwardButton

Returns the backward button used in iChat Theater to control video playback.

```
+ (IMAVButton *)backwardButton
```

Return Value
The backward button.

Discussion
It is your responsibility to set the target-action for this button. By default, clicking this button just changes the appearance of the button.

Availability
Available in Mac OS X v10.6 and later.

Related Sample Code
iChatTheater

Declared In
`IMAVControl.h`

forwardButton

Returns the forward button used in iChat Theater to control video playback.

```
+ (IMAVButton *)forwardButton
```

Return Value
The forward button.

Discussion
It is your responsibility to set the target-action for this button. By default, clicking this button just changes the appearance of the button.

Availability
Available in Mac OS X v10.6 and later.

Related Sample Code
iChatTheater

Declared In
`IMAVControl.h`

muteButton

Returns the mute button used in iChat Theater to control video playback.

```
+ (IMAVButton *)muteButton
```

Return Value
The mute button.

Discussion
It is your responsibility to set the target-action for this button. By default, clicking this button just changes the appearance of the button. If the state of the button is `1`, you should mute the audio source; otherwise, you should pause it.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`IMAVControl.h`

playPauseButton

Returns the play button used in iChat Theater to control video playback.

```
+ (IMAVButton *)playPauseButton
```

Return Value
The play button.

Discussion
It is your responsibility to set the target-action for this button. By default, clicking this button just changes the appearance of the button. If the state of the button is `1`, you should play the video source; otherwise, you should pause it.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`IMAVControl.h`

Instance Methods

setState:

Sets the state of the button.

```
- (void)setState:(NSInteger)value
```

Parameters

value

An integer value indicating the state of the button.

Discussion

The state depends on the type of button. For example, if the state of the button returned by the `playPauseButton` (page 11) method is `1`, the video source is playing; otherwise it is paused.

Availability

Available in Mac OS X v10.6 and later.

See Also

`- state` (page 12)

Declared In

`IMAVControl.h`

state

Returns the state of the button.

```
- (NSInteger)state
```

Return Value

An integer value indicating the state of the button.

Discussion

The state depends on the type of button. For example, if the state of the button returned by the `playPauseButton` (page 11) method is `1`, the video source is playing; otherwise it is paused.

Availability

Available in Mac OS X v10.6 and later.

See Also

`- setState:` (page 12)

Declared In

`IMAVControl.h`

IMAVControl Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/InstantMessage.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	InstantMessage/IMAVControl.h
Companion guide	Instant Message Programming Guide

Overview

The `IMAVControl` class is an abstract superclass for controls that you add to the iChat Theater control bar, an instance of the `IMAVControlBar` class. Similar to the `NSControl` class in the Application Kit framework, this class provide a common interface for setting a control's target, action, and value. You can also enable or disable a control. Use the various `IMAVControl` subclasses to create control objects.

Tasks

Setting the Target and Action

- `action` (page 14)
  Returns the action-message selector associated with the control.
- `setAction:` (page 16)
  Sets the receiver's action method to the specified selector.
- `setTarget:` (page 19)
  Sets the target object to receive action messages.
- `target` (page 20)
  Returns the receiver's target object.

Enabling and Disabling

- `isEnabled` (page 16)
  Returns whether the receiver reacts to mouse events.

- `setEnabled:` (page 17)
  Sets whether the receiver (and its cell) reacts to mouse events.

Getting and Setting the Value

- `doubleValue` (page 15)
  Returns the receiver's value as a double-precision floating-point number.
- `floatValue` (page 15)
  Returns the receiver's value as a single-precision floating-point number.
- `intValue` (page 16)
  Returns the receiver's value as an integer.
- `integerValue` (page 15)
  Returns the receiver's value as an `NSInteger` value.
- `setDoubleValue:` (page 17)
  Sets the receiver's value using a double-precision floating-point number.
- `setFloatValue:` (page 18)
  Sets the receiver's value using a single-precision floating-point number.
- `setIntValue:` (page 18)
  Sets the receiver's value using an integer.
- `setIntegerValue:` (page 18)
  Sets the receiver's value using an `NSInteger` value.

