Article

Connecting players with their friends in your game

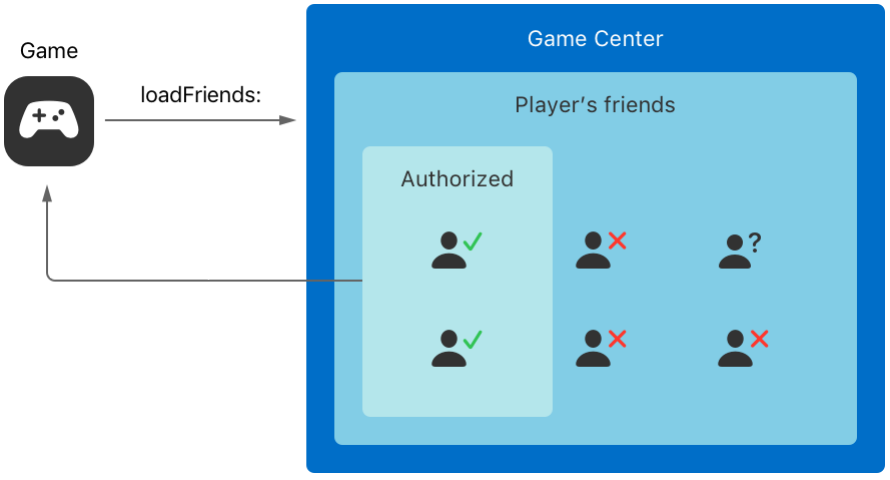
Give players the ability to connect and interact with friends in your game.

Overview

Friends are such an important part of social play that GameKit gives you privacy-friendly access to a player's Game Center friends. For example, you can show what level friends are on in your game, launch a mission together with friends, or show a friends-only recurring leaderboard.

If the player has friends, you can access the list with their permission. The first time a player tries to use functionality in your game that requires Game Center friends access, the system displays a prompt asking the player whether your game can access their friends list. If the player grants access to their friends, you'll have access only to friends who also grant access.

For example, if a player grants access, then only the friends who also grant your game access to their friends appear in the friends list. When a player grants access to their friends, GameKit syncs the permission across the player's devices so the player only needs to respond to this system prompt once.



> **Important**
>
> The code listings in this article use GameKit asynchronous methods that you invoke within a `Task` structure in SwiftUI. For details on asynchronous flows, see Concurrency.

Sending friend requests

Game Center provides an interface for sending and receiving friend requests including an inbox that shows requests that a player may have missed. Players can also send friend requests within your game. For example, use this feature to encourage players to add more friends in your multiplayer game.

Add a friend request button to your interface using an add person SF Symbol (for example, `person.crop.circle.badge.plus`) for the button image. For details on using SF Symbols, see Configuring and displaying symbol images in your UI.

Implement the button's action to call the `presentFriendRequestCreator(from:)` method that presents the Game Center request friend view controller. After the player sends the friend request using this interface, this method dismisses the view controller and returns control to your game. Before continuing, check whether an error occurs sending the friend request.

```
do {
    try GKLocalPlayer.local.presentFriendRequestCreator(from: rootViewController!)
} catch {
    print("Error: \(error.localizedDescription).")
}
```

Providing a reason to access the friends list

You must provide a reason to access a player's friends by adding the `NSGKFriendListUsageDescription` key to the information property list. The system displays the reason in the prompt it presents to the player to grant your game access. If you don't provide this key, an error occurs when you attempt to access the friends.

For details on the system prompt, see Requesting access to protected resources.

Checking whether the player grants access

Display the prompt to access friends when it's convenient for the player by checking whether the player grants you permission to access their friends using the `loadFriendsAuthorizationStatus(_:)` method.

For example, if the player hasn't denied or authorized access yet, you may want to delay accessing their friends until there's a good moment to prompt the player for permission. To ensure you allow enough time for the player to make a decision, don't display the prompt during game play.

Implement the completion handler you pass to this method to take the appropriate action:

- If the status is `GKFriendsAuthorizationStatus.authorized`, you can access the friends list.
- If the status is `GKFriendsAuthorizationStatus.notDetermined`, the system prompts the player when you access the friends.
- If the status is `GKFriendsAuthorizationStatus.denied` or `GKFriendsAuthorizationStatus.restricted`, the system won't prompt the player and you need to delete any friends data you previously collected for this player.

```
do {
    let authorizationStatus = try await GKLocalPlayer.local.loadFriendsAuthorizatior

    // Handle GKFriendsAuthorizationStatus.
    switch authorizationStatus {
      case .notDetermined:
        // The player hasn't denied or authorized access to their friends.
        let friends = try await GKLocalPlayer.local.loadFriends()
        // Insert your friends code here.
      case .denied:
        // The player denies your request to access their friends.
      case .restricted:
        // Delete friends data from your game.
      case .authorized:
        // The player authorizes your request to access their friends.
        let friends = try await GKLocalPlayer.local.loadFriends()
        // Insert your friends code here.
    @unknown default:
        print("Status unknown.")
    }
} catch {
    print("Error: \(error.localizedDescription).")
}
```

Accessing the player's friends

Once the player grants access, use the `loadFriends(_:)` method to get the friends who also grant you access to their friends.

```
do {
    let friends = try await GKLocalPlayer.local.loadFriends()
    // Insert your friends code here.
}
catch {
    print("Error: \(error.localizedDescription).")
}
```

For example, you can display details on their friends leaderboard scores to a player. Pass the friends array to the `loadEntries(for:timeScope:completionHandler:)` leaderboard method to get their avatar, display name, and score on the leaderboard.

See Also

Players

- Saving the player's game data to an iCloud account

  Save game data during play or after a game in the player's iCloud account that's accessible from any device.

- Protecting the player's privacy using scoped identifiers

  Use the scoped identifiers that GameKit provides you as player IDs when transmitting or saving player data.

`class GKLocalPlayer`

The local player who signs in to Game Center on the device running the game.

`class GKPlayer`

A remote player who the local player running your game can invite and communicate with through Game Center.

`class GKBasePlayer`

A class that provides common data and methods for the different player objects.

`protocol GKLocalPlayerListener`

A protocol that handles events for Game Center players.

```
static let GKPlayerAuthenticationDidChangeNotificationName:
NSNotification.Name
```

A notification that posts after GameKit authenticates the local player.

```
static let GKPlayerDidChangeNotificationName: NSNotification.Name
```

A notification that posts when a player object's data changes.