

Quick Actions for iOS and Android

Control your app shortcut items on **IOS** and **Android** platform on the home screen and in app library.



Platform Supports: **iOS 10.3+** **iPadOS 10.3+** **Android API 25+**

QuickActions

For control quick actions in runtime.

Member	Description
static bool LastPerformedId { get; }	Retrieves the Id of the last performed quick action. If app launched from quick action this property will contain the id of performed quick action.
static event Action<string> Performed	Raised when a quick action is performed Returns Id of quick action.
static bool Add (QuickActionItem item)	Add new quick action. Can only add quick actions with an Id that has not yet been added. Will return false if quick action with the same Id already exists or if adding failed. Will return true if quick action added successfully.
static void Add (List<QuickActionItem> shortcutItems)	Add multiple quick actions. Can only add quick actions with an Id that has not yet been added.
static List<QuickActionItem> GetAll ()	Retrieves all added quick actions.
static QuickActionItem Get (string id)	Retrieves quick action by Id . Null will be returned if quick action does not exist.
static bool Remove (QuickActionItem item)	Remove quick action by item Id . Will return false if quick action with the same Id does not exist or if removing failed. Will return true if quick action removed successfully.
static bool Remove (string id)	Remove quick action by Id . Will return false if quick action with the same Id does not exist or if removing failed. Will return true if quick action removed successfully.
static void RemoveAll ()	Removes all quick actions.

static bool IsAdded (QuickActionItem item)	Checks by item Id if quick action is already added.
static bool IsAdded (string id)	Checks by Id if quick action is already added.
static bool IsPlatformSupported	Checks if the platform is supported quick actions.

Example to add quick action

```
var item = new QuickActionItem("some_id", "some_title", "some_subTitle", IconType.None, "some_userInfo");
var result = QuickActions.Add(item);
Debug.Log(result ? "Quick action: added" : "Quick action: add failed");
```

QuickActionItem

Used for display and manage quick action.

QuickActionItem is **Serializable**, that means you can **serialize** it to **Json** or **deserialize** from **Json**, or serialize in unity inspector.

Member	Description
string Id { get; }	Unique identifier of quick action for management and identification.
string Title { get; }	User-visible title for the quick action.
string SubTitle { get; }	User-visible subtitle for the quick action.
IconType IconType { get; }	Constants for system-provided icons.
string UserInfo { get; }	App-specific information that you can provide for use when your app performs the quick action.



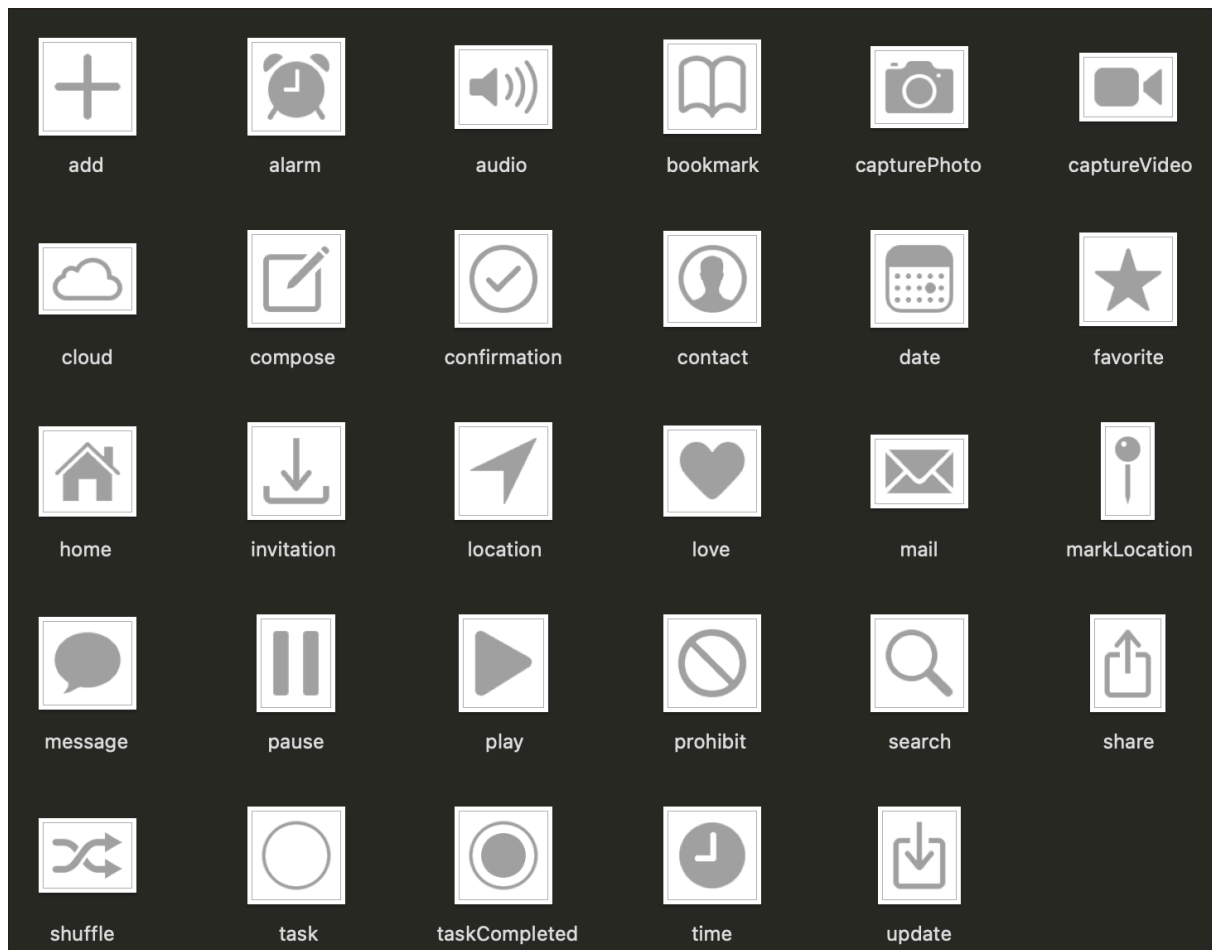
Note about **SubTitle** for **Android**:

