

---

# SBApplication Class Reference

[Cocoa > Interapplication Communication](#)



2007-05-29



Apple Inc.  
© 2007 Apple Inc.  
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

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

Apple, the Apple logo, Cocoa, iTunes, Mac, Mac OS, and Objective-C are trademarks of Apple Inc., registered in the United States and other countries.

Simultaneously published in the United States and Canada.

**Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.**

**IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY**

**DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.**

**THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.**

**Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.**

# Contents

---

## **SBApplication Class Reference 5**

---

Overview	5
Subclassing Notes	5
Tasks	6
Getting a Scriptable Application Instance	6
Initializing a Scriptable Application Object	6
Creating a Scripting Class	6
Controlling the Application	6
Getting Class Names and Codes	7
Managing the Delegate	7
Class Methods	7
applicationWithBundleIdentifier:	7
applicationWithProcessIdentifier:	8
applicationWithURL:	8
Instance Methods	9
activate	9
classForScriptingClass:	9
classNamesForCodes	10
codesForPropertyNames	10
delegate	11
initWithBundleIdentifier:	11
initWithProcessIdentifier:	12
initWithURL:	12
isRunning	13
launchFlags	13
sendMode	13
setDelegate:	14
setLaunchFlags:	14
setSendMode:	15
setTimeout:	15
timeout	16

---

## **Document Revision History 17**

---

## **Index 19**

---



# SBAApplication Class Reference

---

<b>Inherits from</b>	SBObject : NSObject
<b>Conforms to</b>	NSCoding NSCoding (SBObject) NSObject (NSObject)
<b>Framework</b>	/System/Library/Frameworks/ScriptingBridge.framework
<b>Availability</b>	Available in Mac OS X v10.5 and later
<b>Declared in</b>	SBAApplication.h
<b>Related sample code</b>	SBSystemPrefs

## Overview

The `SBAApplication` class provides a mechanism enabling an Objective-C program to send Apple events to a scriptable application and receive Apple events in response. It thereby makes it possible for that program to control the application and exchange data with it. Scripting Bridge works by bridging data types between Apple event descriptors and Cocoa objects.

Although `SBAApplication` includes methods that manually send and process Apple events, you should never have to call these methods directly. Instead, subclasses of `SBAApplication` implement application-specific methods that handle the sending of Apple events automatically.

For example, if you wanted to get the current iTunes track, you can simply use the `currentTrack` method of the dynamically defined subclass for the iTunes application—which handles the details of sending the Apple event for you—rather than figuring out the more complicated, low-level alternative:

```
[iTunes propertyWithCode:'pTrk'];
```

If you do need to send Apple events manually, consider using the `NSAppleEventDescriptor` class.

## Subclassing Notes

---

You rarely instantiate `SBAApplication` objects directly. Instead, you get the shared instance of a application-specific subclass typically by calling one of the `applicationWith...` class methods, using a bundle identifier, process identifier, or URL to identify the application.

## Tasks

### Getting a Scriptable Application Instance

- + [applicationWithBundleIdentifier:](#) (page 7)  
Returns the shared instance representing the target application specified by its bundle identifier.
- + [applicationWithProcessIdentifier:](#) (page 8)  
Returns the shared instance representing a target application specified by its process identifier.
- + [applicationWithURL:](#) (page 8)  
Returns the shared instance representing a target application specified by the given URL.

### Initializing a Scriptable Application Object

- [initWithBundleIdentifier:](#) (page 11)  
Returns an instance of an `SBApplication` subclass that represents the target application identified by the given bundle identifier.
- [initWithProcessIdentifier:](#) (page 12)  
Returns an instance of an `SBApplication` subclass that represents the target application identified by the given process identifier.
- [initWithURL:](#) (page 12)  
Returns an instance of an `SBApplication` subclass that represents the target application identified by the given URL.

### Creating a Scripting Class

- [classForScriptingClass:](#) (page 9)  
Returns a class object that represents a particular class in the target application.

### Controlling the Application

- [activate](#) (page 9)  
Moves the target application to the foreground immediately.
- [isRunning](#) (page 13)  
Returns whether the target application represented by the receiver is running.
- [launchFlags](#) (page 13)  
Returns the launch flags for the application represented by the receiver.
- [setLaunchFlags:](#) (page 14)  
Returns the launch flags for the application represented by the receiver.
- [sendMode](#) (page 13)  
Returns the mode for sending Apple events to the target application.
- [setSendMode:](#) (page 15)  
Sets the mode for sending Apple events to the target application.

