

---

# Security Foundation Framework Reference

Security



2006-05-23



Apple Inc.  
© 2003, 2006 Apple Computer, 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, 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

**Introduction**      **Security Foundation Framework Reference** 5

---

**Part I**              **Classes** 7

---

**SFAuthorization Class Reference** 9

---

Overview 9

Tasks 9

Class Methods 10

Instance Methods 12

**Document Revision History** 19

---

**Index** 21

---



# Security Foundation Framework Reference

---

<b>Framework</b>	/Library/Frameworks/SecurityFoundation.framework
<b>Header file directories</b>	/Library/Frameworks/SecurityFoundation.framework/Headers
<b>Declared in</b>	SFAuthorization.h

This document was previously titled *Authorization Services Objective-C Reference*, and it replaced the SFAuthorization chapter of the *Security Objective-C API* document. The other chapters of that document have been replaced by *Security Interface Framework Reference*.

The Security Foundation framework contains a single Objective-C class, `SFAuthorization`, which allows you to restrict a user's access to particular features in your Mac OS X application or daemon.



# Classes

---



# SFAuthorization Class Reference

---

<b>Inherits from</b>	NSObject
<b>Conforms to</b>	NSCoding NSObject (NSObject)
<b>Framework</b>	/Library/Frameworks/SecurityFoundation.framework
<b>Declared in</b>	SecurityFoundation/SFAuthorization.h
<b>Availability</b>	Available in Mac OS X v10.3 and later
<b>Companion guide</b>	Authorization Services Programming Guide

## Overview

The `SFAuthorization` class allows you to restrict a user's access to particular features in your Mac OS X application or daemon.

The `SFAuthorization` class is an Objective-C interface for some of the functions in the Authorization Services API. You can use the [`authorizationRef`](#) (page 12) method to obtain an authorization reference, used in other calls to Authorization Services functions. The Authorization Services API is documented in *Authorization Services C Reference*.

## Tasks

### Allocating and Initializing an Authorization Object

- + [`authorization`](#) (page 10)  
Returns an authorization object initialized with a default environment, flags, and rights.
- + [`authorizationWithFlags:rights:environment:`](#) (page 11)  
Returns an authorization object initialized with the specified flags, rights and environment.
- [`init`](#) (page 12)  
Initializes an authorization object with default environment, flags, and rights.
- [`initWithFlags:rights:environment:`](#) (page 12)  
Initializes an authorization object with the specified flags, rights, and environment.

## Obtaining an Authorization Reference

- [authorizationRef](#) (page 12)  
Returns the authorization reference for this object.

## Authorizing Rights

- [obtainWithRights:flags:environment:authorizedRights:error:](#) (page 14)  
Authorizes and preauthorizes rights to access a privileged operation and returns the granted rights.
- [obtainWithRight:flags:error:](#) (page 14)  
Authorizes and preauthorizes one specific right.
- [permitWithRight:flags:](#) (page 16) **Deprecated in Mac OS X v10.5**  
Authorizes and preauthorizes one specific right. (**Deprecated.** Use [obtainWithRight:flags:](#) (page 14) instead.)
- [permitWithRights:flags:environment:authorizedRights:](#) (page 17) **Deprecated in Mac OS X v10.5**  
Authorizes and preauthorizes rights to access a privileged operation and returns the granted rights. (**Deprecated.** Use [obtainWithRights:flags:environment:authorizedRights:error:](#) (page 14) instead.)

## Preventing Credentials from Being Shared

- [invalidateCredentials](#) (page 13)  
Prevents any rights that were obtained by this object from being preserved.

## Class Methods

### authorization

Returns an authorization object initialized with a default environment, flags, and rights.

```
+ (id)authorization
```

#### Return Value

The authorization object.

#### Availability

Available in Mac OS X v10.3 and later.

