Peter Gustafson writing sample
Epic Games
June 29, 2021

Ecom Interface

Overview
The Ecom Interface exposes all catalog, purchasing, and ownership entitlement features available with Epic Games.

Below are terms and definitions for the Ecom Interface:
Term	Definition	Example
Artifacts	Artifacts represent your downloadable, playable content such as game clients and DLC, and are a concept specific to the Epic Games Store (EGS).	Fortnite alien artifacts
Audience	An Audience is another type of `CatalogItemId` that acts as an abstraction layer between a content-providing Item and the Offer to sell it.	When a user purchases the base game Offer, they are granted the general audience, and the general audience in turn grants the Item for the game itself. This abstraction layer is useful for bundles like deluxe editions of games, for subsequently adding content to groups of users (for example when a new DLC releases that is included in a Season Pass), customer support, and other purposes.
`CatalogItemIds`	Think of `CatalogItemIds` as the foundational layer of other E-Commerce concepts.	Items, Audiences, and Offers.
Consumable	A Consumable is a Catalog Item that can be purchased multiple times, increasing the user's inventory count, until they are redeemed or consumed in-game.	Virtual currency or in-game Items like potions or time limited double experience modifiers.
Durables	A Durable is a `CatalogItemId` that is intended to be purchased only once and permanently resides on the user's account. A Durable is purely rights-driven content, such as the ability to equip a bonus in-game character skin.	A map pack or new campaign.
Entitlements	Entitlements are `EntitlementIds` that are unique per user, and per purchase of an Item. An Entitlement is not a `CatalogItemId`.	A user purchases the same Consumable Offer 3 times. After the purchase the user is assigned 3 different `EntitlementIds`.
Items	Items are content-providing `CatalogItemIds` with downloadable bits.	Full games or downloadable content.
Offers	Offers pair one or more `CatalogItemId` numbers with associated regional pricing (which can be free). When an offer is purchased, an entitlement is granted, and each of the subsequent Items (Audiences and associated Items) are awarded as a result.	Fortnite, Rocket League & Genshin Impact.

Example Use Case
As an example, let's say a player purchases the Super Deluxe (SD)

Edition of a game. Below are the actions involved in the game purchase:

Action	Description	Access
SD Edition purchase	An `entitlementId` is created in your account to manage your access permissions.	Specific Audiences and Levels.
`AudienceId` created	The `AudienceId` identifies the authorized audiences and your level of ownership.	Users with an authorized `AudienceId` can access your game. Make a call to the Ownership API to query all users with access to your `AudienceId`.
`ArtifactId` created	The `ArtifactId` is created in your account.	The `ArtifactId` identifies your purchase of the SD edition.

> Note: as you use the Ecom Interface, your product must have Epic Account Services (EAS) active, and must obtain [user consent](https://dev.epicgames.com/docs/services/en-US/EpicAccountServices/DataPrivacyVisibility/index.html) to access Basic Profile data. Without EAS and user consent, you will still be able to initialize the EOS SDK and the Ecom Interface, but all Ecom Interface function calls to the back-end service will fail.

Ownership vs. Entitlements

Below are descriptions how Ownership and Entitlements work in the Epic Game Store.

Entitlements
Think of Entitlements as receipts with Entitlement identification numbers that EOS records as an `entitlementId` for each receipt. Entitlements can be searched both in the SDK as well as using the API . Read how to implement the [Epic Games SDK](https://dev.epicgames.com/docs/services/en-US/CSharp/GettingStarted/index.html).

- An Entitlement record is a unique identification number for each user.

- For Consumable purchases, each transaction is assigned an entitlementId.

- Three unique store purchases means three entitlement ID numbers are created.

- Entitlement API queries return users who have purchased your game.

Ownership

The Ownership API is used to confirm the existence of the Entitlement records and allows developers to redeem and consume them.

Redeem Consumables

Your game’s backend makes a call to Epic’s backend or the game client passes an entitlement token to the game’s backend. Here's how it goes:

-   You award the user with in-game Items and mark the entitlements as redeemed. This step requires relating an outstanding `entitlementName` or the `catalogItemId` field which should be the same.

-   These fields represent the immutable offer or Item of the entitlement. The game’s backend stores a table which maps a particular `catalogItemId` to a specific set of in-game inventory.

Example
Catalog Items for redemption are managed using the backend services. As an example use case, let's say `catalogItemId` *X* is a 100 coins offer. But `catalogItemId` *Y* is a 500 coins offer. In this example, the game’s backend makes a call using either the SDK API or through RESTful service.

