Article

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.

Overview

If you store player data outside of Game Center, use the scoped player identifiers that GameKit provides for you to identify the local player’s data — not their name and avatar, which may change, or other personal information that’s private. The identifiers that Game Center provides are persistent and scoped to your game instances.

GameKit scopes the identifiers of other players in the game instance to each leaderboard object or to each multiplayer match object, not your game instances (except for friend identifiers as described below). Therefore, in a multiplayer game, save the game data for the local player only, not the data for the other players in the match.

For design guidance on protecting the player’s privacy, see Human Interface Guidelines > Privacy.

Initialize the local player

Before you can get a scoped identifier, initialize the local player who is running your game on the device. During the initialization process, the player signs in to their Game Center account if they haven’t already done so. Identifiers for a player are persistent; that is, the values are the same across all instances of your game. For more information, see Authenticating a player.

Get the local player’s scoped identifier

After initialization, you can get a unique identifier for the local player from the GKLocalPlayer shared object using the gamePlayerID property:

```
// Get the scoped local player identifier.
let identifier = GKLocalPlayer.local.gamePlayerID
```

If you need to track a player across multiple games belonging to your developer team, you can use the teamPlayerID property instead. Game Center scopes this property to instances of all games that use the same Team ID. But if you transfer your game to another team, the value of the teamPlayerID property changes.

If you use either the teamPlayerID or the deprecated, non-scoped playerID property, you need to provide details on your game's privacy practices in App Store Connect. To learn more, see App privacy details on the App Store.

If initialization fails, the gamePlayerID and teamPlayerID properties return a temporary identifier that's not persistent. If you need unique and persistent identifiers, use the scopedIDsArePersistent() method before accessing the properties. Don't rely on the format of the identifiers to determine whether they're persistent.

Later, you can use either the gamePlayerID or the teamPlayerID values in GameKit calls to identify the player.

Get scoped identifiers for friends of the local player

If you have permission from the local player to access their friends list, you can also get scoped identifiers for their friends. The friend identifiers are persistent and scoped to game instances the local player and their friends run.

You get the scoped identifiers for the friends from the GKPlayer objects that the loadFriends(_:) method returns using the gamePlayerID property. In the game instances that mutual friends run, the identifiers for the GKPlayer objects that the loadFriends(_:) method returns are the same:

```
// Load the local player's friends.
let players = try await GKLocalPlayer.local.loadFriends()

// Get scoped identifiers for the mutual friends.
for player in players {
    let let identifier = player.gamePlayerID
    // Save or use the scoped identifiers.
}
```

For details on loading the local player's friends, see Providing a reason to access the friends list and Accessing the player's friends.

See Also

Players

Connecting players with their friends in your game

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

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.

`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.