
NSURLConnection Class Reference

Networking, Internet, & Web: Protocol Streams



2009-08-17



Apple Inc.
© 2009 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, Mac, Mac OS, and Safari 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

NSURLConnection Class Reference 5

Overview	5
Tasks	6
Preflighting a Request	6
Loading Data Synchronously	6
Loading Data Asynchronously	6
Stopping a Connection	7
RunLoop Scheduling	7
Connection Authentication	7
Connection Data and Responses	7
Connection Completion	8
Class Methods	8
canHandleRequest:	8
connectionWithRequest:delegate:	8
sendSynchronousRequest:returningResponse:error:	9
Instance Methods	10
cancel	10
initWithRequest:delegate:	10
initWithRequest:delegate:startImmediately:	11
scheduleInRunLoop:forMode:	12
start	12
unsubscribeFromRunLoop:forMode:	12
Delegate Methods	13
connection:canAuthenticateAgainstProtectionSpace:	13
connection:didCancelAuthenticationChallenge:	14
connection:didFailWithError:	14
connection:didReceiveAuthenticationChallenge:	14
connection:didReceiveData:	15
connection:didReceiveResponse:	16
connection:didSendBodyData:totalBytesWritten:totalBytesExpectedToWrite:	16
connection:willCacheResponse:	17
connection:willSendRequest:redirectResponse:	18
connectionDidFinishLoading:	18
connectionShouldUseCredentialStorage:	19

Document Revision History 21

Index 23

NSURLConnection Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.2 with Safari 1.0 installed. Available in Mac OS X v10.2.7 and later.
Companion guide	URL Loading System
Declared in	NSURLConnection.h
Related sample code	CocoaSOAP ImageClient LSMSmartCategorizer URL CacheInfo XMLBrowser

Overview

An `NSURLConnection` object provides support to perform the loading of a URL request. The interface for `NSURLConnection` is sparse, providing only the controls to start and cancel asynchronous loads of a URL request.

`NSURLConnection`'s delegate methods allow an object to receive informational callbacks about the asynchronous load of a URL request. Other delegate methods provide facilities that allow the delegate to customize the process of performing an asynchronous URL load.

Note that these delegate methods will be called on the thread that started the asynchronous load operation for the associated `NSURLConnection` object.

The following contract governs the delegate methods defined in this interface:

- Zero or more `connection:willSendRequest:redirectResponse:` (page 18) messages will be sent to the delegate before any further messages are sent if it is determined that the download must redirect to a new location. The delegate can allow the redirect, modify the destination or deny the redirect.
- Zero or more `connection:didReceiveAuthenticationChallenge:` (page 14) messages will be sent to the delegate if it is necessary to authenticate in order to download the request and `NSURLConnection` does not already have authenticated credentials.

- Zero or more [connection:didCancelAuthenticationChallenge:](#) (page 14) messages will be sent to the delegate if the connection cancels the authentication challenge due to the protocol implementation encountering an error.
- Zero or more [connection:didReceiveResponse:](#) (page 16) messages will be sent to the delegate before receiving a [connection:didReceiveData:](#) (page 15) message. The only case where [connection:didReceiveResponse:](#) is not sent to a delegate is when the protocol implementation encounters an error before a response could be created.
- Zero or more [connection:didReceiveData:](#) (page 15) messages will be sent before any of the following messages are sent to the delegate: [connection:willCacheResponse:](#) (page 17), [connectionDidFinishLoading:](#) (page 18), [connection:didFailWithError:](#) (page 14).
- Zero or one [connection:willCacheResponse:](#) (page 17) messages will be sent to the delegate after [connection:didReceiveData:](#) (page 15) is sent but before a [connectionDidFinishLoading:](#) (page 18) message is sent.
- Unless a `NSURLConnection` receives a [cancel](#) (page 10) message, the delegate will receive one and only one of [connectionDidFinishLoading:](#) (page 18), or [connection:didFailWithError:](#) (page 14) message, but never both. In addition, once either of messages are sent, the delegate will receive no further messages for the given `NSURLConnection`.