#### See Also

- + [authorizationWithFlags:rights:environment:](#) (page 11)
- [initWithFlags:rights:environment:](#) (page 12)

#### Declared In

SFAuthorization.h

**authorizationWithFlags:rights:environment:**

Returns an authorization object initialized with the specified flags, rights and environment.

```
+ (id)authorizationWithFlags:(AuthorizationFlags)flags rights:(const
    AuthorizationRights *)rights environment:(const AuthorizationEnvironment
    *)environment
```

**Parameters**

*flags*

A bit mask for specifying authorization options. Use the following option sets:

- Pass the constant `kAuthorizationFlagDefaults` if no options are necessary.
- Specify the `kAuthorizationFlagExtendRights` mask to request rights. You can also specify the `kAuthorizationFlagInteractionAllowed` mask to allow user interaction.
- Specify the `kAuthorizationFlagPartialRights` and `kAuthorizationFlagExtendRights` masks to request partial rights. You can also specify the `kAuthorizationFlagInteractionAllowed` mask to allow user interaction.
- Specify the `kAuthorizationFlagPreAuthorize` and `kAuthorizationFlagExtendRights` masks to preauthorize rights.
- Specify the `kAuthorizationFlagDestroyRights` mask to prevent the Security framework from preserving the rights obtained during this call.

*rights*

A pointer to a set of authorization rights you create. Pass `NULL` if the application requires no rights at this time.

*environment*

Data used when authorizing or preauthorizing rights. In Mac OS X v10.3 and later, you can pass icon or prompt data to be used in the authentication dialog box. Possible values for this parameter are listed in `Security.framework/Headers/AuthorizationTags.h`. The data passed in this parameter is not stored in the authorization reference; it is used only during authorization. If you are not passing any data in this parameter, pass the constant `kAuthorizationEmptyEnvironment`.

**Return Value**

The authorization object.

**Discussion**

Normally, such initialization is not required, as you pass in flags, rights, and environmental data when you request authorization.

**Availability**

Available in Mac OS X v10.3 and later.

**See Also**

+ [authorization](#) (page 10)

**Declared In**

`SFAuthorization.h`

## Instance Methods

### authorizationRef

Returns the authorization reference for this object.

```
- (AuthorizationRef)authorizationRef
```

#### Return Value

The authorization reference.

#### Discussion

You can use the authorization reference in calls to Authorization Services functions.

For additional information see *Authorization Services C Reference*.

#### Availability

Available in Mac OS X v10.3 and later.

#### Declared In

SFAuthorization.h

### init

Initializes an authorization object with default environment, flags, and rights.

```
- (id)init
```

#### Availability

Available in Mac OS X v10.5 and later.

#### Declared In

SFAuthorization.h

### initWithFlags:rights:environment:

Initializes an authorization object with the specified flags, rights, and environment.

```
- (id)initWithFlags:(AuthorizationFlags)flags rights:(const AuthorizationRights *)rights environment:(const AuthorizationEnvironment *)environment
```

**Parameters***flags*

A bit mask for specifying authorization options. Use the following option sets:

- Pass the constant `kAuthorizationFlagDefaults` if no options are necessary.
- Specify the `kAuthorizationFlagExtendRights` mask to request rights. You can also specify the `kAuthorizationFlagInteractionAllowed` mask to allow user interaction.
- Specify the `kAuthorizationFlagPartialRights` and `kAuthorizationFlagExtendRights` masks to request partial rights. You can also specify the `kAuthorizationFlagInteractionAllowed` mask to allow user interaction.
- Specify the `kAuthorizationFlagPreAuthorize` and `kAuthorizationFlagExtendRights` masks to preauthorize rights.
- Specify the `kAuthorizationFlagDestroyRights` mask to prevent the Security framework from preserving the rights obtained during this call.

*rights*

A pointer to a set of authorization rights you create. Pass `NULL` if the application requires no rights at this time.

*environment*

Data used when authorizing or preauthorizing rights. In Mac OS X v10.3 and later, you can pass icon or prompt data to be used in the authentication dialog box. Possible values for this parameter are listed in `Security/AuthorizationTags.h`. If you are not passing any data in this parameter, pass the constant `kAuthorizationEmptyEnvironment`.

**Return Value**

The authorization object.

**Discussion**

You can use this method to initialize an authorization object. Normally, such initialization is not required, as you pass in flags, rights, and environmental data when you request authorization.

