Class

URLSession

An object that coordinates a group of related, network data transfer tasks.

iOS 7.0+ | iPadOS 7.0+ | Mac Catalyst 13.1+ | macOS 10.9+ | tvOS 9.0+ | visionOS 1.0+ | watchOS 2.0+

```
class URLSession
```

Mentioned in

- Pausing and resuming uploads
- Analyzing HTTP traffic with Instruments
- Processing URL session data task results with Combine
- Downloading files from websites
- Downloading files in the background

Overview

The `URLSession` class and related classes provide an API for downloading data from and uploading data to endpoints indicated by URLs. Your app can also use this API to perform background downloads when your app isn't running or, in iOS, while your app is suspended. You can use the related `URLSessionDelegate` and `URLSessionTaskDelegate` to support authentication and receive events like redirection and task completion.

Note

The `URLSession` API involves many different classes that work together in a fairly complex way which may not be obvious if you read the reference documentation by itself. Before using the API, read the overview in the URL Loading System topic. The articles in the Essentials, Uploading, and Downloading sections offer examples of performing common tasks with `URLSession`.

Your app creates one or more `URLSession` instances, each of which coordinates a group of related data-transfer tasks. For example, if you're creating a web browser, your app might create one session per tab or window, or one session for interactive use and another for background downloads. Within each session, your app adds a series of tasks, each of which represents a request for a specific URL (following HTTP redirects, if necessary).

Types of URL sessions

The tasks within a given URL session share a common session configuration object, which defines connection behavior, like the maximum number of simultaneous connections to make to a single host, whether connections can use the cellular network, and so on.

`URLSession` has a singleton `shared` session (which doesn't have a configuration object) for basic requests. It's not as customizable as sessions you create, but it serves as a good starting point if you have very limited requirements. You access this session by calling the shared class method. For other kinds of sessions, you create a `URLSession` with one of three kinds of configurations:

- A default session behaves much like the shared session, but lets you configure it. You can also assign a delegate to the default session to obtain data incrementally.
- Ephemeral sessions are similar to shared sessions, but don't write caches, cookies, or credentials to disk.
- Background sessions let you perform uploads and downloads of content in the background while your app isn't running.

See Creating a session configuration object in the `URLSessionConfiguration` class for details on creating each type of configuration.

Types of URL session tasks

Within a session, you create tasks that optionally upload data to a server and then retrieve data from the server either as a file on disk or as one or more `NSData` objects in memory. The `URLSession` API provides four types of tasks:

- Data tasks send and receive data using `NSData` objects. Data tasks are intended for short, often interactive requests to a server.
- Upload tasks are similar to data tasks, but they also send data (often in the form of a file), and support background uploads while the app isn’t running.
- Download tasks retrieve data in the form of a file, and support background downloads and uploads while the app isn’t running.
- WebSocket tasks exchange messages over TCP and TLS, using the WebSocket protocol defined in RFC 6455.

Using a session delegate

Tasks in a session also share a common delegate object. You implement this delegate to provide and obtain information when various events occur, including when:

- Authentication fails.
- Data arrives from the server.
- Data becomes available for caching.

If you don’t need the features provided by a delegate, you can use this API without providing one by passing `nil` when you create a session.

Important

The session object keeps a strong reference to the delegate until your app exits or explicitly invalidates the session. If you don’t invalidate the session, your app leaks memory until the app terminates.

Each task you create with the session calls back to the session’s delegate, using the methods defined in `URLSessionTaskDelegate`. You can also intercept these callbacks before they reach the session delegate by populating a separate `delegate` that’s specific to the task.

Asynchronicity and URL sessions

Like most networking APIs, the `URLSession` API is highly asynchronous. It returns data to your app in one of three ways, depending on the methods you call:

- If you’re using Swift, you can use the methods marked with the `async` keyword to perform common tasks. For example, `data(from:delegate:)` fetches data, while `download(from:delegate:)` downloads files. Your call point uses the `await` keyword to suspend running until the transfer completes. You can also use the `bytes(from:delegate:)` method to receive data as an `AsyncSequence`. With this approach, you use the `for-await-in` syntax to iterate over the data as your app receives it. The `URL` type also offers covenience methods to fetch bytes or lines from the shared URL session.