`NSURLConnection` also has a convenience class method, [sendSynchronousRequest:returningResponse:error:](#) (page 9), to load a URL request synchronously.

`NSHTTPURLResponse` is a subclass of `NSURLResponse` that provides methods for accessing information specific to HTTP protocol responses. An `NSHTTPURLResponse` object represents a response to an HTTP URL load request.

Tasks

Preflighting a Request

- + [canHandleRequest:](#) (page 8)
Returns whether a request can be handled based on a "preflight" evaluation.

Loading Data Synchronously

- + [sendSynchronousRequest:returningResponse:error:](#) (page 9)
Performs a synchronous load of the specified URL request.

Loading Data Asynchronously

- + [connectionWithRequest:delegate:](#) (page 8)
Creates and returns an initialized URL connection and begins to load the data for the URL request.
- [initWithRequest:delegate:](#) (page 10)
Returns an initialized URL connection and begins to load the data for the URL request.

- [initWithRequest:delegate:startImmediately:](#) (page 11)
Returns an initialized URL connection and begins to load the data for the URL request, if specified.
- [start](#) (page 12)
Causes the receiver to begin loading data, if it has not already.

Stopping a Connection

- [cancel](#) (page 10)
Cancels an asynchronous load of a request.

RunLoop Scheduling

- [scheduleInRunLoop:forMode:](#) (page 12)
Determines the runloop and mode that the receiver uses to send delegate messages to the receiver.
- [unscheduleFromRunLoop:forMode:](#) (page 12)
Causes the receiver to stop sending delegate messages using the specified runloop and mode.

Connection Authentication

- [connection:canAuthenticateAgainstProtectionSpace:](#) (page 13) *delegate method*
Sent to determine whether the delegate is able to respond to a protection space's form of authentication.
- [connection:didCancelAuthenticationChallenge:](#) (page 14) *delegate method*
Sent when a connection cancels an authentication challenge.
- [connection:didReceiveAuthenticationChallenge:](#) (page 14) *delegate method*
Sent when a connection must authenticate a challenge in order to download its request.
- [connectionShouldUseCredentialStorage:](#) (page 19) *delegate method*
Sent to determine whether the URL loader should consult the credential storage for authenticating the connection.

Connection Data and Responses

- [connection:willCacheResponse:](#) (page 17) *delegate method*
Sent before the connection stores a cached response in the cache, to give the delegate an opportunity to alter it.
- [connection:didReceiveResponse:](#) (page 16) *delegate method*
Sent when the connection has received sufficient data to construct the URL response for its request.
- [connection:didReceiveData:](#) (page 15) *delegate method*
Sent as a connection loads data incrementally.
- [connection:didSendBodyData:totalBytesWritten:totalBytesExpectedToWrite:](#) (page 16) *delegate method*
Sent as the body (message data) of a request is transmitted (such as in an http POST request).

- `connection:willSendRequest:redirectResponse:` (page 18) *delegate method*
Sent when the connection determines that it must change URLs in order to continue loading a request.

Connection Completion

- `connection:didFailWithError:` (page 14) *delegate method*
Sent when a connection fails to load its request successfully.
- `connectionDidFinishLoading:` (page 18) *delegate method*
Sent when a connection has finished loading successfully.

Class Methods

canHandleRequest:

Returns whether a request can be handled based on a "preflight" evaluation.

```
+ (BOOL)canHandleRequest:(NSURLRequest *)request
```

Parameters

request

The request to evaluate.

Return Value

YES if a "preflight" operation determines that a connection with *request* can be created and the associated I/O can be started, NO otherwise.

Discussion

The result of this method is valid as long as no NSURLProtocol classes are registered or unregistered, and the specified *request* remains unchanged. Applications should be prepared to handle failures even if they have performed request preflighting by calling this method.

