Class

UserDefaults

An interface to the user's defaults database, where you store key-value pairs persistently across launches of your app.

iOS 2.0+ | iPadOS 2.0+ | Mac Catalyst 13.0+ | macOS 10.0+ | tvOS 9.0+ | visionOS 1.0+ | watchOS 2.0+

```
class UserDefaults
```

Mentioned in

Using the file system effectively

Overview

The `UserDefaults` class provides a programmatic interface for interacting with the defaults system. The defaults system allows an app to customize its behavior to match a user's preferences. For example, you can allow users to specify their preferred units of measurement or media playback speed. Apps store these preferences by assigning values to a set of parameters in a user's defaults database. The parameters are referred to as *defaults* because they're commonly used to determine an app's default state at startup or the way it acts by default.

At runtime, you use `UserDefaults` objects to read the defaults that your app uses from a user's defaults database. `UserDefaults` caches the information to avoid having to open the user's defaults database each time you need a default value. When you set a default value, it's changed synchronously within your process, and asynchronously to persistent storage and other processes.

Important

Don't try to access the preferences subsystem directly. Modifying preference property list files may result in loss of changes, delay of reflecting changes, and app crashes. To configure preferences, use the `defaults` command-line utility in macOS instead.

With the exception of managed devices in educational institutions, a user's defaults are stored locally on a single device, and persisted for backup and restore. To synchronize preferences and other data across a user's connected devices, use `NSUbiquitousKeyValueStore` instead.

Important

This API has the potential of being misused to access device signals to try to identify the device or user, also known as fingerprinting. Regardless of whether a user gives your app permission to track, fingerprinting is not allowed. When you use this API in your app or third-party SDK (an SDK not provided by Apple), declare your usage and the reason for using the API in your app or third-party SDK's `PrivacyInfo.xcprivacy` file. For more information, including the list of valid reasons for using the API, see Describing use of required reason API.

Storing Default Objects

The `UserDefaults` class provides convenience methods for accessing common types such as floats, doubles, integers, Boolean values, and URLs. These methods are described in Setting Default Values.

A default object must be a property list—that is, an instance of (or for collections, a combination of instances of) `NSData`, `NSString`, `NSNumber`, `NSDate`, `NSArray`, or `NSDictionary`. If you want to store any other type of object, you should typically archive it to create an instance of NSData.

Values returned from `UserDefaults` are immutable, even if you set a mutable object as the value. For example, if you set a mutable string as the value for "MyStringDefault", the string you later retrieve using the `string(forKey:)` method will be immutable. If you set a mutable string as a default value and later mutate the string, the default value won't reflect the mutated string value unless you call `set(_:forKey:)` again.

For more details, see Preferences and Settings Programming Guide.

Persisting File References

A file URL specifies a location in the file system. If you use the `set(_:forKey:)` method to store the location for a particular file and the user moves that file, your app may not be able to locate that file on next launch. To store a reference to a file by its file system identity, you can instead create `NSURL` bookmark data using the `bookmarkData(options:includingResourceValuesForKeys:relativeTo:)` method and persist it using the `set(_:forKey:)` method. You can then use the `URLByResolvingBookmarkData:options:relativeToURL:bookmarkDataIsStale:error:` method to resolve the bookmark data stored in user defaults to a file URL.

Responding to Defaults Changes

You can use key-value observing to be notified of any updates to a particular default value. You can also register as an observer for `didChangeNotification` on the `default` notification center in order to be notified of all updates to a local defaults database.

For more details, see Key-Value Observing Programming Guide and Notification Programming Topics.

Using Defaults in Managed Environments

If your app supports managed environments, you can use `UserDefaults` to determine which preferences are managed by an administrator for the benefit of the user. In a managed environment, such as a computer lab or classroom, an administrator or teacher can configure the systems by establishing a set of default preferences for users. If a preference is managed in this manner (as determined by the methods described in Accessing Managed Environment Keys), your app should prevent users from editing that preference by disabling or hiding controls.

For more details, see Mobile Device Management Protocol Reference.

An app running on a device managed by an educational institution can use the iCloud key-value store to share small amounts of data with other instances of itself on the user's other devices. For example, a textbook app might store the current page number being read by the user so that other instances of the app can open to the same page when launched.

For more information, see Storing Preferences in iCloud in Preferences and Settings Programming Guide.

Sandbox Considerations

A sandboxed app cannot access or modify the preferences for any other app, with the following exceptions:

- App extensions on macOS and iOS

- Other apps in your application group on macOS

Adding a third-party app's domain using the `addSuite(named:)` method doesn't allow your app to access to that app's preferences. Attempting to access or modify another app's preferences doesn't result in an error; instead, macOS reads and writes files located within your app's container, rather than the actual preference files for the other application.

Thread Safety

The UserDefaults class is thread-safe.

Topics

Getting the Standard User Defaults Object

```
class var standard: UserDefaults
```

Returns the shared defaults object.

Creating User Defaults Objects

```
convenience init()
```

Creates a user defaults object initialized with the defaults for the app and current user.

```
init?(suiteName: String?)
```

Creates a user defaults object initialized with the defaults for the specified database name.

