Sample Code

Defining your app's Focus filter

Customize your app's behavior to reflect the device's current Focus.

Download

iOS 16.0+ | iPadOS 16.0+ | Xcode 14.0+

Overview

Use App Intents to define your app's Focus filters, act on changes from the system, and filter notifications based on its parameters.

> Note
>
> This sample code project is associated with WWDC22 session 10121: Meet Focus filters.

Configure the sample code project

Configure the iOS and App Intents extension targets to include your development team and a bundle identifier. The bundle identifier needs to support the App Groups capability.

If you're using a suite name other than *group.exampleChatApp*, modify it in the following files: `Repository.swift`, `ExampleAppIntentsExtension.entitlements`, and `ExampleChatApp.entitlements`.

Create a Focus filter

After launching the sample app, create a Focus filter for it by choosing Settings > Focus > Do Not Disturb (or another Focus) > Add Filter > Example Chat App. Modify the filter parameters, and then tap the Add button. Turn on the modified Focus to receive those parameters in the sample app's

App Intents extension. The system calls the perform function of the object conforming to `SetFocusFilterIntent`, and all @Parameters are available in the Focus filter. When you turn off the modified Focus, the system provides the sample app's `SetFocusFilterIntent` conforming object with the following default parameters:

- `alwaysUseDarkMode: false`
- `status: nil`
- `account: nil`

Define Focus filter parameters

To define Focus filter parameters for an object that conforms to the `SetFocusFilterIntent` parameter, the sample project annotates the relevant variables with the `@Parameter` property wrapper. It provides a default value for nonoptional parameters unless the filter requires the system to prompt for this value each time it enables it.

```
struct ExampleFocusFilter: SetFocusFilterIntent {
    /// Providing a default value ensures setting this required Boolean value.
    @Parameter(title: "Use Dark Mode", default: false)
    var alwaysUseDarkMode: Bool

    @Parameter(title: "Status Message")
    var status: String?

    /// A representation of a chat account this app uses for notification filtering
    /// The user receives suggestions from the suggestedEntities() function that Acc
    @Parameter(title: "Selected Account")
    var account: AccountEntity?
}
```

Use custom parameters by adding app entity conformance

To use a custom object as a `SetFocusFilterIntent` parameter, the sample project adds App `Entity` conformance to the object. In the example below, `AppEntity` conformance allows the system to use the `AccountEntity` object as a `SetFocusFilterIntent @Parameter`:

```
struct AccountEntity: AppEntity {
    static var typeDisplayRepresentation: TypeDisplayRepresentation {
        TypeDisplayRepresentation(name: "A chat account")
    }
```

```
    static var defaultQuery = AccountEntityQuery()

    let id: String
    let displayName: String
    let displaySubtitle: String
    let image: DisplayRepresentation.Image

    var displayRepresentation: DisplayRepresentation {
        DisplayRepresentation(title: "\(displayName) account",
                              subtitle: "\(displaySubtitle)",
                              image: image)
    }
}
```

Suggest app entities using entity queries

The sample project adds a default entity query to suggest values when configuring the Focus filter in Settings. In the example below, the `AccountEntityQuery` that `AccountEntity` uses suggests chat accounts that are currently logged in:

```
struct AccountEntityQuery: EntityQuery {
    func entities(for identifiers: [AccountEntity.ID]) async throws -> [AccountEntit
        Repository.shared.accountsLoggedIn.filter {
            identifiers.contains($0.id)
        }
    }

    func suggestedEntities() async throws -> [AccountEntity] {
        Repository.shared.accountsLoggedIn
    }
}
```

Filter notifications using the app context

To filter notifications with Focus filters, the sample project sets the `filterCriteria` parameter for the notifications, and then lets the system know whether to suppress them by providing a predicate that evaluates against `filterCriteria`. In the example below, the `FocusFilterAppContext appContext` variable returns a predicate that states: If there's an account `@Parameter` for this filter, suppress the notification unless the `filterCriteria` matches the `account.id` value:

```
struct ExampleFocusFilter: SetFocusFilterIntent {
    var appContext: FocusFilterAppContext {
        logger.debug("App Context Called")
        let predicate: NSPredicate
        if let account = account {
            predicate = NSPredicate(format: "SELF IN %@", [account.id])
        } else {
            predicate = NSPredicate(value: true)
        }
        return FocusFilterAppContext(notificationFilterPredicate: predicate)
    }
}
```

Use an App Intents extension for background execution

The sample project adds an App Intents extension to handle Focus filters while the app is in the background. It sets the target membership of the object conforming to `SetFocusFilterIntent` to the App Intents extension only, and uses App Groups, or custom services, to share filter data between the extension and the app.

See Also

Focus filters

`protocol` **`SetFocusFilterIntent`**

An interface for providing an app intent that you use to adapt your app's behavior when Focus changes.

`struct` **`FocusFilterAppContext`**

A type that contains app-specific contextual information for a particular Focus, such as the notification filter criteria to apply.

`struct` **`FocusFilterSuggestionContext`**

A type you use to suggest app configurations for a given Focus.