Availability

Available in Mac OS X v10.2 with Safari 1.0 installed.

Available in Mac OS X v10.2.7 and later.

See Also

```
+ registerClass:
+ unregisterClass:
```

Declared In

NSURLConnection.h

connectionWithRequest:delegate:

Creates and returns an initialized URL connection and begins to load the data for the URL request.

```
+ (NSURLConnection *)connectionWithRequest:(NSURLRequest *)request
  delegate:(id)delegate
```

Parameters*request*

The URL request to load. The *request* object is deep-copied as part of the initialization process. Changes made to *request* after this method returns do not affect the request that is used for the loading process.

delegate

The delegate object for the connection. The delegate will receive delegate messages as the load progresses. Messages to the delegate will be sent on the thread that calls this method. For the connection to work correctly the calling thread's run loop must be operating in the default run loop mode.]

Return Value

The URL connection for the URL request. Returns `nil` if a connection can't be created.

Availability

Available in Mac OS X v10.2 with Safari 1.0 installed.

Available in Mac OS X v10.2.7 and later.

See Also

- [initWithRequest:delegate:](#) (page 10)

Declared In

`NSURLConnection.h`

sendSynchronousRequest:returningResponse:error:

Performs a synchronous load of the specified URL request.

```
+ (NSData *)sendSynchronousRequest:(NSURLRequest *)request
    returningResponse:(NSURLResponse **)response error:(NSError **)error
```

Parameters*request*

The URL request to load. The *request* object is deep-copied as part of the initialization process. Changes made to *request* after this method returns do not affect the request that is used for the loading process.

response

Out parameter for the URL response returned by the server.

error

Out parameter used if an error occurs while processing the request. May be `NULL`.

Return Value

The downloaded data for the URL request. Returns `nil` if a connection could not be created or if the download fails.

Discussion

A synchronous load is built on top of the asynchronous loading code made available by the class. The calling thread is blocked while the asynchronous loading system performs the URL load on a thread spawned specifically for this load request. No special threading or run loop configuration is necessary in the calling thread in order to perform a synchronous load.

If authentication is required in order to download the request, the required credentials must be specified as part of the URL. If authentication fails, or credentials are missing, the connection will attempt to continue without credentials.

Availability

Available in Mac OS X v10.2 with Safari 1.0 installed.

Available in Mac OS X v10.2.7 and later.

Related Sample Code

XMLBrowser

Declared In

NSURLConnection.h

Instance Methods

cancel

Cancels an asynchronous load of a request.

- (void)cancel

Discussion

Once this method is called, the receiver's delegate will no longer receive any messages for this NSURLConnection.

Availability

Available in Mac OS X v10.2 with Safari 1.0 installed.

Available in Mac OS X v10.2.7 and later.

See Also

+ [connectionWithRequest:delegate:](#) (page 8)

- [initWithRequest:delegate:](#) (page 10)

Related Sample Code

LSMSmartCategorizer

Declared In

NSURLConnection.h

initWithRequest:delegate:

Returns an initialized URL connection and begins to load the data for the URL request.

- (id)initWithRequest:(NSURLRequest *)request delegate:(id)delegate

Parameters*request*

The URL request to load. The *request* object is deep-copied as part of the initialization process. Changes made to *request* after this method returns do not affect the request that is used for the loading process.

delegate

The delegate object for the connection. The delegate will receive delegate messages as the load progresses. Messages to the delegate will be sent on the thread that calls this method. By default, for the connection to work correctly the calling thread's run loop must be operating in the default run loop mode. See [scheduleInRunLoop:forMode:](#) (page 12) to change the runloop and mode.

Return Value

The URL connection for the URL request. Returns `nil` if a connection can't be initialized.

Availability

Available in Mac OS X v10.2 with Safari 1.0 installed.

Available in Mac OS X v10.2.7 and later.

