SFAuthorizationView Class Reference

Security > Cocoa

2006-05-23

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

Simultaneously published in the United States and Canada.

Contents

Figures

SFAuthorizationView Class Reference

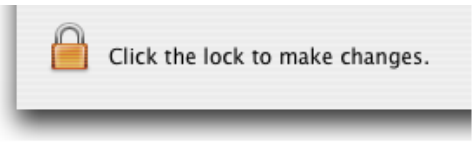
Inherits from	NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/SecurityInterface.framework
Availability	Available in Mac OS X v10.3 and later
Companion guide	Authorization Services Programming Guide
Declared in	SFAuthorizationView.h

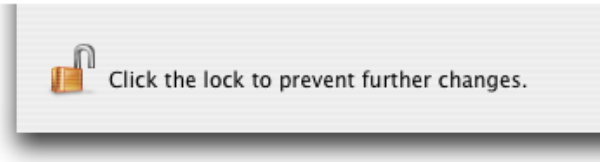
Overview

The `SFAuthorizationView` class displays a lock icon that can be used as a visual indication that a user interface has restricted access.

The lock appears locked when the user must be authorized and appears open when the user has been authorized. The closed and open lock icons of the authorization view are shown in the following figure.

Figure 1 Authorization view lock icon





When you add an authorization view as a custom view to a window or dialog box, you must initialize it before it displays correctly. To initialize the view, use the `setString:` (page 15) method to create a default rights structure (containing a prompt string) or the `setAuthorizationRights:` (page 12) method to specify a rights structure. You must also either specify automatic updates (`setAutoupdate:` (page 13) or `setAutoupdate:interval:` (page 13)) or perform a manual update (`updateStatus:` (page 15)) to set the lock icon to its initial state.

You can implement delegate methods that are invoked when the authorization view changes state. You can optionally implement the delegate methods to obtain the state of the authorization object when you are using an authorization view.

When the user clicks a locked authorization view icon, the Security Server displays an authentication dialog (to request a user name and password, for example). When the user provides the requested credentials, the lock icon unlocks and the user is considered preauthorized to perform the functions specified by the authorization rights structure. You can call the `updateStatus:` (page 15) method to determine whether the user has been preauthorized: this method returns `YES` if the view is in the unlocked state, `NO` otherwise. Before committing changes or performing actions that require authorization, you should check the user's authorization again, even if they are preauthorized.

The default behavior of this view is to preauthorize rights; if this is not possible it unlocks and waits for authorization to be checked when explicitly required.

Tasks

Setting Up the Authorization View

- `setString:` (page 15)
  Sets the requested-right string to use with the default authorization rights set.
- `setAuthorizationRights:` (page 12)
  Sets the authorization rights for this view.
- `setAutoupdate:` (page 13)
  Sets the authorization view to update itself automatically.
- `setAutoupdate:interval:` (page 13)
  Sets the authorization view to update itself at a specific interval.
- `setFlags:` (page 14)
  Sets the current authorization flags for the view.
- `setEnabled:` (page 14)
  Sets the current state of the authorization view.

Setting and Getting the Delegate for the View

- `setDelegate:` (page 14)
  Sets the delegate for this authorization view.
- `delegate` (page 11)
  Returns the delegate for this view.

Updating the View

- `updateStatus:` (page 15)
  Manually updates the authorization view.

Getting Information About the Authorization View

- `authorization` (page 10)
  Returns the authorization object associated with this view.
- `authorizationRights` (page 10)
  Returns the authorization rights for this view.
- `authorizationState` (page 10)
  Returns the current state of the authorization view.
- `isEnabled` (page 12)
  Indicates whether the authorization view is enabled (`YES`) or disabled (`NO`).

Setting the Authorization State

- `authorize:` (page 11)
  Attempts to unlock the lock icon in the view.
- `deauthorize:` (page 11)
  Sets the authorization state to unauthorized and locks the lock icon in the view.

Actions performed

- `authorizationViewCreatedAuthorization:` (page 16) *delegate method*
  Sent to the delegate to indicate the authorization object has been created or changed. If you have saved a copy of the authorization object for your own purposes, you should discard it and call `authorization` (page 10) for a new authorization object.
- `authorizationViewDidAuthorize:` (page 16) *delegate method*
  Sent to the delegate to indicate the user was authorized and the authorization view was changed to unlocked.
- `authorizationViewDidDeauthorize:` (page 17) *delegate method*
  Sent to the delegate to indicate the user was deauthorized and the authorization view was changed to locked.
- `authorizationViewReleasedAuthorization:` (page 17) *delegate method*
  Sent to the delegate to indicate that deauthorization is about to occur.

Action about to be performed

- `authorizationViewShouldDeauthorize:` (page 17) *delegate method*
  Sent to the delegate when a user clicks the open lock icon.

