Structure

PurchaseAction

An action that starts an In-App Purchase.

StoreKit | SwiftUI | iOS 17.0+ | iPadOS 17.0+ | Mac Catalyst 17.0+ | macOS 14.0+ | tvOS 17.0+ | visionOS 1.0+ | watchOS 10.0+

```
@MainActor @preconcurrency
struct PurchaseAction
```

Overview

StoreKit provides several APIs you can use to enable customers to initiate a purchase. Choose the API that suits your app's implementation, specifically:

- Use `PurchaseAction` for apps that use SwiftUI, including multi-scene apps for visionOS.
- Use `purchase(confirmIn:options:)` for apps that use UIKit.
- Use `purchase(options:)` if your app runs on watchOS or macOS.

> **Important**
>
> If you use StoreKit views such as `ProductView`, `StoreView`, or `SubscriptionStoreView` you don't need to call any other API to initiate a purchase. StoreKit manages the purchase action automatically, including presenting the purchase confirmation UI. For more information, see StoreKit views.

Use the purchase action API

Use `PurchaseAction` instead of `purchase(options:)` for SwiftUI implementations, including multi-scene apps for visionOS. Call the instance to start an in-app purchase.

To use this API, read the `PurchaseAction` environment value to get an instance of the structure for a given `Environment`. You call the instance directly because it defines a `callAsFunction(_:options:)` method that Swift calls when you call the instance.

When you initiate an in-app purchase, the system presents UI for the customer to confirm the purchase details. The purchase action you get from the environment automatically includes the UI context. It presents the confirmation UI in proximity to the scene in which the view displays.

The following code shows an example of starting an in-app purchase when a person taps a button:

```
struct PurchaseExample: View {
    @Environment(\.purchase) private var purchase: PurchaseAction
    let product: Product
    let purchaseOptions: [Product.PurchaseOption]

    var body: some View {
        Button {
            Task {
                let purchaseResult = try? await purchase(product, options: purchaseO
                 // Process the purchase result.
            }
        } label: {
            Text(product.displayName)
        }
    }
}
```

Note that the second line in the code example can omit the type name, as follows, because the compiler can infer the type:

```
@Environment(\.purchase) private var purchase
```

Topics

Calling the action

```
func callAsFunction(Product, options: Set<Product.PurchaseOption>)
async throws -> Product.PurchaseResult
```

Starts an in-app purchase for the indicated product and purchase options.

Instance Methods

```
func callAsFunction(AdvancedCommerceProduct, compactJWS: String,
options: Set<AdvancedCommerceProduct.PurchaseOption>) async throws ->
AdvancedCommerceProduct.PurchaseResult
```

Relationships

Conforms To

`Sendable`, `SendableMetatype`

See Also

Purchase requests and results

```
func purchase(options: Set<Product.PurchaseOption>) async throws ->
Product.PurchaseResult
```

Initiates a purchase for the product with the App Store and displays the confirmation sheet.

```
enum PurchaseResult
```

The result of a purchase.