**Availability**

Available in Mac OS X v10.3 and later.

**See Also**

+ [authorization](#) (page 10)

+`alloc` (`NSObject`)

+ [authorizationWithFlags:rights:environment:](#) (page 11)

- [obtainWithRights:flags:environment:authorizedRights:error:](#) (page 14)

**Declared In**

`SFAuthorization.h`

**invalidateCredentials**

Prevents any rights that were obtained by this object from being preserved.

- `(void)invalidateCredentials`

**Discussion**

This method effectively ensures that authorizations are not shared.

**Availability**

Available in Mac OS X v10.3 and later.

**See Also**

- [obtainWithRights:flags:environment:authorizedRights:error:](#) (page 14)
- [obtainWithRight:flags:](#) (page 14)
- + [authorizationWithFlags:rights:environment:](#) (page 11)
- [initWithFlags:rights:environment:](#) (page 12)

**Declared In**

SFAuthorization.h

**obtainWithRight:flags:error:**

Authorizes and preauthorizes one specific right.

```
- (BOOL)obtainWithRight:(AuthorizationString)rightName
    flags:(AuthorizationFlags)flags error:(NSError **)error
```

**Parameters**

*rightName*

The name of an authorization right.

*flags*

A bit mask for specifying authorization options. See

[obtainWithRights:flags:environment:authorizedRights:error:](#) (page 14) for details about possible flag values.

*error*

On completion, the result code returned by the method. See “Result Codes” in *Authorization Services C Reference*.

**Return Value**

YES if operation completes successfully.

**Discussion**

Use this method to authorize or preauthorize a single right.

**Availability**

Available in Mac OS X v10.3 and later.

**See Also**

- [obtainWithRights:flags:environment:authorizedRights:error:](#) (page 14)

**Declared In**

SFAuthorization.h

**obtainWithRights:flags:environment:authorizedRights:error:**

Authorizes and preauthorizes rights to access a privileged operation and returns the granted rights.

```
- (BOOL)obtainWithRights:(const AuthorizationRights *)rights
    flags:(AuthorizationFlags)flags environment:(const AuthorizationEnvironment
*)environment authorizedRights:(AuthorizationRights **)authorizedRights
    error:(NSError **)error;
```

### Parameters

#### *rights*

A pointer to a set of authorization rights you create. Pass NULL if the application requires no rights at this time.

#### *flags*

A bit mask for specifying authorization options. Use the following option sets:

- Pass the constant `kAuthorizationFlagDefaults` if no options are necessary.
- Specify the `kAuthorizationFlagExtendRights` mask to request rights. You can also specify the `kAuthorizationFlagInteractionAllowed` mask to allow user interaction.
- Specify the `kAuthorizationFlagPartialRights` and `kAuthorizationFlagExtendRights` masks to request partial rights. You can also specify the `kAuthorizationFlagInteractionAllowed` mask to allow user interaction.
- Specify the `kAuthorizationFlagPreAuthorize` and `kAuthorizationFlagExtendRights` masks to preauthorize rights.
- Specify the `kAuthorizationFlagDestroyRights` mask to prevent the Security framework from preserving the rights obtained during this call.

#### *environment*

Data used when authorizing or preauthorizing rights. In Mac OS X v10.3 and later, you can pass icon or prompt data to be used in the authentication dialog box. Possible values for this parameter are listed in `Security/AuthorizationTags.h`. If you are not passing any data in this parameter, pass the constant `kAuthorizationEmptyEnvironment`.

#### *authorizedRights*

A pointer to a newly allocated `AuthorizationRights` structure. On return, this structure contains the rights granted by the Security framework. If you do not require this information, pass NULL. If you specify the `kAuthorizationFlagPreAuthorize` mask in the `flags` parameter, the method returns all the requested rights, including those not granted, but the flags of the rights that could not be preauthorized include the `kAuthorizationFlagCanNotPreAuthorize` bit.

Free the memory associated with this set of rights by calling the Authorization Services function `AuthorizationFreeItemSet`.

#### *error*

On completion, the result code returned by the method. See “Result Codes” in *Authorization Services C Reference*.