Instance Methods

authorization

Returns the authorization object associated with this view.

```
- (SFAuthorization *)authorization
```

Discussion
The authorization object is defined in *Security Foundation Framework Reference*.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`SFAuthorizationView.h`

authorizationRights

Returns the authorization rights for this view.

```
- (AuthorizationRights *)authorizationRights
```

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setAuthorizationRights:` (page 12)
- `setString:` (page 15)

Declared In
`SFAuthorizationView.h`

authorizationState

Returns the current state of the authorization view.

```
- (SFAuthorizationViewState)authorizationState
```

Availability
Available in Mac OS X v10.3 and later.

See Also
- `authorize:` (page 11)
- `deauthorize:` (page 11)

Declared In
`SFAuthorizationView.h`

authorize:

Attempts to unlock the lock icon in the view.

```
- (BOOL)authorize:(id)inSender
```

Parameters

inSender

The authorization view to unlock.

Discussion

This method has the same behavior as if the user clicked on the lock icon; if the user is authorized, the lock icon unlocks. If this method succeeds, it returns `YES`; if it fails, the lock icon remains locked and the method returns `NO`.

Availability

Available in Mac OS X v10.3 and later.

See Also

- `deauthorize:` (page 11)
- `authorizationState` (page 10)

Declared In

`SFAuthorizationView.h`

deauthorize:

Sets the authorization state to unauthorized and locks the lock icon in the view.

```
- (BOOL)deauthorize:(id)inSender
```

Parameters

inSender

The authorization view to lock.

Discussion

If this method succeeds, it returns `YES`; if it fails, the lock icon remains unlocked and the method returns `NO`.

Availability

Available in Mac OS X v10.3 and later.

See Also

- `authorize:` (page 11)

Declared In

`SFAuthorizationView.h`

delegate

Returns the delegate for this view.

```
- (id)delegate
```

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setDelegate:` (page 14)

Declared In
`SFAuthorizationView.h`

isEnabled

Indicates whether the authorization view is enabled (`YES`) or disabled (`NO`).

```
- (BOOL)isEnabled
```

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setEnabled:` (page 14)

Declared In
`SFAuthorizationView.h`

setAuthorizationRights:

Sets the authorization rights for this view.

```
- (void)setAuthorizationRights:(const AuthorizationRights *)authorizationRights
```

Parameters

`authorizationRights`

An authorization rights structure specifying the authorization rights represented by the authorization view.

Discussion
Either this method or the `setString:` method must be called before the view displays correctly.

The authorization rights structures are defined in `AuthorizationRights` in *Authorization Services C Reference*.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `authorizationRights` (page 10)
- `setString:` (page 15)

Declared In
`SFAuthorizationView.h`

setAutoupdate:

Sets the authorization view to update itself automatically.

```
- (void)setAutoupdate:(BOOL)autoupdate
```

Parameters

autoupdate

Specifies whether the authorization view should update itself automatically. Set to `YES` to enable autoupdates.

Discussion

If autoupdates are enabled and the authorization times out (for example), the authorization view automatically relocks. If autoupdates are disabled, you have to call the `updateStatus:` (page 15) method to manually update the view if the status changes when the user has not clicked on the lock icon. Autoupdates are disabled by default. Because autoupdates poll, they can affect system performance.

Availability

Available in Mac OS X v10.3 and later.

See Also

- `setAutoupdate:interval:` (page 13)
- `updateStatus:` (page 15)

Declared In

`SFAuthorizationView.h`

setAutoupdate:interval:

Sets the authorization view to update itself at a specific interval.

```
- (void)setAutoupdate:(BOOL)autoupdate interval:(NSTimeInterval)interval
```

Parameters

autoupdate

Specifies whether the authorization view should update itself automatically. Set to `YES` to enable autoupdates.

interval

If `autoupdate` is `YES`, sets the interval at which updates take place, in seconds.

Discussion

If autoupdates are enabled and the authorization times out (for example), the authorization view automatically relocks. If autoupdates are disabled, you have to call the `updateStatus:` (page 15) method to manually update the view if the status changes when the user has not clicked on the lock icon. Autoupdates are disabled by default. Because autoupdates poll, they can affect system performance. For that reason, you might want to set a time interval so that the polling does not take place as often.

Availability

Available in Mac OS X v10.3 and later.

See Also

- `setAutoupdate:` (page 13)

Declared In
SFAuthorizationView.h

setDelegate:

Sets the delegate for this authorization view.

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

Parameters

delegate

The object to which messages about the state of the authorization object should be sent.

Discussion
If you want to be notified of state changes (for example, when the user clicks the button), set a delegate and implement the delegate methods described in the delegate methods section.

Availability
Available in Mac OS X v10.3 and later.

See Also
- delegate (page 11).

Declared In
SFAuthorizationView.h

setEnabled:

Sets the current state of the authorization view.

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

Parameters

enabled

Specifies whether the authorization view should be enabled (YES) or disabled (NO).

Discussion
A disabled view is visible but dimmed.

Availability
Available in Mac OS X v10.3 and later.

See Also
- authorizationState (page 10)

Declared In
SFAuthorizationView.h

setFlags:

Sets the current authorization flags for the view.

```
- (void)setFlags:(AuthorizationFlags)flags
```

Parameters

flags

The authorization flags to set for this view.