See Also

+ [connectionWithRequest:delegate:](#) (page 8)

- [initWithRequest:delegate:startImmediately:](#) (page 11)

Declared In

NSURLConnection.h

initWithRequest:delegate:startImmediately:

Returns an initialized URL connection and begins to load the data for the URL request, if specified.

```
- (id)initWithRequest:(NSURLRequest *)request delegate:(id)delegate
startImmediately:(BOOL)startImmediately
```

Parameters*request*

The URL request to load. The *request* object is deep-copied as part of the initialization process. Changes made to *request* after this method returns do not affect the request that is used for the loading process.

delegate

The delegate object for the connection. The delegate will receive delegate messages as the load progresses. Messages to the delegate will be sent on the thread that calls this method. By default, for the connection to work correctly the calling thread's run loop must be operating in the default run loop mode. See [scheduleInRunLoop:forMode:](#) (page 12) to change the runloop and mode.]

startImmediately

YES if the connection should be loading data immediately, otherwise NO.

Return Value

The URL connection for the URL request. Returns `nil` if a connection can't be initialized.

Availability

Available in Mac OS X v10.5 and later.

Related Sample Code

URL CacheInfo

Declared In

NSURLConnection.h

scheduleInRunLoop:forMode:

Determines the runloop and mode that the receiver uses to send delegate messages to the receiver.

```
- (void)scheduleInRunLoop:(NSRunLoop *)aRunLoop forMode:(NSString *)mode
```

Parameters

aRunLoop

The NSRunLoop instance to use for delegate messages.

mode

The mode in which to supply delegate messages.

Discussion

At creation, a connection is scheduled on the current thread (the one where the creation takes place) in the default mode. That can be changed to add or remove runloop + mode pairs using the following methods. It is permissible to be scheduled on multiple run loops and modes, or on the same run loop in multiple modes, so scheduling in one place does not cause unscheduling in another.

You may call these methods after the connection has started. However, if the connection is scheduled on multiple threads or if you are not calling these methods from the thread where the connection is scheduled, there is a race between these methods and the delivery of delegate methods on the other threads. The caller must either be prepared for additional delegation messages on the other threads, or must halt the run loops on the other threads before calling these methods to guarantee that no further callbacks will occur.

Availability

Available in Mac OS X v10.5 and later.

Declared In

NSURLConnection.h

start

Causes the receiver to begin loading data, if it has not already.

```
- (void)start
```

Availability

Available in Mac OS X v10.5 and later.

Declared In

NSURLConnection.h

unsubscribeFromRunLoop:forMode:

Causes the receiver to stop sending delegate messages using the specified runloop and mode.

```
- (void)unsubscribeFromRunLoop:(NSRunLoop *)aRunLoop forMode:(NSString *)mode
```

Parameters*aRunLoop*

The runloop instance to unschedule.

mode

The mode to unschedule.

Discussion

At creation, a connection is scheduled on the current thread (the one where the creation takes place) in the default mode. That can be changed to add or remove runloop + mode pairs using the following methods. It is permissible to be scheduled on multiple run loops and modes, or on the same run loop in multiple modes, so scheduling in one place does not cause unscheduling in another.

You may call these methods after the connection has started. However, if the connection is scheduled on multiple threads or if you are not calling these methods from the thread where the connection is scheduled, there is a race between these methods and the delivery of delegate methods on the other threads. The caller must either be prepared for additional delegation messages on the other threads, or must halt the run loops on the other threads before calling these methods to guarantee that no further callbacks will occur.

Availability

Available in Mac OS X v10.5 and later.

Declared In

NSURLConnection.h

Delegate Methods

connection:canAuthenticateAgainstProtectionSpace:

Sent to determine whether the delegate is able to respond to a protection space's form of authentication.

```
- (BOOL)connection:(NSURLConnection *)connection
    canAuthenticateAgainstProtectionSpace:(NSURLProtectionSpace *)protectionSpace
```

Parameters*connection*

The connection sending the message.

protectionSpace

The protection space that generates an authentication challenge.

