Purchase Interface

Online subsystems for executing in-game purchases.

Two interfaces power your game's ability to support user purchases: The **Store Interface**, which provides the ability to make offers to your users, and the **Purchase Interface**, which enables users to accept these offers. With the Purchase Interface, you can:

- Check to see whether a user is able to make purchases.
- Begin the process of purchasing.
- See the history of past purchases.

To retrieve or filter a list of available offers that a user might purchase, use the Store Interface.

In-Game Purchases

Purchases (in the sense of the Purchase Interface) involve the transfer of real money—not in-game currency—in exchange for access to in-game items or content. However, the Purchase Interface itself does not handle financial details like taking credit card information or setting up parental controls. Instead, those details are left to the online service. The Purchase

Interface checks for user ability to make purchases, starts the process of purchases, and can check with the online service to get a list of previously-made purchases.

Ability to Purchase

Before attempting to make a purchase, you can check to see if the user is able to do so. Some online services have account restriction features, such as parental controls, that prevent users from making purchases. You can check this information before showing offers to the user, so that you can hide or alter buttons in your user interface, or provide error messages if the user attempts to buy something without the ability to do so. To check whether a user is able to make purchases, use the `IsAllowedToPurchase` function. The Online Subsystem will have already cached this information, so this function returns a `bool` directly rather than using a delegate.

Making a Purchase

Once you're ready to make a purchase, and have selected an offer (as defined in the Store Interface), call the `Checkout` function with an `FPurchaseCheckoutRequest`. This will generally bring up a user interface specific to the online service, and will eventually call your provided delegate, of type `FOnPurchaseCheckoutComplete`. The delegate will contain success or failure information, and, if the purchase succeeded, a purchase receipt (class `FPurchaseReceipt`).

The purchase receipt contains the offer or offers that the user has purchased, using the `FReceiptOfferEntry` data structure. Within each `FReceiptOfferEntry` there are one or more line items (type FLineItemInfo), each containing an opaque string generated by the online service, and referring to one specific transaction that has been taken place. Once the item has been unlocked, credited to the user, or does whatever it does in the game, a call to `FinalizePurchase` with this ID string will confirm the sale with the online service and perform the transfer of funds.

> Be sure that you have given the item to the user and saved it to their account (if appropriate) before calling `FinalizePurchase`. On some platforms, failure to do this can result in your game charging a user real-world money for an item they do not receive.

Reviewing Past Purchases

Past purchases can be reviewed by looking over the user's receipts. Call `QueryReceipts`, which will request the receipts for a specific user. When complete, a user-provided delegate of type `FOnQueryReceiptsComplete` will receive the resulting success or failure information. Upon success, receipts for the user account will be cached locally and can be accessed through the `GetReceipts` function in the form of an array of `FPurchaseReceipt` elements. This list will include a list of all non-consumable purchases, as well as pending purchases of consumables.