Discussion

You can use this method to change the authorization flag settings made with the `setAuthorizationRights:` method or to specify flags other than the default (`kAuthorizationFlagDefaults`) used by the `setString:` method.

The authorization flags are described in Authorization Options in *Authorization Services C Reference*.

Availability

Available in Mac OS X v10.3 and later.

See Also

- `setAuthorizationRights:` (page 12)
- `setString:` (page 15)

Declared In

`SFAuthorizationView.h`

setString:

Sets the requested-right string to use with the default authorization rights set.

```
- (void)setString:(AuthorizationString)authorizationString
```

Parameters

authorizationString

The string to be displayed.

Discussion

This is a convenience method that creates an authorization rights set when you specify only the name of the requested right. The requested-right string is displayed in the Details pane of the user authentication dialog box. Either this method or the `setAuthorizationRights:` method must be called before the view displays correctly.

Availability

Available in Mac OS X v10.3 and later.

See Also

- `setAuthorizationRights:` (page 12)
- `authorizationRights` (page 10)

Declared In

`SFAuthorizationView.h`

updateStatus:

Manually updates the authorization view.

```
- (BOOL)updateStatus:(id)inSender
```

Parameters

inSender

The authorization view to update.

Discussion

Calls to `updateStatus:` return `YES` if in the unlocked state, `NO` otherwise.

If autoupdates have not been set, you must call `updateStatus` for the authorization view's initial state to display correctly. The Security Framework calls this method for you when you change the state of the lock (by calling `deauthorize:` (page 11), for example.

Availability

Available in Mac OS X v10.3 and later.

See Also

- `setAutoupdate:` (page 13)
- `setAutoupdate:interval:` (page 13).

Declared In

`SFAuthorizationView.h`

Delegate Methods

authorizationViewCreatedAuthorization:

Sent to the delegate to indicate the authorization object has been created or changed. If you have saved a copy of the authorization object for your own purposes, you should discard it and call `authorization` (page 10) for a new authorization object.

```
- (void)authorizationViewCreatedAuthorization:(SFAuthorizationView *)view
```

Availability

Available in Mac OS X v10.3 and later.

Declared In

`SFAuthorizationView.h`

authorizationViewDidAuthorize:

Sent to the delegate to indicate the user was authorized and the authorization view was changed to unlocked.

```
- (void)authorizationViewDidAuthorize:(SFAuthorizationView *)view
```

Availability

Available in Mac OS X v10.3 and later.

Declared In

`SFAuthorizationView.h`

authorizationViewDidDeauthorize:

Sent to the delegate to indicate the user was deauthorized and the authorization view was changed to locked.

```
- (void)authorizationViewDidDeauthorize:(SFAuthorizationView *)view
```

Availability
Available in Mac OS X v10.3 or later.

Declared In
SFAuthorizationView.h

authorizationViewReleasedAuthorization:

Sent to the delegate to indicate that deauthorization is about to occur.

```
- (void)authorizationViewReleasedAuthorization:(SFAuthorizationView *)view
```

Discussion
This method is called after deauthorization has been approved (either you called the `deauthorize:` method, or the user clicked an open lock icon and the `authorizationViewShouldDeauthorize:` delegate method did not cancel the operation), and before the user is deauthorized (that is, before the `authorizationViewDidDeauthorize:` delegate method is called).

Availability
Available in Mac OS X v10.3 and later.

See Also
- `deauthorize:` (page 11)
- `authorizationViewShouldDeauthorize:` (page 17)
- `authorizationViewDidDeauthorize:` (page 17)

Declared In
SFAuthorizationView.h

authorizationViewShouldDeauthorize:

Sent to the delegate when a user clicks the open lock icon.

```
- (BOOL)authorizationViewShouldDeauthorize:(SFAuthorizationView *)view
```

Discussion
The delegate can react to this before deauthorization happens and avoid it by returning `NO`. This delegate method is not called when you call the `deauthorize:` method.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `deauthorize:` (page 11)

Declared In
SFAuthorizationView.h

Constants

These constants are of type `SFAuthorizationViewState` and define the current state of the authorization view.

Constant	Description
`SFAuthorizationStartupState`	Indicates the state is starting up. Available in Mac OS X v10.3 and later. Declared in `SFAuthorizationView.h`.
`SFAuthorizationViewLockedState`	Indicates the state is locked. Available in Mac OS X v10.3 and later. Declared in `SFAuthorizationView.h`.
`SFAuthorizationViewInProgressState`	Indicates the state is in progress. Available in Mac OS X v10.3 and later. Declared in `SFAuthorizationView.h`.
`SFAuthorizationViewUnlockedState`	Indicates the state is unlocked. Available in Mac OS X v10.3 and later. Declared in `SFAuthorizationView.h`.

Document Revision History

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

Date	Notes
2006-05-23	First publication of this content as a separate document.

Document Revision History

Index