Article

Accessing the contact store

Request permission from the person to read and write their contact data.

Overview

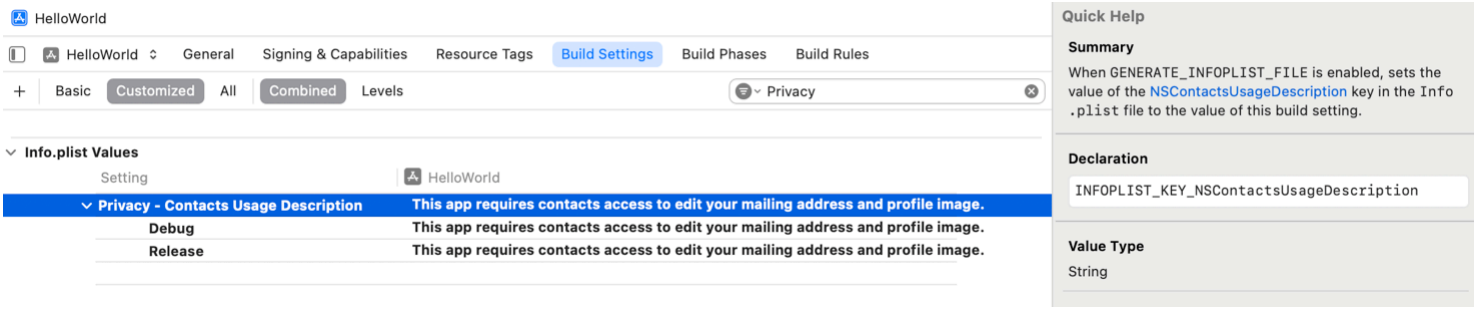
Your app can't access a person's Contacts entries until that person grants permission. When your app requests access to contact data, the person can grant either limited or full access, or they can deny the request.

Granting limited access to your app allows them to select specific contacts to share during the request authorization flow. They can expand or restrict the list of contacts your app has access to later in the Settings app. Your app has automatic access to contacts it creates, but it can only fetch, edit, or delete contacts that the person grants you access to. If the person approves full access for the app, you can create, fetch, edit, and delete contact data, including contacts you didn't create. If the person denies the request, your app gets no access to the person's data.

Contacts UI includes new APIs you can adopt for an enhanced privacy experience when someone gives full, partial, or no access to your app. Use `ContactAccessButton` and `contactAccessPicker(isPresented:completionHandler:)` to let the person control which contacts your app can access.

Provide a contacts usage description

In Xcode, configure the required `NSContactsUsageDescription` setting in the Build Settings tab of your app target. The value for this setting is a string that explains why your app needs access to a person's contacts. The system displays the string when prompting the person to authorize your app for access. The following image shows an example contacts usage description for an app that requires contacts access to edit your mailing address and profile image:



Important

Your app crashes if you attempt to access contact data without a contacts usage description.

Check your app authorization status

Verify the authorization status of your app before presenting features that require Contacts access in your UI. If your app hasn't requested access yet, any attempts to use these features automatically prompt the person for Contacts access. If the person previously denied access to your app, attempts to use these features fail with a `CNError.Code.authorizationDenied` error.

The authorization status of your app is `CNAuthorizationStatus.notDetermined` until the person authorizes or denies access. The person can approve or deny your app's request for authorization, then change the authorization status later in the Settings app.

To determine your authorization status, call the `authorizationStatus(for:)` class method of `CNContactStore` with an entity type `CNEntityType.contacts`:

```
let status = CNContactStore.authorizationStatus(for: .contacts)

switch status {
case .notDetermined:
// The person hasn't yet decided whether the app may access contact data.
case .restricted:
// The app isn't authorized and the person can't authorize the app due to restrictio
case .denied:
// The person explicitly denies access to contact data.
case .authorized:
// The person authorizes access to all contact data.
case .limited:
// The person authorizes access to some contact data.
}
```

Request access to contact data

If the authorization status of your app is `CNAuthorizationStatus.notDetermined`, prompt the person for access by initializing the contact store, then calling its `requestAccess(for:completionHandler:)` or `requestAccess(for:)` method with an entity type `contacts`:

```
// Initialize the contact store.
var store = CNContactStore()

// Request access to contacts.
do {
    let response = try await store.requestAccess(for: .contacts)
} catch {
    // Handle the error.
}
```

If the person grants limited access, your app receives a `CNAuthorizationStatus.limited` authorization status. If the person grants full access, your app receives a `CNAuthorizationStatus.authorized` status. The system remembers your app's authorization status so that subsequent calls to `requestAccess(for:completionHandler:)` or `requestAccess(for:)` don't prompt the person again.

> **Note**
>
> If the person previously granted your app contacts access (`CNAuthorizationStatus.authorized`) in an earlier OS, your app still has full access in iOS 18 and later.

Use Contacts with limited access

Your app can use the entire Contacts API when it has limited contact access. Use `ContactAccessButton` to let people choose contacts to share with your app. When someone searches for a contact, the search results present contacts that your app doesn't have access to. If the person taps the button, the system immediately grants access to the contact without prompting them for authorization, and your app receives a callback that includes the identifier of the newly added contact. To fetch information about this contact, create a fetch request that uses `predicateForContacts(withIdentifiers:)`, pass the identifier to the predicate, and execute the request:

```
@State private var searchQuery = ""
```

```
var body: some View {
    ContactAccessButton(queryString: searchQuery) { identifiers in
        // Fetch contacts whose identifiers match newly added contacts.
        fetchContacts(withIdentifiers: identifiers)
    }
}
```

For more information, see CNContactStore.

If your app needs to read or modify contacts, consider presenting the contact access picker to let people update which contacts you can access. The picker has full access to all contacts on the device regardless of your app's authorization status. When the person dismisses the picker, your app receives a callback that returns the identifiers of additional contacts the person chose. The callback doesn't provide any information about contacts you can no longer access or those you can already access. The following example creates a fetch request that searches for all approved contacts, executes the request, and then uses the identifiers to highlight newly added contacts in the fetch result in your UI:

```
// Indicates whether to present the contact access picker.
@State private var isPresented = false

var body: some View {
    // Tap the add button to present the picker.
    addButton
        .contactAccessPicker(isPresented: $isPresented) { identifiers in
            // Fetch all contacts your app can access.
            let contacts = fetchContacts()

            // Highlight contacts whose identifiers match newly selected contacts.
            highlightContacts(withIdentifiers: identifiers, in: contacts)
        }
}

// An add button.
private var addButton: some View {
    Button {
        isPresented.toggle()
    } label: {
        Label("Add contacts", systemImage: "person.fill.badge.plus")
    }
}
```

If you don’t adopt the contact access button or contact access picker, the person needs to use the Settings app to control which contacts you can access.

Add entitlement to view or update notes

To read or write the `note` field from a contact in iOS 13, macOS 13, or later, add the `com.apple.developer.contacts.notes` entitlement to your app. The entitlement requires permission from Apple to use, and you can’t publicly distribute your app until you have permission to use it. For more information about adding the entitlement and requesting permission, see `com.apple.developer.contacts.notes`.

See Also

Essentials

{} Accessing a person’s contact data using Contacts and ContactsUI

Allow people to grant your app access to contact data by adding the Contact access button and Contact access picker to your app.

`class CNContactStore`

The object that fetches and saves contacts, groups, and containers from the user’s Contacts database.

`NSContactsUsageDescription`

A message that tells people why the app is requesting access to their contacts.

`com.apple.developer.contacts.notes`

A Boolean value that indicates whether the app may access the notes in contact entries.