Type Property

updates

The asynchronous sequence that emits a transaction when the system creates or updates transactions that occur outside the app or on other devices.

iOS 15.0+ | iPadOS 15.0+ | macOS 12.0+ | tvOS 15.0+ | visionOS 1.0+ | watchOS 8.0+

```
static var updates: Transaction.Transactions { get }
```

Mentioned in

- Supporting win-back offers in your app
- Supporting subscription offer codes in your app
- Testing win-back offers in the sandbox environment
- Testing purchases made outside your app
- Getting started with In-App Purchase using StoreKit views

Discussion

Use `updates` to receive new transactions while the app is running. This sequence receives transactions that occur outside of the app, such as Ask to Buy transactions, subscription offer code redemptions, and purchases that customers make in the App Store. It also emits transactions that customers complete in your app on another device.

Note that after a successful in-app purchase on the same device, StoreKit returns the transaction through `Product.PurchaseResult.success(_:)`.

Important

Create a Task to iterate through the transactions from the listener as soon as your app launches. If your app has unfinished transactions, the updates listener receives them once, immediately after the app launches. Without the Task to listen for these transactions, your app may miss them.

The following example shows a class that creates a Task when it initializes. The task retrieves and processes any unfinished transactions.

```
final class TransactionObserver {

    var updates: Task<Void, Never>? = nil

    init() {
        updates = newTransactionListenerTask()
    }

    deinit {
        // Cancel the update handling task when you deinitialize the class.
        updates?.cancel()
    }

    private func newTransactionListenerTask() -> Task<Void, Never> {
        Task(priority: .background) {
            for await verificationResult in Transaction.updates {
                self.handle(updatedTransaction: verificationResult)
            }
        }
    }

    private func handle(updatedTransaction verificationResult: VerificationResult<Tr
        guard case .verified(let transaction) = verificationResult else {
            // Ignore unverified transactions.
            return
        }

        if let revocationDate = transaction.revocationDate {
            // Remove access to the product identified by transaction.productID.
            // Transaction.revocationReason provides details about
            // the revoked transaction.
            <#...#>
        } else if let expirationDate = transaction.expirationDate,
            expirationDate < Date() {
            // Do nothing, this subscription is expired.
            return
        } else if transaction.isUpgraded {
            // Do nothing, there is an active transaction
            // for a higher level of service.
            return
        } else {
            // Provide access to the product identified by
            // transaction.productID.
            <#...#>
        }
    }

}
```

The `updates` listener receives unfinished transactions just once at app launch, but you can use the `unfinished` listener to get your app's unfinished transactions at any time. For information on finishing transactions, see `finish()`.

See Also

Related Documentation

Supporting subscription offer codes in your app

Provide subscription service for customers who redeem offer codes through the App Store or within your app.

Transaction history and entitlements

`struct Transaction`

Information that represents the customer's purchase of a product in your app.

`static var all: Transaction.Transactions`

A sequence that emits all the customer's transactions for your app.

`static var currentEntitlements: Transaction.Transactions`

A sequence of the latest transactions that entitle a customer to In-App Purchases and subscriptions.