Getting and Setting the Tag

- `setTag:` (page 19)
  Sets the tag of the receiver.
- `tag` (page 19)
  Returns the tag identifying the receiver.

Instance Methods

action

Returns the action-message selector associated with the control.

```
- (SEL)action
```

Return Value
The action message that is sent to the target.

Availability
Available in Mac OS X v10.6 and later.

See Also
- setAction: (page 16)

Declared In
IMAVControl.h

doubleValue

Returns the receiver's value as a double-precision floating-point number.

```
- (double)doubleValue
```

Return Value
The value interpreted as a double-precision floating-point number.

Availability
Available in Mac OS X v10.6 and later.

See Also
- setDoubleValue: (page 17)

Declared In
IMAVControl.h

floatValue

Returns the receiver's value as a single-precision floating-point number.

```
- (float)floatValue
```

Return Value
The value interpreted as a single-precision floating-point number.

Availability
Available in Mac OS X v10.6 and later.

See Also
- setFloatValue: (page 18)

Declared In
IMAVControl.h

integerValue

Returns the receiver's value as an NSInteger value.

```
- (NSInteger)integerValue
```

Return Value
The value interpreted as an NSInteger value.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `setIntegerValue:` (page 18)

Declared In
`IMAVControl.h`

intValue

Returns the receiver's value as an integer.

```
- (int)intValue
```

Return Value
The value interpreted as an integer.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `setIntValue:` (page 18)

Declared In
`IMAVControl.h`

isEnabled

Returns whether the receiver reacts to mouse events.

```
- (BOOL)isEnabled
```

Return Value
`YES` if the receiver responds to mouse events; otherwise, `NO`.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `setEnabled:` (page 17)

Declared In
`IMAVControl.h`

setAction:

Sets the receiver's action method to the specified selector.

```
- (void)setAction:(SEL)aSelector
```

Parameters

aSelector

The action message to send to the target. Specify NULL to prevent action messages from being sent to the receiver's target.

Availability
Available in Mac OS X v10.6 and later.

See Also
- action (page 14)

Related Sample Code
iChatTheater

Declared In
IMAVControl.h

setDoubleValue:

Sets the receiver's value using a double-precision floating-point number.

```
- (void)setDoubleValue:(double)aDouble
```

Parameters

aDouble

The value interpreted as a double-precision floating-point number.

Availability
Available in Mac OS X v10.6 and later.

See Also
- doubleValue (page 15)

Declared In
IMAVControl.h

setEnabled:

Sets whether the receiver (and its cell) reacts to mouse events.

```
- (void)setEnabled:(BOOL)flag
```

Parameters

flag

YES if you want the receiver to react to mouse events; otherwise, NO.

Availability
Available in Mac OS X v10.6 and later.

See Also
- isEnabled (page 16)

Related Sample Code
iChatTheater

Declared In
IMAVControl.h

setFloatValue:

Sets the receiver's value using a single-precision floating-point number.

```
- (void)setFloatValue:(float)aFloat
```

Parameters

aFloat
The value interpreted as a single-precision floating-point number.

Availability
Available in Mac OS X v10.6 and later.

See Also
- floatValue (page 15)

Declared In
IMAVControl.h

setIntegerValue:

Sets the receiver's value using an NSInteger value.

```
- (void)setIntegerValue:(NSInteger)anInteger
```

Parameters

anInteger
The value interpreted as an NSInteger value.

Availability
Available in Mac OS X v10.6 and later.