### Return Value

YES if operation completes successfully.

### Discussion

There are three main reasons to use this method. The first reason is to preauthorize rights by specifying the `kAuthorizationFlagPreAuthorize`, `kAuthorizationFlagInteractionAllowed`, and `kAuthorizationFlagExtendRights` masks as authorization options. Preauthorization is most useful when a right has a zero timeout. For example, you can preauthorize in the application and if it succeeds, call the helper tool and request authorization. This eliminates calling the helper tool if the Security framework cannot later authorize the specified rights.

The second reason to use this method is to authorize rights before performing a privileged operation by specifying the `kAuthorizationFlagInteractionAllowed`, and `kAuthorizationFlagExtendRights` masks as authorization options.

The third reason to use this method is to authorize partial rights. By specifying the `kAuthorizationFlagPartialRights`, `kAuthorizationFlagInteractionAllowed`, and `kAuthorizationFlagExtendRights` masks as authorization options, the Security framework grants all rights it can authorize. On return, the authorized set contains all the rights.

If you do not specify the `kAuthorizationFlagPartialRights` mask and the Security framework denies at least one right, then the method returns `NO` and the error parameter returns `errAuthorizationDenied`.

If you do not specify the `kAuthorizationFlagInteractionAllowed` mask and the Security framework requires user interaction, then the method returns `NO` and the error parameter returns `errAuthorizationInteractionNotAllowed`.

If you specify the `kAuthorizationFlagInteractionAllowed` mask and the user cancels the authentication process, then the method returns `NO` and the error parameter returns `errAuthorizationCanceled`.

#### Availability

Available in Mac OS X v10.5 and later.

#### See Also

`AuthorizationFreeItemSet`

`AuthorizationCopyRights`

- [obtainWithRight:flags:](#) (page 14)

+ [authorizationWithFlags:rights:environment:](#) (page 11)

- [initWithFlags:rights:environment:](#) (page 12)

#### Declared In

`SFAuthorization.h`

## permitWithRight:flags:

Authorizes and preauthorizes one specific right. (Deprecated in Mac OS X v10.5. Use [obtainWithRight:flags:](#) (page 14) instead.)

```
- (OSStatus)permitWithRight:(AuthorizationString)rightName
    flags:(AuthorizationFlags)flags
```

#### Parameters

*rightName*

The name of an authorization right.

*flags*

A bit mask for specifying authorization options. See `permitWithRights:flags:environment:authorizedRights` for details about possible flag values.

#### Discussion

Use this method to authorize or preauthorize a single right.

#### Availability

Available in Mac OS X v10.3 and later.

Deprecated in Mac OS X v10.5.

#### See Also

- [permitWithRights:flags:environment:authorizedRights](#) (page 17)

#### Declared In

SFAuthorization.h

## permitWithRights:flags:environment:authorizedRights:

Authorizes and preauthorizes rights to access a privileged operation and returns the granted rights.

(Deprecated in Mac OS X v10.5. Use

[obtainWithRights:flags:environment:authorizedRights:error:](#) (page 14) instead.)

```
- (OSStatus)permitWithRights:(const AuthorizationRights *)rights
    flags:(AuthorizationFlags)flags environment:(const AuthorizationEnvironment
*)environment authorizedRights:(AuthorizationRights **)authorizedRights
```

#### Parameters

*rights*

A pointer to a set of authorization rights you create. Pass NULL if the application requires no rights at this time.

*flags*

A bit mask for specifying authorization options. Use the following option sets:

- Pass the constant `kAuthorizationFlagDefaults` if no options are necessary.
- Specify the `kAuthorizationFlagExtendRights` mask to request rights. You can also specify the `kAuthorizationFlagInteractionAllowed` mask to allow user interaction.
- Specify the `kAuthorizationFlagPartialRights` and `kAuthorizationFlagExtendRights` masks to request partial rights. You can also specify the `kAuthorizationFlagInteractionAllowed` mask to allow user interaction.
- Specify the `kAuthorizationFlagPreAuthorize` and `kAuthorizationFlagExtendRights` masks to preauthorize rights.
- Specify the `kAuthorizationFlagDestroyRights` mask to prevent the Security framework from preserving the rights obtained during this call.