Discussion

This method is called before [connection:didReceiveAuthenticationChallenge:](#) (page 14), allowing the delegate to inspect a protection space before attempting to authenticate against it. By returning YES, the delegate indicates that it can handle the form of authentication, which it does in the subsequent call to [connection:didReceiveAuthenticationChallenge:](#) (page 14). Not implementing this method is the same as returning NO, in which case default authentication handling is used.

Availability

Available in Mac OS X v10.6 and later.

Declared In

NSURLConnection.h

connection:didCancelAuthenticationChallenge:

Sent when a connection cancels an authentication challenge.

```
- (void)connection:(NSURLConnection *)connection
    didCancelAuthenticationChallenge:(NSURLAuthenticationChallenge *)challenge
```

Parameters

connection

The connection sending the message.

challenge

The challenge that was canceled.

Availability

Available in Mac OS X v10.2 with Safari 1.0 installed.

Available in Mac OS X v10.2.7 and later.

Declared In

NSURLConnection.h

connection:didFailWithError:

Sent when a connection fails to load its request successfully.

```
- (void)connection:(NSURLConnection *)connection didFailWithError:(NSError *)error
```

Parameters

connection

The connection sending the message.

error

An error object containing details of why the connection failed to load the request successfully.

Discussion

Once the delegate receives this message, it will receive no further messages for *connection*.

Availability

Available in Mac OS X v10.2 with Safari 1.0 installed.

Available in Mac OS X v10.2.7 and later.

Declared In

NSURLConnection.h

connection:didReceiveAuthenticationChallenge:

Sent when a connection must authenticate a challenge in order to download its request.

```
- (void)connection:(NSURLConnection *)connection
    didReceiveAuthenticationChallenge:(NSURLAuthenticationChallenge *)challenge
```

Parameters

connection

The connection sending the message.

challenge

The challenge that *connection* must authenticate in order to download its request.

Discussion

This method gives the delegate the opportunity to determine the course of action taken for the challenge: provide credentials, continue without providing credentials, or cancel the authentication challenge and the download.

The delegate can determine the number of previous authentication challenges by sending the message `previousFailureCount` to *challenge*.

If the previous failure count is 0 and the value returned by `proposedCredential` is `nil`, the delegate can create a new `NSURLCredential` object, providing information specific to the type of credential, and send a `useCredential:forAuthenticationChallenge:` message to `[challenge sender]`, passing the credential and *challenge* as parameters. If `proposedCredential` is not `nil`, the value is a credential from the URL or the shared credential storage that can be provided to the user as feedback.

The delegate may decide to abandon further attempts at authentication at any time by sending `[challenge sender] a continueWithoutCredentialForAuthenticationChallenge:` or a `cancelAuthenticationChallenge:` message. The specific action is implementation dependent.

If the delegate implements this method, the download will suspend until `[challenge sender]` is sent one of the following messages: `useCredential:forAuthenticationChallenge:`, `continueWithoutCredentialForAuthenticationChallenge:` or `cancelAuthenticationChallenge:`.

If the delegate does not implement this method the default implementation is used. If a valid credential for the request is provided as part of the URL, or is available from the `NSURLCredentialStorage` the `[challenge sender]` is sent a `useCredential:forAuthenticationChallenge:` with the credential. If the challenge has no credential or the credentials fail to authorize access, then `continueWithoutCredentialForAuthenticationChallenge:` is sent to `[challenge sender]` instead.

Availability

Available in Mac OS X v10.2 with Safari 1.0 installed.

Available in Mac OS X v10.2.7 and later.