See Also
- integerValue (page 15)

Declared In
IMAVControl.h

setIntValue:

Sets the receiver's value using an integer.

```
- (void)setIntValue:(int)anInt
```

Parameters

anInt
The value interpreted as an integer.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `intValue` (page 16)

Declared In
`IMAVControl.h`

setTag:

Sets the tag of the receiver.

```
- (void)setTag:(NSInteger)anInt
```

Parameters

anInt
The new tag.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `tag` (page 19)

Declared In
`IMAVControl.h`

setTarget:

Sets the target object to receive action messages.

```
- (void)setTarget:(id)anObject
```

Parameters

anObject
The new target object to associate with the receiver, or `nil` to remove the current target.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `target` (page 20)

Related Sample Code
iChatTheater

Declared In
`IMAVControl.h`

tag

Returns the tag identifying the receiver.

```
- (NSInteger)tag
```

Return Value
The tag of this control object.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- setTag:` (page 19)

Declared In
`IMAVControl.h`

target

Returns the receiver's target object.

```
- (id)target
```

Return Value
The object that receives the action message.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- setTarget:` (page 19)

Declared In
`IMAVControl.h`

IMAVControlBar Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/InstantMessage.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	InstantMessage/IMAVControl.h
Companion guide	Instant Message Programming Guide
Related sample code	iChatTheater

Overview

The `IMAVControlBar` class represents the control bar used in iChat Theater. However, the control bar does not appear unless you add controls. For example, you might add standard playback controls, like Forward and Previous buttons, if your client is a slideshow application. You set the target and actions for these controls before adding them to the control bar.

There is one `IMAVControlBar` object per client application that you obtain from the `IMAVManager` object using the `controlBar` (page 28) method. You should not create instances of `IMAVControlBar` directly. Once you've obtained the control bar object, use the methods in this class to add and remove controls. Use subclasses of `IMAVControl` to add and configure the standard controls.

Tasks

Getting Controls

- `controls` (page 22)
  Returns all the controls on the control bar.

Adding and Removing Controls

- `addControl:` (page 22)
  Adds the specified control to the control bar.

- `removeAllControls` (page 22)
  Removes all the controls from the control bar.
- `removeControl:` (page 23)
  Removes the specified control from the control bar.

Instance Methods

addControl:

Adds the specified control to the control bar.

```
- (void)addControl:(IMAVControl *)control
```

Parameters

control
  The control to add.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `removeAllControls` (page 22)
- `removeControl:` (page 23)

Related Sample Code
iChatTheater

Declared In
`IMAVControl.h`

controls

Returns all the controls on the control bar.

```
- (NSArray *)controls
```

Return Value
The controls on the control bar.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`IMAVControl.h`

removeAllControls

Removes all the controls from the control bar.

```
- (void)removeAllControls
```

Availability
Available in Mac OS X v10.6 and later.

See Also
- `addControl:` (page 22)
- `removeControl:` (page 23)

Related Sample Code
iChatTheater

Declared In
`IMAVControl.h`

removeControl:

Removes the specified control from the control bar.

```
- (void)removeControl:(IMAVControl *)control
```

Parameters

control
The control to remove.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `addControl:` (page 22)
- `removeAllControls` (page 22)

Declared In
`IMAVControl.h`

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
Related sample code	iChatTheater

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 26)
Returns the shared instance of the `IMAVManager` object, creating it if the object doesn't exist yet.

Getting and Setting Properties

`- state` (page 31)
Returns the current state of the receiver.

`- videoDataSource` (page 32)
Returns the receiver's video data source object.

- `setVideoDataSource:` (page 29)
  Sets the receiver's video data source object that provides video data to iChat AV.
- `controlBar` (page 28)
  Returns the receiver's control bar.

Starting and Stopping Audio/Video Content

- `start` (page 30)
  Starts sending audio and video to iChat AV.
- `stop` (page 31)
  Stops sending audio and video to iChat AV.

Optimizing Audio/Video Performance

- `setVideoOptimizationOptions:` (page 30)
  Sets the video optimization options.
- `videoOptimizationOptions` (page 32)
  Returns the video optimization options.