*environment*

Data used when authorizing or preauthorizing rights. In Mac OS X v10.3 and later, you can pass icon or prompt data to be used in the authentication dialog box. Possible values for this parameter are listed in `Security/AuthorizationTags.h`. If you are not passing any data in this parameter, pass the constant `kAuthorizationEmptyEnvironment`.

*authorizedRights*

A pointer to a newly allocated `AuthorizationRights` structure. On return, this structure contains the rights granted by the Security framework. If you do not require this information, pass NULL. If you specify the `kAuthorizationFlagPreAuthorize` mask in the `flags` parameter, the method returns all the requested rights, including those not granted, but the flags of the rights that could not be preauthorized include the `kAuthorizationFlagCanNotPreAuthorize` bit.

Free the memory associated with this set of rights by calling the Authorization Services function `AuthorizationFreeItemSet`.

**Discussion**

There are three main reasons to use this method. The first reason is to preauthorize rights by specifying the `kAuthorizationFlagPreAuthorize`, `kAuthorizationFlagInteractionAllowed`, and `kAuthorizationFlagExtendRights` masks as authorization options. Preauthorization is most useful when a right has a zero timeout. For example, you can preauthorize in the application and if it succeeds, call the helper tool and request authorization. This eliminates calling the helper tool if the Security framework cannot later authorize the specified rights.

The second reason to use this method is to authorize rights before performing a privileged operation by specifying the `kAuthorizationFlagInteractionAllowed`, and `kAuthorizationFlagExtendRights` masks as authorization options.

The third reason to use this method is to authorize partial rights. By specifying the `kAuthorizationFlagPartialRights`, `kAuthorizationFlagInteractionAllowed`, and `kAuthorizationFlagExtendRights` masks as authorization options, the Security framework grants all rights it can authorize. On return, the authorized set contains all the rights.

If you do not specify the `kAuthorizationFlagPartialRights` mask and the Security framework denies at least one right, then the status of this method on return is `errAuthorizationDenied`.

If you do not specify the `kAuthorizationFlagInteractionAllowed` mask and the Security framework requires user interaction, then the status of this method on return is `errAuthorizationInteractionNotAllowed`.

If you specify the `kAuthorizationFlagInteractionAllowed` mask and the user cancels the authentication process, then the status of this method on return is `errAuthorizationCanceled`.

**Special Considerations**

The `authorizedRights` parameter is not supported in Mac OS X v10.3; use the Authorization Services function `AuthorizationCopyRights` instead. In Mac OS X v10.3 there is an error in the signature in the header file for this parameter. If you pass this argument as `(AuthorizationRights **)authorizedRights`, as shown in this document, it works as described.

**Availability**

Available in Mac OS X v10.3 and later.

Deprecated in Mac OS X v10.5.

**See Also**

`AuthorizationFreeItemSet`

`AuthorizationCopyRights`

- [permitWithRight:flags:](#) (page 16)

+ [authorizationWithFlags:rights:environment:](#) (page 11)

- [initWithFlags:rights:environment:](#) (page 12)

**Declared In**

`SFAuthorization.h`

# Document Revision History

---

This table describes the changes to *Security Foundation Framework Reference*.

Date	Notes
2006-05-23	Changed the title from "Authorization Services Objective-C Reference."

## REVISION HISTORY

### Document Revision History

# Index

---

## A

---

authorization **class method** [10](#)  
authorizationRef **instance method** [12](#)  
authorizationWithFlags:rights:environment:  
    **class method** [11](#)

## I

---

init **instance method** [12](#)  
initWithFlags:rights:environment: **instance  
method** [12](#)  
invalidateCredentials **instance method** [13](#)

## O

---

obtainWithRight:flags:error: **instance method** [14](#)  
obtainWithRights:flags:environment:  
    authorizedRights:error: **instance method** [14](#)

## P

---

permitWithRight:flags: **instance method** [16](#)  
permitWithRights:flags:environment:  
    authorizedRights: **instance method** [17](#)