- In Swift or Objective-C, you can provide a completion handler block, which runs when the transfer completes.
- In Swift or Objective-C, you can receive callbacks to a delegate method as the transfer progresses and immediately after it completes.

In addition to delivering this information to delegates, the `URLSession` provides status and progress properties. Query these properties if you need to make programmatic decisions based on the current state of the task (with the caveat that its state can change at any time).

Protocol support

The `URLSession` class natively supports the `data`, `file`, `ftp`, `http`, and `https` URL schemes, with transparent support for proxy servers and SOCKS gateways, as configured in the user's system preferences.

`URLSession` supports the HTTP/1.1, HTTP/2, and HTTP/3 protocols. HTTP/2 support, as described by RFC 7540, requires a server that supports Application-Layer Protocol Negotiation (ALPN).

You can also add support for your own custom networking protocols and URL schemes (for your app's private use) by subclassing `URLProtocol`.

App Transport Security (ATS)

iOS 9.0 and macOS 10.11 and later use App Transport Security (ATS) for all HTTP connections made with `URLSession`. ATS requires that HTTP connections use HTTPS (RFC 2818).

For more information, see `NSAppTransportSecurity`.

Foundation copying behavior

Session and task objects conform to the `NSCopying` protocol as follows:

- When your app copies a session or task object, you get the same object back.
- When your app copies a configuration object, you get a new copy you can independently modify.

Thread safety

The URL session API is thread-safe. You can freely create sessions and tasks in any thread context. When your delegate methods call the provided completion handlers, the work is automatically scheduled on the correct delegate queue.

Topics

Using the shared session

`class var shared: URLSession`

The shared singleton session object.

Creating a session

`init(configuration: URLSessionConfiguration)`

Creates a session with the specified session configuration.

`init(configuration: URLSessionConfiguration, delegate: (any URLSessionDelegate)?, delegateQueue: OperationQueue?)`

Creates a session with the specified session configuration, delegate, and operation queue.

`class URLSessionConfiguration`

A configuration object that defines behavior and policies for a URL session.

`var configuration: URLSessionConfiguration`

A copy of the configuration object for this session.

Working with a delegate

`var delegate: (any URLSessionDelegate)?`

The delegate assigned when this object was created.

`protocol URLSessionDelegate`

A protocol that defines methods that URL session instances call on their delegates to handle session-level events, like session life cycle changes.

`protocol URLSessionTaskDelegate`

A protocol that defines methods that URL session instances call on their delegates to handle task-level events.

```
var delegateQueue: OperationQueue
```

The operation queue provided when this object was created.

Performing asynchronous transfers

```
func bytes(for: URLRequest, delegate: (any URLSessionTaskDelegate)?)
async throws -> (URLSession.AsyncBytes, URLResponse)
```

Retrieves the contents of a URL based on the specified URL request and delivers an asynchronous sequence of bytes.

```
func bytes(from: URL, delegate: (any URLSessionTaskDelegate)?) async
throws -> (URLSession.AsyncBytes, URLResponse)
```

Retrieves the contents of a given URL and delivers an asynchronous sequence of bytes.

```
struct AsyncBytes
```

An asynchronous sequence of bytes.

```
func data(for: URLRequest, delegate: (any URLSessionTaskDelegate)?)
async throws -> (Data, URLResponse)
```

Downloads the contents of a URL based on the specified URL request and delivers the data asynchronously.

```
func data(from: URL, delegate: (any URLSessionTaskDelegate)?) async
throws -> (Data, URLResponse)
```

Retrieves the contents of a URL and delivers the data asynchronously.

```
func data(for: URLRequest) async throws -> (Data, URLResponse)
```

```
func data(from: URL) async throws -> (Data, URLResponse)
```

```
func download(for: URLRequest, delegate: (any URLSessionTaskDelegate)?)
async throws -> (URL, URLResponse)
```

Retrieves the contents of a URL based on the specified URL request and delivers the URL of the saved file asynchronously.

```
func download(from: URL, delegate: (any URLSessionTaskDelegate)?) async
 throws -> (URL, URLResponse)
```

Retrieves the contents of a URL and delivers the URL of the saved file asynchronously.

```
func download(resumeFrom: Data, delegate: (any URLSessionTaskDelegate
)?) async throws -> (URL, URLResponse)
```

Resumes a previously-paused download and delivers the URL of the saved file asynchronously.