Managing Audio Channels

- `setNumberOfAudioChannels:` (page 29)
  Sets the number of audio channels.
- `numberOfAudioChannels` (page 28)
  Returns the number of audio channels.
- `audioDeviceUID` (page 27)
  Returns the audio device UID.
- `audioDeviceChannels` (page 27)
  Returns an array of audio device channel numbers used by the receiver.

Sharing Files

- `URLToShare` (page 31)
  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.

Related Sample Code
iChatTheater

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 29) 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 27)

Related Sample Code
iChatTheater

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 29) 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 27)
- `state` (page 31)

Related Sample Code
iChatTheater

Declared In
`IMAVManager.h`

controlBar

Returns the receiver's control bar.

```
- (IMAVControlBar *)controlBar
```

Return Value
The control bar configured for use by the receiver.

Availability
Available in Mac OS X v10.6 and later.

Related Sample Code
iChatTheater

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 29)

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 30).

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

See Also
- `state` (page 31)
- `numberOfAudioChannels` (page 28)

Related Sample Code
iChatTheater

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 32)

Related Sample Code
iChatTheater

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 34).

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 31)
- videoOptimizationOptions (page 32)

Related Sample Code
iChatTheater

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 33). If this method is successful, the receiver's state changes to IMAVRunning (page 34), after possibly changing momentarily to IMAVStartingUp (page 33) and IMAVPending (page 33).

Before invoking this method, you need to set the video source using the setVideoDataSource: (page 29) method to provide video content, or set the number of audio channels to greater than 0 using the setNumberOfAudioChannels: (page 29) 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 33) state.

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

See Also
- stop (page 31)

Related Sample Code
iChatTheater

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 33) for a description of the possible return values.

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

Related Sample Code
iChatTheater

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 33), after possibly changing momentarily to IMAVShuttingDown (page 34).

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

See Also
- start (page 30)

Related Sample Code
iChatTheater

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 33). 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 29)

Declared In
`IMAVManager.h`

videoOptimizationOptions

Returns the video optimization options.

```
- (IMVideoOptimizationOptions)videoOptimizationOptions
```

Return Value
Video optimization options. Possible values are described in "`IMVideoOptimizationOptions`" (page 34).

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

See Also
`- setVideoOptimizationOptions:` (page 30)

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 30) 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 30) 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`
: Set this flag if video will be largely static. This tells iChat Theater to attempt to optimize the video resolution for still images, rather than rapidly changing video.

  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 31) 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 31) 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`

IMAVSlider Class Reference

Inherits from	IMAVControl : NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/InstantMessage.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	InstantMessage/IMAVControl.h
Companion guide	Instant Message Programming Guide

Overview

The `IMAVSlider` class represents a slider control that you can add to the control bar used in iChat Theater, an instance of the `IMAVControlBar` class. The time slider can be used to scrub video or audio. It shows the elapsed and remaining time of a movie. Similar to other standard controls, you use a class method to create the slider. You should not create slider objects directly.

Use the `timeSlider` (page 38) class method to get the slider used by iChat Theater. Use the `setMinValue:` (page 39) and `setMaxValue:` (page 39) methods to set the bounds of the slider. For example, set the minimum value to the start time of a movie and the maximum value to the end time of a movie. Use the inherited `doubleValue` (page 15) method to get and the `setDoubleValue:` (page 17) method to set the current location in seconds. Add the slider to the control bar using the `addControl:` (page 22) method in the `IMAVControlBar` class.

For example, this code fragment configures a slider and adds it to the control bar:

```
[[IMAVSlider timeSlider] setMinValue:0.0];
[[IMAVSlider timeSlider] setMaxValue:3600.0];
[[[IMAVManager sharedAVManager] controlBar] addControl:[IMAVSlider timeSlider]];
```

Tasks

Getting the Slider

`+ timeSlider` (page 38)
Returns the time slider used in iChat Theater.

