NS_DEV_DOCFOR:objc_class:NXJournaler;, NXJournaler

Inherits From: Object

Declared In: appkit/NXJournaler.h

Class Description

The NXJournaler class defines an object that lets an application record and play back events and sounds, a process called *journaling*. By using an NXJournaler object, an application can journal events flowing to one or more applicationsÐincluding itself. Optionally, sound can be recorded synchronously with the events. Later, the recorded events and sound can be played back, reenacting the activities as they occurred during the recording. With journaling, you can implement event-based macros or complete self-running demonstrations for your application.

Journaling is initiated by creating a new NXJournaler object and sending it a **setEventStatus:soundStatus:eventStream:soundfile:** message. The status arguments may have the values NX_STOPPED, NX_PLAYING, and NX_RECORDING. The event stream argument is a stream to record to or play back from. If you're recording, any data in the stream will be overwritten. It's not currently possible to add to the end of an existing event stream. The sound file argument is the name of a sound file to record to or play back from.

The new NXJournaler object becomes the application's master journaler, only one of which can exist in a given application. When recording begins for the first time, slave journalers are automatically created in the other NEXTSTEP application that are currently running (and that allow journaling, see below). The slave journalers work at the master's behest, either sending it copies of the events they receive (assuming the master is recording events) or playing back events that they receive from the master (assuming the master is playing back a prerecorded event stream). The master and slave journalers can be accessed by using the Application class's **masterJournaler** and **slaveJournaler** methods. The **masterJournaler** method provides a convenient way for you to access this journaler to start and stop recording. The **slaveJournaler** method is rarely needed, unless your application needs to respond differently when it's a part of a journaling session. If so, you can query the slave journaler to discover if a journaling session is under way and react accordingly.

When recording, by default all events going to any application are captured. Sometimes, you may not want certain applications to be recorded. For example, you might want to prevent the application that's recording the journal from being recorded. There are two ways to control this: with the defaults system and by sending a **setJournalable:** message to the Application object. Of the two, the defaults system is the more general.

To use the defaults system to control journaling, add this code to the **initialize** method of the object that will be controlling the journaling:

```
+ initialize
{
    static NXDefaultsVector myDefaults = {
        {"NXAllowJournaling", "NO"},
        {NULL}};
    NXRegisterDefaults([NXApp appName], myDefaults);
    return self;
}
```

This will prevent the application that contains the object from being journaled unless a user overrides the default for that application in the user's default database.

A user can also disallow journaling of any given application by adding an entry to the defaults database for that application. This would be done by entering the following command line in a Terminal window:

```
dwrite applicationName NXAllowJournaling NO
```

A less common way of allowing or disallowing journaling in an application is to send a **setJournalable:** message to the Application object. This allows more precise run-time control over journaling in that application.

Event recording may be aborted by clicking the right mouse button while holding down the Alternate key. (Note: For this to work, you must have the right mouse button enabled in the Preferences application.) Event playback can be aborted by typing a character with any key on the keyboard.

Instance Variables

None declared in this class.

Method Types

Initializing and freeing an NXJournaler
- init
- free

Controlling journaling
- setEventStatus:soundStatus:eventStream:soundfile:
- getEventStatus:soundStatus:eventStream:soundfile:
- setRecordDevice:
- recordDevice

Identifying associated objects
- speaker
- listener
- setDelegate:
- delegate

Instance Methods

NS_DEV_DOCFOR:objc_method:[NXJournaler-delegate];, delegate

- **delegate**

Returns the NXJournaler's delegate.

See also: - **setDelegate:**

NS_DEV_DOCFOR:objc_method:[NXJournaler-free];, free

- **free**

Frees the NXJournaler. Send this message to an NXJournaler after you're completely done with it.

NS_DEV_DOCFOR:objc_method:[NXJournaler-getEventStatus:soundStatus:eventStream:soundfile:];, getEventStatus:soundStatus:eventStream:soundfile:

- **getEventStatus:**(int *)*eventStatusPtr*
soundStatus:(int *)*soundStatusPtr*
eventStream:(NXStream **) *streamPtr*
soundfile:(char **)*soundfilePtr*

Provides status information about the NXJournaler. Values returned at *eventStatusPtr* and *soundStatusPtr* can be NX_PLAYING, NX_RECORDING, or NX_STOPPED. *streamPtr* is the address of a pointer to the event stream. *soundfilePtr* is the address of a pointer to the name of the sound file. Any of the arguments may be NULL if you don't want that piece of information. Returns **self**.