`func upload(for: URLRequest, from: Data, delegate: (any URLSessionTaskDelegate)?) async throws -> (Data, URLResponse)`

Uploads data to a URL based on the specified URL request and delivers the result asynchronously.

`func upload(for: URLRequest, fromFile: URL, delegate: (any URLSessionTaskDelegate)?) async throws -> (Data, URLResponse)`

Uploads data to a URL and delivers the result asynchronously.

`func upload(for: URLRequest, from: Data) async throws -> (Data, URLResponse)`

`func upload(for: URLRequest, fromFile: URL) async throws -> (Data, URLResponse)`

`protocol URLSessionTaskDelegate`

A protocol that defines methods that URL session instances call on their delegates to handle task-level events.

Adding data tasks to a session

`func dataTask(with: URL) -> URLSessionDataTask`

Creates a task that retrieves the contents of the specified URL.

`func dataTask(with: URL, completionHandler: (Data?, URLResponse?, (any Error)?) -> Void) -> URLSessionDataTask`

Creates a task that retrieves the contents of the specified URL, then calls a handler upon completion.

`func dataTask(with: URLRequest) -> URLSessionDataTask`

Creates a task that retrieves the contents of a URL based on the specified URL request object.

`func dataTask(with: URLRequest, completionHandler: (Data?, URLResponse?, (any Error)?) -> Void) -> URLSessionDataTask`

Creates a task that retrieves the contents of a URL based on the specified URL request object, and calls a handler upon completion.

`class URLSessionDataTask`

A URL session task that returns downloaded data directly to the app in memory.

`protocol URLSessionDataDelegate`

A protocol that defines methods that URL session instances call on their delegates to handle task-level events specific to data and upload tasks.

Adding download tasks to a session

```
func downloadTask(with: URL) -> URLSessionDownloadTask
```

Creates a download task that retrieves the contents of the specified URL and saves the results to a file.

```
func downloadTask(with: URL, completionHandler: (URL?, URLResponse?,
(any Error)?) -> Void) -> URLSessionDownloadTask
```

Creates a download task that retrieves the contents of the specified URL, saves the results to a file, and calls a handler upon completion.

```
func downloadTask(with: URLRequest) -> URLSessionDownloadTask
```

Creates a download task that retrieves the contents of a URL based on the specified URL request object and saves the results to a file.

```
func downloadTask(with: URLRequest, completionHandler: (URL?,
URLResponse?, (any Error)?) -> Void) -> URLSessionDownloadTask
```

Creates a download task that retrieves the contents of a URL based on the specified URL request object, saves the results to a file, and calls a handler upon completion.

```
func downloadTask(withResumeData: Data) -> URLSessionDownloadTask
```

Creates a download task to resume a previously canceled or failed download.

```
func downloadTask(withResumeData: Data, completionHandler: (URL?,
URLResponse?, (any Error)?) -> Void) -> URLSessionDownloadTask
```

Creates a download task to resume a previously canceled or failed download and calls a handler upon completion.

```
class URLSessionDownloadTask
```

A URL session task that stores downloaded data to a file.

```
protocol URLSessionDownloadDelegate
```

A protocol that defines methods that URL session instances call on their delegates to handle task-level events specific to download tasks.

Adding upload tasks to a session

{} Building a resumable upload server with SwiftNIO

Support HTTP resumable upload protocol in SwiftNIO by translating resumable uploads to regular uploads.

`func uploadTask(with: URLRequest, from: Data) -> URLSessionUploadTask`

Creates a task that performs an HTTP request for the specified URL request object and uploads the provided data.

`func uploadTask(with: URLRequest, from: Data?, completionHandler: (Data?, URLResponse?, (any Error)?) -> Void) -> URLSessionUploadTask`

Creates a task that performs an HTTP request for the specified URL request object, uploads the provided data, and calls a handler upon completion.

`func uploadTask(with: URLRequest, fromFile: URL) -> URLSessionUploadTask`

Creates a task that performs an HTTP request for uploading the specified file.

`func uploadTask(with: URLRequest, fromFile: URL, completionHandler: (Data?, URLResponse?, (any Error)?) -> Void) -> URLSessionUploadTask`

Creates a task that performs an HTTP request for uploading the specified file, then calls a handler upon completion.