Changing the Value Limits

- `maxValue` (page 38)
  Returns the maximum value the slider can send to its target.
- `minValue` (page 39)
  Returns the minimum value the slider can send to its target.
- `setMaxValue:` (page 39)
  Sets the maximum value the slider can send to its target.
- `setMinValue:` (page 39)
  Sets the minimum value the slider can send to its target.

Class Methods

timeSlider

Returns the time slider used in iChat Theater.

```
+ (IMAVSlider *)timeSlider
```

Return Value
The time slider.

Discussion
It is your responsibility to set the target-action for this slider. By default, the slider does nothing.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`IMAVControl.h`

Instance Methods

maxValue

Returns the maximum value the slider can send to its target.

```
- (double)maxValue
```

Return Value
The slider's maximum value in seconds.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- setMaxValue:` (page 39)

Declared In
IMAVControl.h

minValue

Returns the minimum value the slider can send to its target.

```
- (double)minValue
```

Return Value
The slider's minimum value in seconds.

Availability
Available in Mac OS X v10.6 and later.

See Also
- setMinValue: (page 39)

Declared In
IMAVControl.h

setMaxValue:

Sets the maximum value the slider can send to its target.

```
- (void)setMaxValue:(double)aDouble
```

Parameters
aDouble
The maximum value of the slider in seconds.

Availability
Available in Mac OS X v10.6 and later.

See Also
- maxValue (page 38)

Declared In
IMAVControl.h

setMinValue:

Sets the minimum value the slider can send to its target.

```
- (void)setMinValue:(double)aDouble
```

Parameters
aDouble
The minimum value of the slider in seconds.

Availability
Available in Mac OS X v10.6 and later.

See Also
- minValue (page 39)

Declared In
IMAVControl.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 iChatTheater

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 42)
Returns an array of the currently available services.

`+ serviceWithName:` (page 45)
Returns the specified service.

Accessing Service Attributes

`+ imageNameForStatus:` (page 43)
Returns the name of the image for the specified status of a person.

+ myIdleTime (page 44)
Returns the number of seconds that the current user is idle.

+ myStatus (page 44)
Returns the status of the current user.

+ notificationCenter (page 44)
Returns the custom notification center for the service.

- localizedName (page 46)
Returns the user-visible localized name of the service.

- localizedShortName (page 47)
Returns a short version, if available, of the user-visible localized name of the service.

- name (page 47)
Returns the fixed canonical name of the service.

- status (page 48)
Returns the login status of the service.

+ imageURLForStatus: (page 43) Deprecated in Mac OS X v10.5
Returns the URL of the image for the specified status of a person.

Accessing Buddies

- peopleWithScreenName: (page 48)
Returns Address Book entries that match the specified screen name of a buddy.

- screenNamesForPerson: (page 48)
Returns an array of strings that are valid screen names for the specified person.

- infoForAllScreenNames (page 45)
Returns information about all buddies for the service.

- infoForPreferredScreenNames (page 46)
Returns information about just the preferred accounts for all buddies.

- infoForScreenName: (page 46)
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 45)

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 52) 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 44)

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 52) 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 43)

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 52) 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 43)

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
iChatTheater

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 47) 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 42)
`- name` (page 47)

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 46) 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 46)
`- infoForScreenName:` (page 46)

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 46) 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 45)
- `infoForScreenName:` (page 46)

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 49) for the key-value pairs that appear in this dictionary.

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

See Also
- `infoForAllScreenNames` (page 45)
- `infoForPreferredScreenNames` (page 46)

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 47)
- `name` (page 47)

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 46)
- `name` (page 47)

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 46)
- `localizedShortName` (page 47)

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 48)

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 48)

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 51).

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 51) 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 52) 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 46) method.