- [timeout](#) (page 16)  
Returns the period the receiver will wait to receive reply Apple events.
- [setTimeout:](#) (page 15)  
TechPubs\_replace\_this

## Getting Class Names and Codes

- [classNamesForCodes](#) (page 10)  
Returns a dictionary mapping four-character class codes to the names of their corresponding Objective-C classes.
- [codesForPropertyNames](#) (page 10)  
Returns a dictionary mapping property keys to their corresponding four-character codes.

## Managing the Delegate

- [delegate](#) (page 11)  
Returns the error-handling delegate of the receiver.
- [setDelegate:](#) (page 14)  
Returns the error-handling delegate of the receiver.

## Class Methods

### **applicationWithBundleIdentifier:**

Returns the shared instance representing the target application specified by its bundle identifier.

```
+ (id)applicationWithBundleIdentifier:(NSString *)ident
```

#### **Parameters**

*ident*

A bundle identifier specifying an application that is OSA-compliant.

#### **Return Value**

An instance of a `SBApplication` subclass that represents the target application whose bundle identifier is *ident*. Returns `nil` if no such application can be found or if the application does not have a scripting interface.

#### **Discussion**

For applications that declare themselves to have a dynamic scripting interface, this method will launch the application if it is not already running.

#### **Availability**

Available in Mac OS X v10.5 and later.

#### **See Also**

- [initWithBundleIdentifier:](#) (page 11)

**Related Sample Code**

SBSystemPrefs

**Declared In**

SBApplication.h

**applicationWithProcessIdentifier:**

Returns the shared instance representing a target application specified by its process identifier.

```
+ (id)applicationWithProcessIdentifier:(pid_t)pid
```

**Parameters***pid*

The BSD process ID of a OSA-compliant application. Often you can get the process ID of a process using the `processIdentifier` method of `NSTask`.

**Return Value**

An instance of an `SBApplication` subclass that represents the target application whose process identifier is *pid*. Returns `nil` if no such application can be found or if the application does not have a scripting interface.

**Discussion**

You should avoid using this method unless you know nothing about a target application but its process ID. In most cases, it is better to use `classForApplicationWithBundleIdentifier:` (page 7), which will dynamically locate the application's path at runtime, or `classForApplicationWithURL:` (page 8), which is not dependent on the target application being open at the time the method is called.

**Availability**

Available in Mac OS X v10.5 and later.

**See Also**

- [initWithProcessIdentifier:](#) (page 12)

**Declared In**

SBApplication.h

**applicationWithURL:**

Returns the shared instance representing a target application specified by the given URL.

```
+ (id)applicationWithURL:(NSURL *)url
```

**Parameters***url*

The Universal Resource Locator (URL) locating an OSA-compliant application.

**Return Value**

An `SBApplication` subclass from which to generate a shared instance of the target application whose URL is *url*. Returns `nil` if no such application can be found or if the application does not have a scripting interface.



**Discussion**

For applications that declare themselves to have a dynamic scripting interface, this method will launch the application if it is not already running. This approach to initializing `SBApplication` objects should be used only if you know for certain the URL of the target application. In most cases, it is better to use `classForApplicationWithBundleIdentifier:` (page 7) which dynamically locates the target application at runtime.

This method currently supports file URLs (`file:`) and remote application URLs (`eppc:`). It checks whether a file exists at the specified path, but it does not check whether an application identified via `eppc:` exists.

**Availability**

Available in Mac OS X v10.5 and later.

**See Also**

- `initWithURL:` (page 12)

**Declared In**

`SBApplication.h`

## Instance Methods

**activate**

Moves the target application to the foreground immediately.

- (void)activate

**Discussion**

If the target application is not already running, this method launches it.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

`SBApplication.h`

**classForScriptingClass:**

Returns a class object that represents a particular class in the target application.

- (Class)classForScriptingClass:(NSString \*)className

**Parameters**

*className*

The name of the scripting class.

**Return Value**

A `Class` object representing the scripting class.

**Discussion**

You invoke this method on an instance of a scriptable application. Once you have the class object, you may allocate an instance of the class and appropriately the raw instance. Or you may use it in a call to `isKindOfClass:` to determine the class type of an object.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

`SBApplication.h`

**classNamesForCodes**

Returns a dictionary mapping four-character class codes to the names of their corresponding Objective-C classes.

```
- (NSDictionary *)classNamesForCodes
```

**Return Value**

A dictionary whose keys are four-character class codes of the external application (as `NSNumber` objects), and whose values are the names of the corresponding `SBObject` subclasses.

**Discussion**

The default implementation returns an empty dictionary. Application-specific subclasses return dictionaries tailored to the types of objects they support.