`func uploadTask(withStreamedRequest: URLRequest) -> URLSessionUploadTask`

Creates a task that performs an HTTP request for uploading data based on the specified URL request.

`func uploadTask(withResumeData: Data) -> URLSessionUploadTask`

`func uploadTask(withResumeData: Data, completionHandler: (Data?, URLResponse?, (any Error)?) -> Void) -> URLSessionUploadTask`

`class URLSessionUploadTask`

A URL session task that uploads data to the network in a request body.

`protocol URLSessionDataDelegate`

A protocol that defines methods that URL session instances call on their delegates to handle task-level events specific to data and upload tasks.

Adding stream tasks to a session

`func streamTask(withHostName: String, port: Int) -> URLSessionStreamTask`

Creates a task that establishes a bidirectional TCP/IP connection to a specified hostname and port.

~~`func streamTask(with: NetService) -> URLSessionStreamTask`~~

Creates a task that establishes a bidirectional TCP/IP connection using a specified network service.

Deprecated

`class URLSessionStreamTask`

A URL session task that is stream-based.

`protocol URLSessionStreamDelegate`

A protocol that defines methods that URL session instances call on their delegates to handle task-level events specific to stream tasks.

Adding WebSocket tasks to a session

`func webSocketTask(with: URL) -> URLSessionWebSocketTask`

Creates a WebSocket task for the provided URL.

`func webSocketTask(with: URLRequest) -> URLSessionWebSocketTask`

Creates a WebSocket task for the provided URL request.

`func webSocketTask(with: URL, protocols: [String]) -> URLSessionWebSocketTask`

Creates a WebSocket task given a URL and an array of protocols.

`class URLSessionWebSocketTask`

A URL session task that communicates over the WebSockets protocol standard.

`protocol URLSessionWebSocketDelegate`

A protocol that defines methods that URL session instances call on their delegates to handle task-level events specific to WebSocket tasks.

Managing the session

`func finishTasksAndInvalidate()`

Invalidates the session, allowing any outstanding tasks to finish.

`func flush(completionHandler: () -> Void)`

Flushes cookies and credentials to disk, clears transient caches, and ensures that future requests occur on a new TCP connection.

```
func getTasksWithCompletionHandler(([URLSessionDataTask], [URLSession
UploadTask], [URLSessionDownloadTask]) -> Void)
```

Asynchronously calls a completion callback with all data, upload, and download tasks in a session.

```
func getAllTasks(completionHandler: ([URLSessionTask]) -> Void)
```

Asynchronously calls a completion callback with all tasks in a session

```
func invalidateAndCancel()
```

Cancels all outstanding tasks and then invalidates the session.

```
func reset(completionHandler: () -> Void)
```

Empties all cookies, caches and credential stores, removes disk files, flushes in-progress downloads to disk, and ensures that future requests occur on a new socket.

```
var sessionDescription: String?
```

An app-defined descriptive label for the session.

Handling errors

URL session error dictionary keys

Keys used in conjunction with error objects returned by URL sessions and tasks.

Background task cancellation

Constants that indicate why a background task was canceled.

Performing tasks as a Combine Publisher

Processing URL session data task results with Combine

Use a chain of asynchronous operators to receive and process data fetched from a URL.

```
func dataTaskPublisher(for: URLRequest) -> URLSession.DataTaskPublisher
```

Returns a publisher that wraps a URL session data task for a given URL request.

```
func dataTaskPublisher(for: URL) -> URLSession.DataTaskPublisher
```

Returns a publisher that wraps a URL session data task for a given URL.

```
struct DataTaskPublisher
```

A publisher that delivers the results of performing URL session data tasks.

Deprecated

`~~class func new() -> Self~~` Deprecated

`~~init()~~` Deprecated

Relationships

Inherits From

`NSObject`

Conforms To

`CVarArg`
`CustomDebugStringConvertible`
`CustomStringConvertible`
`Equatable`
`Hashable`
`NSObjectProtocol`
`Sendable`
`SendableMetatype`

See Also

Essentials

Fetching website data into memory

Receive data directly into memory by creating a data task from a URL session.

Analyzing HTTP traffic with Instruments

Measure HTTP-based network performance and usage of your apps.

`class URLSessionTask`

A task, like downloading a specific resource, performed in a URL session.