Declared In
`InstantMessage/IMService.h`

Person Capability Values

A person's iChat capabilities accessed using the `IMPersonCapabilitiesKey` (page 49) 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,
    IMPersonStatusNoStatus
};
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 50) key for a buddy or returned by the `myStatus` (page 44) 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 44) 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 50) key and may contain any of the other keys as described in `"Screen Name Properties"` (page 49). 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 50) and `IMPersonStatusKey` (page 50) 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 48) 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 43) 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 60) and `renderIntoPixelBuffer:forTime:` (page 61) methods. Otherwise, to provide video when the `CVOpenGLBuffers` representation is preferred, the data source must implement both the `getOpenGLBufferContext:pixelFormat:` (page 60) and `renderIntoOpenGLBuffer:onScreen:forTime:` (page 61) methods.

Tasks

Providing Pixel Buffered Video

- `getPixelBufferPixelFormat:` (page 60) *required method*
  Returns the pixel buffer format. (required)
- `renderIntoPixelBuffer:forTime:` (page 61) *required method*
  Provides data for the next video frame using pixel buffering. (required)

Providing OpenGL Buffered Video

- `getOpenGLBufferContext:pixelFormat:` (page 60) *required method*
  Returns the pixel OpenGL buffer context and pixel format. (required)
- `renderIntoOpenGLBuffer:onScreen:forTime:` (page 61) *required method*
  Provides data for the next video frame using OpenGL buffering. (required)

Instance Methods

getOpenGLBufferContext:pixelFormat:

Returns the pixel OpenGL buffer context and pixel format. (required)

```
- (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 29) is sent to an `IMAVManager` object.

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

See Also
`- renderIntoOpenGLBuffer:onScreen:forTime:` (page 61)

Declared In
`IMAVManager.h`

getPixelBufferPixelFormat:

Returns the pixel buffer format. (required)

```
- (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 29) is sent to an `IMAVManager` object.

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

See Also
`- renderIntoPixelBuffer:forTime:` (page 61)

Declared In
`IMAVManager.h`

renderIntoOpenGLBuffer:onScreen:forTime:

Provides data for the next video frame using OpenGL buffering. (required)

```
- (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 60)

Declared In

`IMAVManager.h`

renderIntoPixelBuffer:forTime:

Provides data for the next video frame using pixel buffering. (required)

```
- (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 60)

Declared In
`IMAVManager.h`

Functions

Instant Message Functions Reference

Framework: InstantMessage/IMService.h

Companion guide Instant Message Programming Guide

Overview

This document describes the functions found in the Instant Message framework.

Functions

This chapter describes the functions and function-like macros defined in the Instant Message framework.

IMComparePersonStatus

Compares two `IMPersonStatus` (page 52) values to determine which one has a higher availablity.

```
NSComparisonResult IMComparePersonStatus(IMPersonStatus status1, IMPersonStatus status2)
```

Parameters

status1

The status to compare with *status2*.

status2

The status to compare with *status1*.

Return Value

`NSOrderedAscending` if *status1* is less than *status2*, `NSOrderedDescending` if *status1* is greater than *status2*, or `NSOrderedSame` if *status1* is equal to *status2*.

Discussion

You should always use this function when comparing status values.

Availability

Available in Mac OS X v10.5 and later.

Related Sample Code

ABPresence

Declared In

`IMService.h`

Document Revision History

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

Date	Notes
2009-07-30	Added functions reference and concurrency information.
2009-05-13	Updated for Mac OS X v10.6 by adding the new IMAVButton, IMAVControl, IMAVControlBar, and IMAVSlider classes.
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

A

action instance method 14
addControl: instance method 22
allServices class method 42
audioDeviceChannels instance method 27
audioDeviceUID instance method 27

B

backwardButton class method 10

C

controlBar instance method 28
controls instance method 22

D

doubleValue instance method 15

F

floatValue instance method 15
forwardButton class method 10

G

getOpenGLBufferContext:pixelFormat: <NSObject> instance method 60
getPixelBufferPixelFormat: <NSObject> instance method 60