Getting Default Values

```
func object(forKey: String) -> Any?
```

Returns the object associated with the specified key.

```
func url(forKey: String) -> URL?
```

Returns the URL associated with the specified key.

```
func array(forKey: String) -> [Any]?
```

Returns the array associated with the specified key.

```
func dictionary(forKey: String) -> [String : Any]?
```

Returns the dictionary object associated with the specified key.

```
func string(forKey: String) -> String?
```

Returns the string associated with the specified key.

```
func stringArray(forKey: String) -> [String]?
```

Returns the array of strings associated with the specified key.

```
func data(forKey: String) -> Data?
```

Returns the data object associated with the specified key.

```
func bool(forKey: String) -> Bool
```

Returns the Boolean value associated with the specified key.

```
func integer(forKey: String) -> Int
```

Returns the integer value associated with the specified key.

```
func float(forKey: String) -> Float
```

Returns the float value associated with the specified key.

```
func double(forKey: String) -> Double
```

Returns the double value associated with the specified key.

```
func dictionaryRepresentation() -> [String : Any]
```

Returns a dictionary that contains a union of all key-value pairs in the domains in the search list.

Setting Default Values

```
func set(Any?, forKey: String)
```

Sets the value of the specified default key.

```
func set(Float, forKey: String)
```

Sets the value of the specified default key to the specified float value.

```
func set(Double, forKey: String)
```

Sets the value of the specified default key to the double value.

```
func set(Int, forKey: String)
```

Sets the value of the specified default key to the specified integer value.

```
func set(Bool, forKey: String)
```

Sets the value of the specified default key to the specified Boolean value.

```
func set(URL?, forKey: String)
```

Sets the value of the specified default key to the specified URL.

Removing Defaults

```
func removeObject(forKey: String)
```

Removes the value of the specified default key.

Maintaining Suites

```
func addSuite(named: String)
```

Inserts the specified domain name into the receiver's search list.

```
func removeSuite(named: String)
```

Removes the specified domain name from the receiver's search list.

Registering Defaults

```
func register(defaults: [String : Any])
```

Adds the contents of the specified dictionary to the registration domain.

Maintaining Persistent Domains

```
func persistentDomain(forName: String) -> [String : Any]?
```

Returns a dictionary representation of the defaults for the specified domain.

```
func setPersistentDomain([String : Any], forName: String)
```

Sets a dictionary for the specified persistent domain.

```
func removePersistentDomain(forName: String)
```

Removes the contents of the specified persistent domain from the user's defaults.

~~`func persistentDomainNames() -> [Any]`~~

Returns an array of the current persistent domain names.

Deprecated

Maintaining Volatile Domains

```
var volatileDomainNames: [String]
```

The current volatile domain names.

`func volatileDomain(forName: String) -> [String : Any]`

Returns the dictionary for the specified volatile domain.

`func setVolatileDomain([String : Any], forName: String)`

Sets the dictionary for the specified volatile domain.

`func removeVolatileDomain(forName: String)`

Removes the specified volatile domain from the user's defaults.

Accessing Managed Environment Keys

`func objectIsForced(forKey: String) -> Bool`

Returns a Boolean value indicating whether the specified key is managed by an administrator.

`func objectIsForced(forKey: String, inDomain: String) -> Bool`

Returns a Boolean value indicating whether the key in the specified domain is managed by an administrator.

Domains

`class let argumentDomain: String`

The domain consisting of defaults parsed from the application's arguments. These are one or more pairs of the form *-default value* included in the command-line invocation of the application.

`class let globalDomain: String`

The domain consisting of defaults meant to be seen by all applications.

`class let registrationDomain: String`

The domain consisting of a set of temporary defaults whose values can be set by the application to ensure that searches will always be successful.

Notifications

`class let didChangeNotification: NSNotification.Name`

Posted when user defaults are changed within the current process.

`class let sizeLimitExceededNotification: NSNotification.Name`

Posted when more data is stored in user defaults than is allowed.

~~class let completedInitialCloudSyncNotification: NSNotification.Name~~

Posted when ubiquitous defaults finish downloading data, either the first time a device is connected to an iCloud account or when a user switches their primary iCloud account.

Deprecated

~~class let didChangeCloudAccountsNotification: NSNotification.Name~~

Posted when the user changes the primary iCloud account.

Deprecated

~~class let noCloudAccountNotification: NSNotification.Name~~

Posted when a cloud default is set, but no iCloud user is logged in.

Deprecated

Legacy

~~convenience init?(user: String)~~

Creates a user defaults object initialized with the defaults for the specified user account.

Deprecated

`func synchronize() -> Bool`

Waits for any pending asynchronous updates to the defaults database and returns; this method is unnecessary and shouldn't be used.

`class func resetStandardUserDefaults()`

This method has no effect and shouldn't be used.

Language-Dependent Information Constants

These constants are deprecated and shouldn't be used.

Structures

`struct DidChangeMessage`

`struct SizeLimitExceededMessage`

Relationships

Inherits From

NSObject

Conforms To

CVarArg
CustomDebugStringConvertible
CustomStringConvertible
Equatable
Hashable
NSObjectProtocol