Structure

Transaction

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

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

```
struct Transaction
```

Mentioned in

- Supporting subscription offer codes in your app
- Supporting win-back offers in your app
- Testing purchases made outside your app
- Testing win-back offers in the sandbox environment
- Choosing a receipt validation technique

Overview

A *transaction* represents a successful In-App Purchase. The App Store generates a transaction each time a customer purchases an In-App Purchase product or renews a subscription. For each transaction that represents a current purchase, your app unlocks the purchased content or service and finishes the transaction.

Use the `Transaction` type to perform these transaction-related tasks:

- Get the customer's transaction history, latest transactions, and current entitlements to unlock content and services.
- Access transaction properties.
- Finish a transaction after your app delivers the purchased content or service.

- Access the raw JSON Web Signature (JWS) string and supporting values to verify the transaction information.
- Listen for new transactions while the app is running.
- Begin a refund request from within your app.

Access transaction history and current entitlements

Your app doesn't create transaction objects. Instead, StoreKit automatically makes up-to-date transactions available to your app, including when someone launches the app for the first time.

> **Related sessions from WWDC22**
>
> Session 110404: Implement proactive in-app purchase restore

You access transactions in several ways:

- Get transaction history anytime by accessing the static `all` sequence, or get just the most recent transaction for a product with the `latestTransaction` property of `Product`.
- Receive notifications for new transactions while your app is running when customers complete a purchase outside of the app, including on another device, through the transaction listener, `updates`.
- Access the latest transaction for a subscription group through the subscription status API, using `transaction`.
- After a successful In-App Purchase, StoreKit returns the transaction through `Product.PurchaseResult.success(_:)`.

The most important use of transaction information is for determining which In-App Purchases the customer has paid access to, so your app can unlock the content or service. The `currentEntitlements` API provides the information you need to unlock all of the customer's paid content in your app. Use `currentEntitlements` to get a list of transactions for all the products the customer is currently entitled to, including non-consumable In-App Purchases and currently active subscriptions.

Verify transactions

The App Store cryptographically signs transaction information in JWS format. StoreKit automatically validates and returns the signed information, wrapped in a `VerificationResult`. When the `VerificationResult` wraps a `Transaction` value, it provides the raw JWS string in the `jwsRepresentation` property. If you get a transaction through `VerificationResult`

`.verified(_:)`, the information passed validation. If you get it through `VerificationResult.unverified(_:_:)`, the information didn't pass StoreKit's automatic validation. Your app can immediately access the transaction information in the Transaction properties.

To perform your own validation on the device, use the verification result's `jwsRepresentation` string, and use the provided convenience properties `headerData`, `payloadData`, and `signatureData`. For added control and security, send the `jwsRepresentation` to your server to verify. Consider using the App Store Server Library to implement your verification. The library provides the functions `verifyAndDecodeTransaction` and `verifyAndDecodeRenewalInfo` in each language the library supports. For more information, see Simplifying your implementation by using the App Store Server Library.

> **Tip**
>
> The `jwsRepresentation` is the same as the `JWSTransaction` that the App Store Server API returns and to the `JWSTransaction` that you receive in `App Store Server Notifications V2`. You can validate them on your server in the same way.

If StoreKit returns a transaction as verified, the transaction is valid for the device. For information about performing your own verification for a device, see `deviceVerification`.

For more information about JWS, see the IETF RFC 7515 specification.

Access purchases made with the original API

All In-App Purchases that customers make are equally available to your app in this `Transaction` API, and in receipts using the Original API for In-App Purchase, as follows:

- New purchases that customers make with the original API are available immediately using the `Transaction` API.
- Purchases that customers make with the `purchase(options:)` method are available in the original API when your app refreshes the receipt. For more information, see `SKReceiptRefreshRequest`.

Topics

Transaction properties

Transaction properties

The properties of a transaction, including identifiers, purchase and revocation dates and details, status, and offer details.

`var appTransactionID: String`

The unique identifier of the app download transaction.

Monitoring transaction-related changes

`static var updates: Transaction.Transactions`

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

`struct Transactions`

An asynchronous sequence of transactions.

Getting transaction history

`static func latest(for: String) async -> VerificationResult<Transaction>?`

Gets the customer's most recent transaction for an In-App Purchase.

`static var all: Transaction.Transactions`

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

`static var unfinished: Transaction.Transactions`

A sequence that emits unfinished transactions for the customer.

`SKIncludeConsumableInAppPurchaseHistory`

A Boolean value that determines whether StoreKit includes finished consumable In-App Purchases in transaction information.

Getting current entitlements

`static var currentEntitlements: Transaction.Transactions`

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

~~`static func currentEntitlement(for: String) async -> VerificationResult<Transaction>?`~~

Gets the latest transactions that entitle the customer to a specified product.

Deprecated

Finishing the transaction

`func finish() async`

Indicates to the App Store that the app delivered the purchased content or enabled the service to finish the transaction.

`static var unfinished: Transaction.Transactions`

A sequence that emits unfinished transactions for the customer.

Verifying transactions

`let deviceVerification: Data`

The device verification value you use to verify whether the transaction belongs to the device.

`let deviceVerificationNonce: UUID`

The UUID for computing the device verification value.

`let signedDate: Date`

The date that the App Store signed the JWS transaction.

Getting transaction info in JSON format

`var jsonRepresentation: Data`

The JSON representation of the transaction information.

Requesting refunds

Testing refund requests

Test your app's implementation of refund requests, and your app's and server's handling of approved and declined refunds.

`func beginRefundRequest(in: UIWindowScene) async throws -> Transaction.RefundRequestStatus`

Presents the refund request sheet for the transaction in a window scene.

`func beginRefundRequest(in: NSViewController) async throws -> Transaction.RefundRequestStatus`

Presents the refund request sheet for the transaction in a view controller.

`static func beginRefundRequest(for: UInt64, in: UIWindowScene) async throws -> Transaction.RefundRequestStatus`

Presents the refund request sheet for the specified transaction in a window scene.

`static func beginRefundRequest(for: UInt64, in: NSViewController) async throws -> Transaction.RefundRequestStatus`

Presents the refund request sheet for the specified transaction in a view controller.

`enum RefundRequestError`

The error codes for refund requests.

`enum RefundRequestStatus`

The status codes for refund requests.

Structures

`struct AdvancedCommerceInfo`

Metadata for transactions that use the Advanced Commerce API.

`struct OfferType`

The types of subscription offers for auto-renewable subscriptions.

Instance Properties

`let advancedCommerceInfo: Transaction.AdvancedCommerceInfo?`

Metadata for transactions that use the Advanced Commerce API.

~~`var offerPeriodStringRepresentation: String?`~~

The string representation of the offer period applied to the subscription offer for this transaction.

Deprecated

Type Methods

`static func all(for: String) -> Transaction.Transactions`

Gets all the transactions associated with this product ID.

`static func currentEntitlements(for: String) -> Transaction.Transactions`

Gets the transactions that entitle the user to items purchased under a product ID.

Relationships

Conforms To

`Copyable`
`CustomDebugStringConvertible`
`Equatable`
`Hashable`
`Identifiable`
`Sendable`
`SendableMetatype`

See Also

Transaction history and entitlements

`static var updates: Transaction.Transactions`

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

`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.