See Also

- `cancelAuthenticationChallenge:`
- `continueWithoutCredentialForAuthenticationChallenge:`
- `useCredential:forAuthenticationChallenge:`

Declared In

`NSURLConnection.h`

connection:didReceiveData:

Sent as a connection loads data incrementally.

```
- (void)connection:(NSURLConnection *)connection didReceiveData:(NSData *)data
```

Parameters

connection

The connection sending the message.

data

The newly available data. The delegate should concatenate the contents of each *data* object delivered to build up the complete data for a URL load.

Discussion

This method provides the only way for an asynchronous delegate to retrieve the loaded data. It is the responsibility of the delegate to retain or copy this data as it is delivered.

Availability

Available in Mac OS X v10.2 with Safari 1.0 installed.

Available in Mac OS X v10.2.7 and later.

Declared In

NSURLConnection.h

connection:didReceiveResponse:

Sent when the connection has received sufficient data to construct the URL response for its request.

```
- (void)connection:(NSURLConnection *)connection didReceiveResponse:(NSURLResponse *)response
```

Parameters

connection

The connection sending the message.

response

The URL response for the connection's request. This object is immutable and will not be modified by the URL loading system once it is presented to the delegate.

Discussion

In rare cases, for example in the case of an HTTP load where the content type of the load data is `multipart/x-mixed-replace`, the delegate will receive more than one `connection:didReceiveResponse:` message. In the event this occurs, delegates should discard all data previously delivered by `connection:didReceiveData:`, and should be prepared to handle the, potentially different, MIME type reported by the newly reported URL response.

The only case where this message is not sent to the delegate is when the protocol implementation encounters an error before a response could be created.

Availability

Available in Mac OS X v10.2 with Safari 1.0 installed.

Available in Mac OS X v10.2.7 and later.

Declared In

NSURLConnection.h

connection:didSendBodyData:totalBytesWritten:totalBytesExpectedToWrite:

Sent as the body (message data) of a request is transmitted (such as in an http POST request).

```
- (void)connection:(NSURLConnection *)connection
    didSendBodyData:(NSInteger)bytesWritten
    totalBytesWritten:(NSInteger)totalBytesWritten
    totalBytesExpectedToWrite:(NSInteger)totalBytesExpectedToWrite
```

Parameters*connection*

The connection sending the message.

bytesWritten

The number of bytes written in the latest write.

totalBytesWritten

The total number of bytes written for this connection.

totalBytesExpectedToWrite

The number of bytes the connection expects to write.

Discussion

This method provides an estimate of the progress of a URL upload.

The value of `totalBytesExpectedToWrite` may change during the upload if the request needs to be retransmitted due to a lost connection or an authentication challenge from the server.

Availability

Available in Mac OS X v10.6 and later.

Declared In

NSURLConnection.h

connection:willCacheResponse:

Sent before the connection stores a cached response in the cache, to give the delegate an opportunity to alter it.

```
- (NSCachedURLResponse *)connection:(NSURLConnection *)connection
    willCacheResponse:(NSCachedURLResponse *)cachedResponse
```

Parameters*connection*

The connection sending the message.

cachedResponse

The proposed cached response to store in the cache.

Return Value

The actual cached response to store in the cache. The delegate may return `cachedResponse` unmodified, return a modified cached response, or return `nil` if no cached response should be stored for the connection.

Availability

Available in Mac OS X v10.2 with Safari 1.0 installed.

Available in Mac OS X v10.2.7 and later.

Declared In

NSURLConnection.h

connection:willSendRequest:redirectResponse:

Sent when the connection determines that it must change URLs in order to continue loading a request.

```
- (NSURLRequest *)connection:(NSURLConnection *)connection
  willSendRequest:(NSURLRequest *)request redirectResponse:(NSURLResponse
 *)redirectResponse
```

Parameters

connection

The connection sending the message.

request

The proposed redirected request. The delegate should inspect the redirected request to verify that it meets its needs, and create a copy with new attributes to return to the connection if necessary.

redirectResponse

The URL response that caused the redirect. May be *nil* in cases where this method is not being sent as a result of involving the delegate in redirect processing.