Upon the successful redemption, you add the appropriate in-game inventory Items to the user’s account with the following parameters:
Parameter	Description
`consumable`	All Consumable values are `true` regardless of the redemption state.
`status`	Changes from `active` pre-redemption to `redeemed` post-redemption.
`active`	The pre-redemption value is `true`; post-redemption is `false`.

Catalog Configuration

The catalog configuration is performed by the EGS Service Delivery team. Contact your Service Delivery team to set-up offers, audiences, and catalog Items.

Testing

 1. The Service Delivery team will create codes that grant redeemers a single `DevAudience` with access to all content for developers.

 2. Service Delivery can provide QA codes that have the base game and individual add-on content broken out to individual codes.

 4. This allows test teams to validate the process of starting from the base game and adding content individually.

Without EAS set to active, you can initialize the EOS SDK and the Ecom Interface, but all Ecom Interface function calls to the back-end service will fail. Below are common use cases working with the Ecom Interface:

- Make purchase offers
- Completing purchase transactions
- Verify ownership and redeem purchases

Access
To access the Ecom Interface you must get an `EOS_HEcom` handle through the Platform Interface function `EOS_Platform_GetEcomInterface`. All Ecom Interface functions require this handle as their first parameter. Next, confirm the `EOS_HPlatform` handle is ticking to receive callbacks.

In-Game Store

Catalog Offers are games and products in the [Epic Games Store (EGS)](https://www.epicgames.com/store/en-US/free-games) and provides two functions:

1. Presents Catalog Offers for purchase and enabling the user to make purchases.

2. The EOS SDK provides functionality to retrieve lists of offers from the store's catalog.

The checkout process pushes the data to the EGS Overlay to finalize the purchase. EOS notifies the game of the transaction and provides a transaction handle.

Catalog Offer Data

Catalog Offers `EOS_Ecom_CatalogOffer` are localized pricing information based on the user who is browsing the store. Price localization includes applied discounts and currency conversions. A localized price will include a currency code, such as USD (US dollars).

For example, an offer that costs 2.99 in US dollars has a value of 299.

Catalog Offer Lists

Retrieve the list of Catalog Offers defined in the Developer Portal by making a call to the `EOS_Ecom_QueryOffers` function. This function takes an `EOS_Ecom_QueryOffersOptions` data structure and returns type `EOS_Ecom_QueryOffersCallbackInfo`. Use the following parameters:

Parameter	Description
`ApiVersion`	Set to `EOS_ECOM_QUERYOFFERS_API_LATEST`.
`LocalUserId`	The local user Epic account ID.
`OverrideCatalogNamespace`	Optional product namespace.

- If the ResultCode in the `EOS_Ecom_QueryOffersCallbackInfo` that the delegate receives is successful, the offer data you requested is available in a cache.

- Use `EOS_Ecom_GetOfferCount` to find the number of offers stored in the cache.

- Use `EOS_Ecom_CopyOfferByIndex` to get a copy of an individual offer.

- Each Item within the offer contains data about images and release details associated with the Item.

- Release the offer by using `EOS_Ecom_CatalogOffer_Release`.

Display Individual Item Data

Below are data parameters and how to use them.
Property	Use
`EOS_Ecom_GetOfferItemCount`	Retrieves offer count. After retrieving an offer from the cache, make a call to `EOS_Ecom_GetOfferItemCount`.
`EOS_Ecom_CopyOfferItemByIndex`	Provides a copy of an individual `EOS_Ecom_CatalogItem`. This structure contains localized text and the ID of the Entitlement.
`EOS_Ecom_GetItemImageInfoCount`	Returns the quantity of images.
`EOS_Ecom_KeyImageInfo`	Retrieves the image URL and dimensions.
`EOS_Ecom_GetItemReleaseCount`	Gets the release details for an Item in your cache.
`EOS_Ecom_CopyItemReleaseByIndex`	Used to return an individual piece of data.
`EOS_Ecom_CatalogRelease_Release`	Releases the data once no longer needed.

