URLSessionConfiguration

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

SDKs
iOS 7.0+
macOS 10.9+
tvOS 9.0+
watchOS 2.0+

Framework
Foundation

Declaration

```
class URLSessionConfiguration : NSObject
```

Overview

An `URLSessionConfiguration` object defines the behavior and policies to use when uploading and downloading data using an `URLSession` object. When uploading or downloading data, creating a configuration object is always the first step you must take. You use this object to configure the timeout values, caching policies, connection requirements, and other types of information that you intend to use with your `URLSession` object.

It is important to configure your `URLSessionConfiguration` object appropriately before using it to initialize a session object. Session objects make a copy of the configuration settings you provide and use those settings to configure the session. Once configured, the session object ignores any changes you make to the `URLSessionConfiguration` object. If you need to modify your transfer policies, you must update the session configuration object and use it to create a new `URLSession` object.

> **Note**
>
> In some cases, the policies defined in this configuration may be overridden by policies specified by an `NSURLRequest` object provided for a task. Any policy specified on the request object is respected unless the session's policy is more restrictive. For example, if the session configuration specifies that cellular networking should not be allowed, the `NSURLRequest` object cannot request cellular networking.

For more information about using configuration objects to create sessions, see `URLSession`.

Types of Session Configurations

The behavior and capabilities of a URL session are largely determined by the kind of

configuration used to create the session.

The singleton shared session (which has no configuration object) is for basic requests. It's not as customizable as sessions that 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. See that method's discussion for more information about its limitations.

Default sessions behave much like the shared session (unless you customize them further), but let you obtain data incrementally using a delegate. You can create a default session configuration by calling the default method on the URLSessionConfiguration class.

Ephemeral sessions are similar to default sessions, but they don't write caches, cookies, or credentials to disk. You can create an ephemeral session configuration by calling the ephemeral method on the URLSessionConfiguration class.

Background sessions let you perform uploads and downloads of content in the background while your app isn't running. You can create a background session configuration by calling the backgroundSessionConfiguration(_:) method on the URLSessionConfiguration class.

Topics

Creating a Session Configuration Object

```
class var `default`: URLSessionConfiguration
```

A default session configuration object.

```
class var ephemeral: URLSessionConfiguration
```

A session configuration that uses no persistent storage for caches, cookies, or credentials.

```
class func background(withIdentifier: String) -> URLSession
Configuration
```

Creates a session configuration object that allows HTTP and HTTPS uploads or downloads to be performed in the background.

Setting General Properties

`var identifier: String?`

The background session identifier of the configuration object.

`var httpAdditionalHeaders: [AnyHashable : Any]?`

A dictionary of additional headers to send with requests.

`var networkServiceType: NSURLRequest.NetworkServiceType`

The type of network service for all tasks within sessions based on this configuration.

`var allowsCellularAccess: Bool`

A Boolean value that determines whether connections should be made over a cellular network.

`var timeoutIntervalForRequest: TimeInterval`

The timeout interval to use when waiting for additional data.

`var timeoutIntervalForResource: TimeInterval`

The maximum amount of time that a resource request should be allowed to take.

`var sharedContainerIdentifier: String?`

The identifier for the shared container into which files in background URL sessions should be downloaded.

`var waitsForConnectivity: Bool`

A Boolean value that indicates whether the session should wait for connectivity to become available, or fail immediately.

Setting Cookie Policies

`var httpCookieAcceptPolicy: HTTPCookie.AcceptPolicy`

A policy constant that determines when cookies should be accepted.

`var httpShouldSetCookies: Bool`

A Boolean value that determines whether requests should contain cookies from the cookie store.

`var httpCookieStorage: HTTPCookieStorage?`

The cookie store for storing cookies within this session.

`class HTTPCookieStorage`

A container that manages the storage of cookies.

`class HTTPCookie`

A representation of an HTTP cookie.

Setting Security Policies

`var tlsMaximumSupportedProtocol: SSLProtocol`

The maximum TLS protocol version that the client should request when making connections in this session.

`var tlsMinimumSupportedProtocol: SSLProtocol`

The minimum TLS protocol that should be accepted during protocol negotiation.

`var urlCredentialStorage: URLCredentialStorage?`

A credential store that provides credentials for authentication.

Setting Caching Policies

`var urlCache: URLCache?`

The URL cache for providing cached responses to requests within the session.

`var requestCachePolicy: NSURLRequest.CachePolicy`

A predefined constant that determines when to return a response from the cache.

Supporting Background Transfers

`var sessionSendsLaunchEvents: Bool`

A Boolean value that indicates whether the app should be resumed or launched in the background when transfers finish.

`var isDiscretionary: Bool`

A Boolean value that determines whether background tasks can be scheduled at the discretion of the system for optimal performance.

`var shouldUseExtendedBackgroundIdleMode: Bool`

A Boolean value that indicates whether TCP connections should be kept open when the app moves to the background.

Supporting Custom Protocols

`var protocolClasses: [AnyClass]?`

An array of extra protocol subclasses that handle requests in a session.

`class URLProtocol`

An abstract class that handles the loading of protocol-specific URL data.

Supporting Multipath TCP

Improving Network Reliability Using Multipath TCP

Use the available radios in iOS devices to improve your app's network reliability and performance.

`var multipathServiceType: URLSessionConfiguration.MultipathServiceType`

A service type that specifies the Multipath TCP connection policy for transmitting data over Wi-Fi and cellular interfaces.

`enum URLSessionConfiguration.MultipathServiceType`

Constants that specify the type of service that Multipath TCP uses.

Setting HTTP Policy and Proxy Properties

`var httpMaximumConnectionsPerHost: Int`

The maximum number of simultaneous connections to make to a given host.

`var httpShouldUsePipelining: Bool`

A Boolean value that determines whether the session should use HTTP pipelining.

`var connectionProxyDictionary: [AnyHashable : Any]?`

A dictionary containing information about the proxy to use within this session.

Supporting Connectivity Changes

`var waitsForConnectivity: Bool`

A Boolean value that indicates whether the session should wait for connectivity to become available, or fail immediately.

Deprecated Methods

~~`class func backgroundSessionConfiguration(String) -> URLSessionConfiguration`~~

Returns a session configuration object that allows HTTP and HTTPS uploads or downloads to be performed in the background.

Deprecated

Entitlements

Multipath Entitlement

A Boolean value that indicates whether the app may use Multipath protocols like Multipath TCP to smoothly hand over traffic from one interface to another.
Key: com.apple.developer.networking.multipath

Relationships

Inherits From

`NSObject`

Conforms To

`CVarArg`
`Equatable`
`Hashable`
`NSCopying`

See Also

Creating a Session

`init(configuration: URLSessionConfiguration)`

Creates a session with the specified session configuration.

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

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

`var configuration: URLSessionConfiguration`

A copy of the configuration object for this session.