Some custom launchers may support displaying the **SubTitle**, but this is a rare case. Most stock launchers (e.g., Pixel Launcher, Samsung One UI, etc.) do not display the **SubTitle**.

Example of serialized QuickActionItem

```
{
  "id":"some_id",
  "title":"some_title",
  "subTitle":"some_subTitle",
  "iconType":"some_iconType",
  "userInfo":"some_userInfo"
}
```

IconType



QuickActionsUtils

To work with **QuickActionItem** as **Json**, there is a **QuickActionsUtils** class that has the necessary API.

Member	Description
<code>string ItemToJson(this QuickActionItem quickActionItem)</code>	Serialize QuickActionItem to json string.
<code>string ItemsToJson(this IEnumerable<QuickActionItem> items)</code>	Serialize an enumerator QuickActionItems to json string.
<code>bool TryToCreateItemsFromJson(string json, out List<QuickActionItem> items)</code>	Try to deserialize json as list of QuickActionItems .
<code>bool TryToCreateItemFromJson(string json, out QuickActionItem item)</code>	Try to deserialize json as QuickActionItem .

UnityAppController in IOS

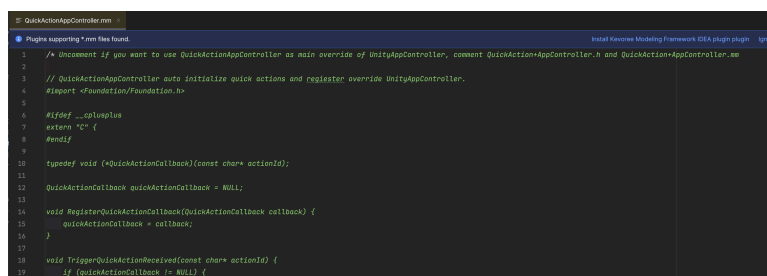
In **IOS** native side uses **IMPL_APP_CONTROLLER_SUBCLASS** to override methods from **UnityAppController**. If you have your own class for overriding **UnityAppController**, functionality related to handling **Quick Actions** that launch the app is likely to not work. To fix this, you can call the initialization of the **Quick Actions** plugin from your custom **UnityAppController** implementation.



If you do not have a custom implementation of **UnityAppController**, everything will work out of the box.

Custom Quick Actions initialization

1. You need to comment all code in `QuickActionAppController.mm`



```
1  /* Uncomment if you want to use QuickActionAppController as main override of UnityAppController, comment QuickActionAppController.h and QuickActionAppController.mm
2
3  // QuickActionAppController auto initialize quick actions and register override UnityAppController.
4  #import <Foundation/Foundation.h>
5
6  #ifdef __cplusplus
7  extern "C" {
8  #endif
9
10 typedef void (*QuickActionCallback)(const char* actionId);
11
12 QuickActionCallback quickActionCallback = NULL;
13
14 void RegisterQuickActionCallback(QuickActionCallback callback) {
15     quickActionCallback = callback;
16 }
17
18 void TriggerQuickActionReceived(const char* actionId) {
19     if (quickActionCallback != NULL) {
```

2. Uncomment all code in file `QuickAction+AppController.h` and `QuickAction+AppController.mm`
3. Manual initialize Quick Actions in your override **UnityAppController**.
 - a. Add dependencies: `#import "QuickAction+AppController.h"`
 - b. In method `(BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions` add code:

```
[self initialize:application withOptions:launchOptions];
```
 - c. Inside the method `(void)application:(UIApplication *)application performActionForShortcutItem:(UIApplicationShortcutItem *)shortcutItem completionHandler:(void (^)(BOOL))completionHandler` add code:

```
[self initializePerformAction: application shortcutItem: shortcutItem completionHandler: completionHandler];
```

Example of **AppControllerClass**:

```
#import <Foundation/Foundation.h>
#import "UnityAppController.h"
#import "QuickAction+AppController.h"

@interface AppController: UnityAppController {}
@end
```

```

IMPL_APP_CONTROLLER_SUBCLASS(AppController)
@implementation AppController
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    [self initialize:application withOptions:launchOptions];
    return [super application:application didFinishLaunchingWithOptions:launchOptions];
}

- (void)application:(UIApplication *)application performActionForShortcutItem:(UIApplicationShortcutItem *)shortcutItem completionHandler:(void (^)(BOOL))completionHandler {
    [self initializePerformAction: application shortcutItem: shortcutItem completionHandler:
completionHandler];
}
@end

```

Now the functionality of capturing Quick Action that launched the application will work well.