Store Purchases

Three actions are involved in buying from the store:

1. Making the purchase.
2. Verifying ownership
3. Redeeming Entitlements (optional).

After a user makes a purchase, you make a call to `EOS_Ecom_Checkout` with an `EOS_Ecom_CheckoutOptions` structure. Below are the parameters to use in your call:

Parameter	Description
`ApiVersion`	Set to `EOS_ECOM_QUERYOFFERS_API_LATEST`.
`LocalUserId`	The local user Epic account ID.
`EOS_EpicAccountId`	The local user making the purchase.
`OverrideCatalogNamespace`	If not available, the current `SandboxId` is supplied as the catalog namespace.
`EntryCount`	The number of `EOS_EcomCheckoutEntry` elements supplied to the structure in the Entries parameter.
`Entries`	An array of `EOS_Ecom_CheckoutEntry` elements containing the `EOS_Ecom_CatalogOfferId` data for the purchase.

After making the call, EOS generates a purchase token.

- EOS uses the purchase token to open its own overlay for the user to review the purchase, select payment options, and confirm or cancel the transaction.

- When the overlay closes, the `EOS_Ecom_CheckoutCallbackInfo` will run with an `EOS_Ecom_CheckoutCallbackInfo` parameter detailing the transaction.

- Within the callback information, the TransactionId will be non-null if the transaction succeeded, and null if it failed or was canceled.

- If any additional calls to `EOS_Ecom_Checkout` are made before the callback, it will return an `EOS_AlreadyPending` error.

Access Purchase Data

Below are the properties to access the purchase data.

Property	Use
`EOS_Ecom_CopyTransactionById`	If you have a valid transaction ID following a successful user purchase, pass it to the `EOS_Ecom_CopyTransactionById` function to receive an `EOS_Ecom_HTransaction`.
`EOS_Ecom_Transaction_GetEntitlementsCount`	Returns the quantity of the Entitlements in the transaction.
`EOS_Ecom_Entitlement_Release`	Releases the data.

Fulfill Purchases

After making a purchase, the user's Epic account is assigned an Entitlement. Some users may not see the notification in the developer portal.

In some cases, fulfilling a purchase can be as simple as checking if the user owns a specific Entitlement. In these cases, verifying ownership through the EOS SDK is sufficient. In other cases, such as purchases involving Consumable Items or game currency, you may need to fulfill the order in-game or with a third-party backend service by redeeming the Entitlement.

Enumerating Entitlements

To retrieve a user's Entitlements, call the `EOS_Ecom_QueryEntitlements` function with an `EOS_Ecom_QueryEntitlementsOptions` using the parameters below:

Parameter	Description
`ApiVersion`	Set to `EOS_ECOM_QUERYOFFERS_API_LATEST`.
`LocalUserId`	The local user Epic account ID.
`EntitlementNames`	An array of Entitlement names.
`EntitlementNameCount`	The quantity of Entitlement names included in the EntitlementNames property. Accepts up to a maximum of `EOS_ECOM_QUERYENTITLEMENTS_MAX_ENTITLEMENT_IDS`. If 0 is provided, your request returns all Entitlements associated with the user account.
`bIncludeRedeemed`	If `true`, the user's redeemed Entitlements is returned.
`EOS_Ecom_GetEntitlementsCount`	Returns the quantity of Entitlements in cache (optional).
`EOS_Ecom_CopyEntitlementByIndex`	Retrieves a copy of an individual element (optional).

When the operation completes, EOS caches the resulting information and invokes your callback function. If the `ResultCode` returns `EOS_Success`, the cache contains the data you requested.

Redeem an Entitlement

After fulfilling an Entitlement, make a call to `EOS_Ecom_RedeemEntitlements` with an `EOS_Ecom_RedeemEntitlementsOptions` structure. Below are the parameters:

Parameter	Description
`ApiVersion`	Set to `EOS_ECOM_QUERYOFFERS_API_LATEST`.
`LocalUserId`	The local user Epic account ID.
`EntitlementIdCount`	The number of elements in the `EntitlementIds`.
`EntitlementIds`	The Entitlements identification numbers.

Your `EOS_Ecom_OnRedeemEntitlementsCallback` callback returns an `EOS_Ecom_RedeemEntitlementsCallbackInfo` structure.

After redeeming an Entitlement, it will no longer display in the results from `EOS_Ecom_QueryEntitlements` calls unless the `EOS_Ecom_QueryEntitlementsOptions` parameter has its `bIncludeRedeemed` set to `true`.

Ownership Verification

The Ecom Interface provides two methods for ownership verification.

Online Method

The online method integrates directly with the Epic Entitlement Service. It’s useful for trusted game servers or less-secure client systems for simple validation.