See also: - setEventStatus:soundStatus:eventStream:soundfile:

NS_DEV_DOCFOR:objc_method:[NXJournaler-init];, init

- init

Initializes a newly allocated NXJournaler object. The delegate of the new object is **nil**. This is the designated initializer for an NXJournaler object. Returns **self**.

NS_DEV_DOCFOR:objc_method:[NXJournaler-listener];, listener

- listener

Returns the listener used by the NXJournaler to communicate with other applications.

See also: - speaker

NS_DEV_DOCFOR:objc_method:[NXJournaler-recordDevice];, recordDevice

- (int)**recordDevice**

Returns whether sound is recorded from the CODEC microphone or from the DSP. The return value is either NX_CODEC or NX_DSP.

See also: - setRecordDevice:

NS_DEV_DOCFOR:objc_method:[NXJournaler-setDelegate:];, setDelegate:

- setDelegate:*anObject*

Sets the delegate used by the NXJournaler. The delegate is sent the method **journalerDidEnd:** when either playing or recording the journal finishes. If the journal was aborted, the delegate will first receive the message **journalerDidUserAbort:**. Returns **self**.

See also: - delegate

NS_DEV_DOCFOR:objc_method:[NXJournaler-setEventStatus:soundStatus:eventStream:soundfile:];, setEventStatus:soundStatus:eventStream:soundfile:

- setEventStatus:(int)*eventStatus*
soundStatus:(int)*soundStatus*
eventStream:(NXStream *)*stream*
soundfile:(const char *)*soundfile*

Controls the recording and playback of events and sounds. This is the main control point of the NXJournaler. The arguments *eventStatus* and *soundStatus* may be independently set to NX_STOPPED, NX_PLAYING, NX_RECORDING. By setting *eventStatus* to NX_RECORDING and *soundStatus* to NX_STOPPED, it's possible to record events without the sound. By setting *eventStatus* to NX_PLAYING and *soundStatus* to NX_RECORDING, it's possible to dub new sound over an existing event track.

The *stream* argument is the stream to record events to or playback events from. When recording,

any preexisting data in the stream will be overwritten. It's not currently possible to record onto the end of an existing event stream.

The *soundfile* argument is the name of the file to record sound to or playback sound from.

If you logically OR NX_NONABORTABLEMASK into *eventStatus*, journaling will be made nonabortable.

See also: - **getEventStatus:soundStatus:eventStream:soundfile:**

NS_DEV_DOCFOR:objc_method:[NXJournaler-setRecordDevice:];, setRecordDevice:

- **setRecordDevice:**(int)*device*

Sets whether sound is recorded from the CODEC microphone (the default device) or from the DSP. The constants NX_CODEC and NX_DSP can be used to specify the device. The recording from the DSP assumes that a peripheral is sending CD-quality data (stereo,16-bit linear, 44.1 kHz) to the DSP port. However, to save space, the data is reduced to a 22.05-kHz, mono sound.

See also: - **recordDevice**

NS_DEV_DOCFOR:objc_method:[NXJournaler-speaker];, speaker

- **speaker**

Returns the speaker used by the NXJournaler to communicate with the other applications.

See also: - **listener**

Methods Implemented By The Delegate

NS_DEV_DOCFOR:objc_method:[NXJournaler-journalerDidEnd:];, journalerDidEnd:

- **journalerDidEnd:***journaler*

Responds to a message informing the delegate that recording or playback of the journal is finished or has been aborted.

See also: - **journalerDidUserAbort:**

NS_DEV_DOCFOR:objc_method:[NXJournaler-journalerDidUserAbort:];, journalerDidUserAbort:

- **journalerDidUserAbort:***journaler*

Responds to a message informing the delegate that the user has aborted the recording or playback session. A **journalerDidUserAbort:** message is sent when the NXJournaler in the controlling application receives notice from one of the controlled applications that the user has generated an abort event during recording or playback. The delegate receives this message just before the NXJournaler stops the recording or playback.

See also: - **journalerDidEnd:**