You should never call this method directly.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

`SBApplication.h`

**codesForPropertyNames**

Returns a dictionary mapping property keys to their corresponding four-character codes.

```
- (NSDictionary *)codesForPropertyNames
```

**Return Value**

A dictionary whose keys are the keys of properties of the external application, and whose values are the corresponding four-character codes (as `NSNumber` objects).

**Discussion**

The default implementation returns an empty dictionary. Application-specific subclasses return dictionaries tailored to the types of objects they support.

You should never call this method directly.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

SBApplication.h

**delegate**

Returns the error-handling delegate of the receiver.

- (id)delegate

**Return Value**

The object acting as error-handling delegate of the receiver.

**Availability**

Available in Mac OS X v10.5 and later.

**See Also**- [setDelegate:](#) (page 14)**Declared In**

SBApplication.h

**initWithBundleIdentifier:**Returns an instance of an `SBApplication` subclass that represents the target application identified by the given bundle identifier.- (id)initWithBundleIdentifier:(NSString \*)*ident***Parameters***ident*

A bundle identifier specifying an application that is OSA-compliant.

**Return Value**An initialized shared instance of an `SBApplication` subclass that represents a target application with the bundle identifier of *ident*. Returns `nil` if no such application can be found or if the application does not have a scripting interface.**Discussion**

If you must initialize an `SBApplication` object explicitly, you should use this initializer if possible; unlike [initWithProcessIdentifier:](#) (page 12) and [initWithURL:](#) (page 12), this method is not dependent on changeable factors such as the target application's path or process ID. Even so, you should rarely have to initialize an `SBApplication` object yourself; instead, you should initialize an application-specific subclass such as `iTunesApplication`.

Note that this method does not check whether an application with the given bundle identifier actually exists.

**Availability**

Available in Mac OS X v10.5 and later.

**See Also**+ [applicationWithBundleIdentifier:](#) (page 7)**Declared In**

SBApplication.h

## initWithProcessIdentifier:

Returns an instance of an `SBApplication` subclass that represents the target application identified by the given process identifier.

```
- (id)initWithProcessIdentifier:(pid_t)pid
```

### Parameters

*pid*

A BSD process ID specifying an application that is OSA-compliant. Often you can get the process ID of a process using the `processIdentifier` method of `NSTask`.

### Return Value

An initialized `SBApplication` that you can use to communicate with the target application specified by the process ID. Returns `nil` if no such application can be found or if the application does not have a scripting interface.

### Discussion

You should avoid using this method unless you know nothing about an external application but its PID. In most cases, it is better to use [initWithBundleIdentifier:](#) (page 11), which will dynamically locate the external application's path at runtime, or [initWithURL:](#) (page 12), which is not dependent on the external application being open at the time the method is called.

### Availability

Available in Mac OS X v10.5 and later.

### See Also

+ [applicationWithProcessIdentifier:](#) (page 8)

### Declared In

`SBApplication.h`

## initWithURL:

Returns an instance of an `SBApplication` subclass that represents the target application identified by the given URL.

```
- (id)initWithURL:(NSURL *)url
```

### Parameters

*url*

A Universal Resource Locator (URL) specifying an application that is OSA-compliant.

### Return Value

An initialized `SBApplication` that you can use to communicate with the target application specified by the process ID. Returns `nil` if an application could not be found or if the application does not have a scripting interface.

### Discussion

This approach to initializing `SBApplication` objects should be used only if you know for certain the URL of the target application. In most cases, it is better to use [classForApplicationWithBundleIdentifier:](#) (page 7) which dynamically locates the target application at runtime. Even so, you should rarely have to initialize an `SBApplication` yourself.

This method currently supports file URLs (`file:`) and remote application URLs (`eppc:`). It checks whether a file exists at the specified path, but it does not check whether an application identified via `eppc:` exists.

#### Availability

Available in Mac OS X v10.5 and later.

#### See Also

+ [applicationWithURL:](#) (page 8)

#### Declared In

`SBApplication.h`

## isRunning

Returns whether the target application represented by the receiver is running.

- (BOOL)isRunning

#### Return Value

YES if the application is running, NO otherwise.

#### Availability

Available in Mac OS X v10.5 and later.

#### Declared In

`SBApplication.h`

## launchFlags

Returns the launch flags for the application represented by the receiver.

- (LSLaunchFlags)launchFlags

#### Return Value

A mask specifying the launch flags that are used when the target application is launched. For more information, see *Launch Services Reference*.

#### Availability

Available in Mac OS X v10.5 and later.