Return Value

The actual URL request to use in light of the redirection response. The delegate may copy and modify *request* as necessary to change its attributes, return *request* unmodified, or return *nil*.

Discussion

If the delegate wishes to cancel the redirect, it should call the *connection* object's `cancel` method. Alternatively, the delegate method can return *nil* to cancel the redirect, and the connection will continue to process. This has special relevance in the case where *redirectResponse* is not *nil*. In this case, any data that is loaded for the connection will be sent to the delegate, and the delegate will receive a `connectionDidFinishLoading` or `connection:didFailLoadingWithError:` message, as appropriate.

Special Considerations

The delegate can receive this message as a result of transforming a request's URL to its canonical form, or for protocol-specific reasons, such as an HTTP redirect. The delegate implementation should be prepared to receive this message multiple times.

Availability

Available in Mac OS X v10.2 with Safari 1.0 installed.

Available in Mac OS X v10.2.7 and later.

Declared In

`NSURLConnection.h`

connectionDidFinishLoading:

Sent when a connection has finished loading successfully.

```
- (void)connectionDidFinishLoading:(NSURLConnection *)connection
```

Parameters

connection

The connection sending the message.

Discussion

The delegate will receive no further messages for *connection*.

Availability

Available in Mac OS X v10.2 with Safari 1.0 installed.

Available in Mac OS X v10.2.7 and later.

Declared In

NSURLConnection.h

connectionShouldUseCredentialStorage:

Sent to determine whether the URL loader should consult the credential storage for authenticating the connection.

- (BOOL)connectionShouldUseCredentialStorage:(NSURLConnection *)*connection*

Parameters

connection

The connection sending the message.

Discussion

This method is called before any attempt to authenticate is made. By returning `NO`, the delegate tells the connection not to consult the credential storage and makes itself responsible for providing credentials for any authentication challenges. Not implementing this method is the same as returning `YES`. The delegate is free to consult the credential storage itself when it receives a [connection:didReceiveAuthenticationChallenge:](#) (page 14) message.

Availability

Available in Mac OS X v10.6 and later.

Declared In

NSURLConnection.h

Document Revision History

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

Date	Notes
2009-08-17	Added API information for credential storage, protection space authentication, and upload progress estimation.
2007-04-01	Updated for Mac OS X v10.5.
2006-05-23	First publication of this content as a separate document.

REVISION HISTORY

Document Revision History

Index

C

`cancel` **instance method** 10
`canHandleRequest:` **class method** 8
`connection:canAuthenticateAgainstProtectionSpace:`
 <NSObject> **delegate method** 13
`connection:didCancelAuthenticationChallenge:`
 <NSObject> **delegate method** 14
`connection:didFailWithError:` <NSObject> **delegate**
 method 14
`connection:didReceiveAuthenticationChallenge:`
 <NSObject> **delegate method** 14
`connection:didReceiveData:` <NSObject> **delegate**
 method 15
`connection:didReceiveResponse:` <NSObject>
 delegate method 16
`connection:didSendBodyData:totalBytesWritten:`
 totalBytesExpectedToWrite: <NSObject>
 delegate method 16
`connection:willCacheResponse:` <NSObject>
 delegate method 17
`connection:willSendRequest:redirectResponse:`
 <NSObject> **delegate method** 18
`connectionDidFinishLoading:` <NSObject> **delegate**
 method 18
`connectionShouldUseCredentialStorage:`
 <NSObject> **delegate method** 19
`connectionWithRequest:delegate:` **class method** 8

I

`initWithRequest:delegate:` **instance method** 10
`initWithRequest:delegate:startImmediately:`
 instance method 11

S

`scheduleInRunLoop:forMode:` **instance method** 12

`sendSynchronousRequest:returningResponse:error:`
 class method 9
`start` **instance method** 12

U

`unscheduleFromRunLoop:forMode:` **instance method**
 12