UIPasteboard | Apple Developer Documentation

Archived: 2026-04-06 02:09:07 UTC

Overview

For sharing data with any other app, use the systemwide general pasteboard. For sharing data with another app from your team — that has the same team ID as the app to share from — configure an App Group. For more information about configuring an App Group, see Configuring app groups.

In typical usage, an object in your app writes data to a pasteboard when the user requests a copy, cut, or duplicate operation on a selection in the user interface. Another object in the same or different app then reads that data from the pasteboard and presents it to the user at a new location. This usually happens when the user requests a paste operation.

The general pasteboard and named pasteboards

The system identifies the systemwide general pasteboard with the `general` pasteboard name, and you can use it for any type of data. Obtain the general pasteboard from the `general` shared system pasteboard object.

You can create named pasteboards with the class methods `init(name:create:)` and `withUniqueName()` for sharing data within your app, and from your app to other apps that have the same Team ID.

Using pasteboards

The `UIPasteboard` class provides methods for reading and writing individual pasteboard items, as well as methods for reading and writing multiple pasteboard items at once. For more information, see Getting and setting pasteboard items in the topic groups below. The data to write to a pasteboard can be in one of the following forms:

- If the data is an object that conforms to `NSItemProviderWriting`, use `setItemProviders(_:localOnly:expirationDate:)` to write it to the pasteboard.
- If you can represent the data with a common object — such as `NSString`, `NSArray`, `NSDictionary`, `NSDate`, `NSNumber`, `UIImage`, or `NSURL` — you can write it to the pasteboard as a value using a method such as `setValue(_:forPasteboardType:)`.
- If the data is binary, use the `setData(_:forPasteboardType:)` method to write it to the pasteboard.

The `UIPasteboard` class provides convenience methods for writing and reading strings, images, URLs, and colors to and from single or multiple pasteboard items. See Getting and setting pasteboard items of standard data types in the topic groups below.

`UIPasteboard` provides properties for directly checking whether specific data types are present on a pasteboard, see Checking for data types on a pasteboard in the topic groups below. Use these properties, rather than attempting

to read pasteboard data, to avoid causing the system to needlessly attempt to fetch data before necessary, or when the data might not be present. For example, use the `hasStrings` property to determine whether to present a string-data paste option in the user interface, using code like the following:

```
if UIPasteboard.general.hasStrings {
   // Enable string-related control...
}
```

Use the following properties to avoid user notifications and alerts when the system doesn't establish user intent:

- `numberOfItems`
- `types` , `types(forItemSet:)`
- `itemSet(withPasteboardTypes:)`
- `hasColors` , `hasImages` , `hasStrings` , `hasURLs`
- `canLoadObject(ofClass:)` , `canLoadObject(ofClass:)`
- any of the pattern-detection methods in the Detecting patterns of content in pasteboard items group in the topic groups below

The system notifies the user when you access properties or call methods that pull data from the pasteboard if the system doesn't determine that the user intends to access that data.

Pasteboard items and representation types

When you write an object to a pasteboard, the pasteboard stores it as a *pasteboard item*. A pasteboard item consists of one or more key-value pairs in which the key identifies the representation type (sometimes called a *pasteboard type*) of the value.

A uniform type identifier frequently functions as the key for a representation type. For example, you can use the `UTTypeJPEG` uniform type identifier (a constant for `public.jpeg`) as a representation type key for JPEG data.

For a discussion of uniform type identifiers, and a list of common ones, see Uniform Type Identifiers.

Your app can use any string to name a representation type; however, for app-specific data types, it's best practice to use reverse-DNS notation to ensure the uniqueness of the type (for example, `com.myCompany.myApp.myType`).

You can provide flexibility for data sharing by providing multiple representation types for a pasteboard item during a copy or cut operation. Various contexts within your app or other apps can then make use of an appropriate representation type. For example, when a user copies an image, your app can write multiple representation types, such as in the PNG, JPEG, and GIF data formats, to a pasteboard. If the original image is in PNG format, but the receiving app can handle only GIF images, it can still use the pasteboard data.

For more about representation types, read the discussion for the `types` instance method.

Sharing pasteboards between devices

When a user signs into iCloud, the general pasteboard automatically transfers its contents to nearby devices that use the same iCloud account. You can control Handoff behavior when writing contents to the general pasteboard, and can set an expiration for items, using the `setItemProviders(_:localOnly:expirationDate:)`, `setObjects(_:localOnly:expirationDate:)`, or `setItems(_:options:)` methods, as follows:

- To exclude a pasteboard from Handoff, specify `false` for the `localOnly` parameter, or call the `setItems(_:options:)` method with the `localOnly` option.
- To indicate an expiration time and date for copied data, provide the `expirationDate` parameter, or call the `setItems(_:options:)` method with the `expirationDate` option. At the time and date that you set, the system removes the pasteboard items from the pasteboard.

Using pasteboards with other objects

Although the `UIPasteboard` class is central to copy, paste, and duplicate operations, you can employ protocols and instances of other UIKit classes in these operations as well, such as the following:

- `UIEditMenuInteraction` — Displays a menu with edit actions, such as Copy, Cut, Paste, Select, and Select All, above or below the selection.
- `UIActivityItemsConfigurationReading` — Objects implement this protocol to indicate that they support copying and sharing data.
- `UIPasteConfigurationSupporting` — Objects implement this protocol to indicate whether they support pasting with a specific `UIPasteConfiguration`.
- `UIResponder` — Responders implement `canPerformAction(_:withSender:)` to enable or disable commands in the above-mentioned menu based on the current context.
- `UIResponderStandardEditActions` — Responders implement methods that this informal protocol declares to handle the chosen menu commands (for example, `copy:` and `paste:`).

A typical app that implements copy, paste, and duplicate operations also manages and presents related selections in its user interface. In addition, your app needs to coordinate changes in pasteboard content with changes to its data model, as appropriate for your app.