#### See Also

- [setLaunchFlags:](#) (page 14)

#### Declared In

`SBApplication.h`

## sendMode

Returns the mode for sending Apple events to the target application.

- (AESendMode)sendMode

**Return Value**

A mask specifying the mode for sending Apple events to the target application. For more information, see *Apple Event Manager Reference*.

**Availability**

Available in Mac OS X v10.5 and later.

**See Also**

- [setSendMode:](#) (page 15)

**Declared In**

SBApplication.h

**setDelegate:**

Returns the error-handling delegate of the receiver.

```
- (void)setDelegate:(id)delegate
```

**Parameters**

*delegate*

The object acting as delegate of the receiver.

**Discussion**

The delegate should implement the `eventDidFail:withError:` method of the `SBApplicationDelegate` informal protocol.

**Availability**

Available in Mac OS X v10.5 and later.

**See Also**

- [delegate](#) (page 11)

**Declared In**

SBApplication.h

**setLaunchFlags:**

Returns the launch flags for the application represented by the receiver.

```
- (void)setLaunchFlags:(LSLaunchFlags)flags
```

**Parameters**

*flags*

A mask specifying the launch flags that are used when the target application is launched. For more information, see *Launch Services Reference*.

**Discussion**

The default `SBApplication` launch flags are `kLSLaunchDontAddToRecents` (so the target application is not added to the Recent Items menu), `kLSLaunchDontSwitch` (so the target application launches in the background), and `kLSLaunchAndHide` (so the target application is hidden as soon as it is launched).

**Availability**

Available in Mac OS X v10.5 and later.

**See Also**

- [launchFlags](#) (page 13)

**Declared In**

SBApplication.h

**setSendMode:**

Sets the mode for sending Apple events to the target application.

```
(void)setSendMode:(AESendMode)sendMode
```

**Parameters**

*sendMode*

A mask specifying the mode for sending Apple events to the target application. For a list of valid modes, see *Apple Event Manager Reference*.

**Discussion**

The default send mode is `kAEWaitReply`. If the send mode is something other than `kAEWaitReply`, the receiver might not correctly handle reply events from the target application.

**Availability**

Available in Mac OS X v10.5 and later.

**See Also**

- [sendMode](#) (page 13)

**Declared In**

SBApplication.h

**setTimeout:**

TechPubs\_replace\_this

```
(void)setTimeout:(long)timeout
```

**Parameters**

*timeout*

The time in ticks that the receiver will wait to receive a reply Apple event from the target application before giving up.

**Discussion**

The default timeout value is `kAEDefaultTimeout`, which is about a minute. If you want the receiver to wait indefinitely for reply Apple events, use `kNoTimeout`. For more information, see *Apple Event Manager Reference*.

**Availability**

Available in Mac OS X v10.5 and later.

**See Also**

- [timeout](#) (page 16)

**Declared In**

SBApplication.h

## **timeout**

Returns the period the receiver will wait to receive reply Apple events.

- (long)timeout

### **Return Value**

The time in ticks that the receiver will wait to receive a reply Apple event from the target application before giving up. For more information, see *Apple Event Manager Reference*.

### **Availability**

Available in Mac OS X v10.5 and later.

### **See Also**

- [setTimeout:](#) (page 15)

### **Declared In**

SBApplication.h



# Document Revision History

---

This table describes the changes to *SBAApplication Class Reference*.

Date	Notes
2007-05-29	New document that describes the automatically defined class through which Cocoa applications can communicate with scriptable applications using Objective-C.

## REVISION HISTORY

### Document Revision History

# Index

---

## A

---

activate **instance method** [9](#)  
applicationWithBundleIdentifier: **class method** [7](#)  
applicationWithProcessIdentifier: **class method** [8](#)  
applicationWithURL: **class method** [8](#)

## C

---

classForScriptingClass: **instance method** [9](#)  
classNamesForCodes **instance method** [10](#)  
codesForPropertyNames **instance method** [10](#)

## D

---

delegate **instance method** [11](#)

## I

---

initWithBundleIdentifier: **instance method** [11](#)  
initWithProcessIdentifier: **instance method** [12](#)  
initWithURL: **instance method** [12](#)  
isRunning **instance method** [13](#)

## L

---

launchFlags **instance method** [13](#)

## S

---

sendMode **instance method** [13](#)

setDelegate: **instance method** [14](#)  
setLaunchFlags: **instance method** [14](#)  
setSendMode: **instance method** [15](#)  
setTimeout: **instance method** [15](#)

## T

---

timeout **instance method** [16](#)