I

imageNameForStatus: class method 43
imageURLForStatus: class method 43
IMAVInactive constant 33
IMAVManagerState 33
IMAVManagerStateChangedNotification notification 35
IMAVManagerURLToShareChangedNotification notification 35
IMAVPending constant 33
IMAVRequested constant 33
IMAVRunning constant 34
IMAVShuttingDown constant 34
IMAVStartingUp constant 33
IMCapabilityAudioConference constant 51
IMCapabilityDirectIM constant 51
IMCapabilityFileSharing constant 51
IMCapabilityFileTransfer constant 51
IMCapabilityText constant 51
IMCapabilityVideoConference constant 51
IMComparePersonStatus function 65
IMMyStatusChangedNotification notification 54
IMPersonAVBusyKey constant 49
IMPersonCapabilitiesKey constant 49
IMPersonEmailKey constant 49
IMPersonFirstNameKey constant 50
IMPersonIdleSinceKey constant 50
IMPersonInfoChangedNotification notification 54
IMPersonLastNameKey constant 50
IMPersonPictureDataKey constant 50
IMPersonScreenNameKey constant 50
IMPersonServiceNameKey constant 50
IMPersonStatus 52
IMPersonStatusAvailable constant 53
IMPersonStatusAway constant 53
IMPersonStatusChangedNotification notification 54
IMPersonStatusIdle constant 53
IMPersonStatusKey constant 50
IMPersonStatusMessageKey constant 50
IMPersonStatusNoStatus constant 53

IMPersonStatusOffline constant 53
IMPersonStatusUnknown constant 53
IMServiceStatus 51
IMServiceStatusChangedNotification notification 54
IMServiceStatusDisconnected constant 52
IMServiceStatusLoggedIn constant 52
IMServiceStatusLoggedOut constant 52
IMServiceStatusLoggingIn constant 52
IMServiceStatusLoggingOut constant 52
IMStatusImagesChangedAppearanceNotification notification 55
IMVideoOptimizationDefault constant 34
IMVideoOptimizationOptions 34
IMVideoOptimizationReplacement constant 34
IMVideoOptimizationStills constant 34
infoForAllScreenNames instance method 45
infoForPreferredScreenNames instance method 46
infoForScreenName: instance method 46
integerValue instance method 15
intValue instance method 16
isEnabled instance method 16

L

localizedName instance method 46
localizedShortName instance method 47

M

maxValue instance method 38
minValue instance method 39
muteButton class method 11
myIdleTime class method 44
myStatus class method 44

N

name instance method 47
notificationCenter class method 44
numberOfAudioChannels instance method 28

P

peopleWithScreenName: instance method 48
Person Capability Values 51
playPauseButton class method 11

R

removeAllControls instance method 22
removeControl: instance method 23
renderIntoOpenGLBuffer:onScreen:forTime: <NSObject> instance method 61
renderIntoPixelBuffer:forTime: <NSObject> instance method 61

S

Screen Name Properties 49
screenNamesForPerson: instance method 48
serviceWithName: class method 45
setAction: instance method 16
setDoubleValue: instance method 17
setEnabled: instance method 17
setFloatValue: instance method 18
setIntegerValue: instance method 18
setIntValue: instance method 18
setMaxValue: instance method 39
setMinValue: instance method 39
setNumberOfAudioChannels: instance method 29
setState: instance method 12
setTag: instance method 19
setTarget: instance method 19
setVideoDataSource: instance method 29
setVideoOptimizationOptions: instance method 30
sharedAVManager class method 26
start instance method 30
state instance method 12, 31
status instance method 48
stop instance method 31

T

tag instance method 19
target instance method 20
timeSlider class method 38

U

URLToShare instance method 31

V

videoDataSource instance method 32

videoOptimizationOptions instance method 32