To determine if a user owns a specific Catalog Item, make a call to `EOS_Ecom_QueryOwnership` to get ownership information from the server. Use the parameters below:

Parameter	Description
`ApiVersion`	Set to `EOS_ECOM_QUERYOFFERS_API_LATEST`.
`LocalUserId`	The local user Epic account ID.
`CatalogItemIds`	The number of elements in `EntitlementIds`.
`CatalogNamespace`	Optional product namespace.

 - EOS returns the data you requested (and your void pointer) stored in `EOS_Ecom_OnQueryOwnershipCallback` .
 - This structure contains an array of `EOS_Ecom_EntitlementOwnership` users.
 - Items the server doesn't recognize are returned as `not owned`.

Endpoint

Use this endpoint:
```
https://ecommerceintegration-public-service-ecomprod02.ol.epicgames.com/ecommerceintegration/api/public/publickeys/{kid}
```
Set-up Call Request
The following code is what you use to make the call request:
```

EXAMPLE REQUEST CODE NEEDED PLEASE
EXAMPLE REQUEST CODE NEEDED PLEASE
EXAMPLE REQUEST CODE NEEDED PLEASE

```

Response

Below is the successful response:

```

    GET

    /ecommerceintegration/api/public/publickeys/
pbvnNIE97vErdePGIRoG41h8hnP_2wIxG8xbwZCIj3g HTTP/1.1

    Host: ecommerceintegration-public-service-
ecomprod02.ol.epicgames.com

    {

    "kty": "RSA",

    "e": "AQAB",

    "kid": "pbvnNIE97vErdePGIRoG41h8hnP_2wIxG8xbwZCIj3g",

    "n":
"gcStqtD8XD9c9ifNuxXT9Xd_EEZLLCw34yxINRQPt0MxEWkoOFsuisRWGktSFtGrnUuQn
p8GQY0k4Pyl_yDItWAcRtO7JUjrhQnxx3xXp_0P8xJMH1ny-
RcxHF3bEJWhDzNW5PBpBjQTQZis-83499z-4OlNA7oUnDKEJkqNfzh4mMDFluPxvW_Hwpa
w71nhzJI7-N-BdsPsLdqUANajLsFKq9fr06Lek_tm-6-
RUxNPE3yS0x0UIsGyapA4Apcczz0xTzRDfwOkq_TyKGZiZc7vtgjkWnqdsCyXZC7dzKJvg
0ggO3mKXhqZNNC_2pz24o1X_xCbG8rXtuvX8-ux-Q"
    }
```

Offline Method
> Note: the Offline Method is not recommended by EOS since the SDK doesn't function offline.

The offline method provides a signed token that the user verifies, or passes to a third-party service. When integrating with a third-party service for ownership verification, the offline method is recommended because it avoids granting the outside service access to the user's data.

To check ownership and cache the results locally, make a call to `EOS_Ecom_QueryOwnershipToken` with the following parameters:

Parameter	Description
`ApiVersion`	Set to `EOS_ECOM_QUERYOFFERS_API_LATEST`.
`LocalUserId`	The local user Epic account ID.

`CatalogItemIds`	The number of elements in `EntitlementIds`.
`CatalogItemIdCount`	The quantity of Catalog Items.
`CatalogNamespace`	Optional product namespace.

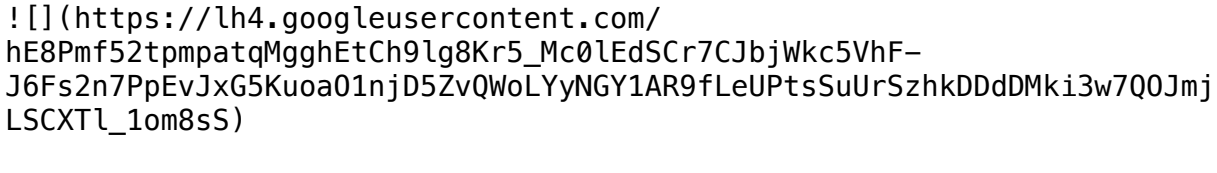
Upon success, you receive the `EOS_Ecom_QueryOwnershipTokenCallbackInfo` structure that includes a JSON Web Token (JWT) with a five-minute expiration time.

Verify the JWT with a public key and unpack it to extract the Key ID. Send the Key ID to third-party services if needed to verify the Entitlement information came from Epic Games Services (EGS).

Token Details

The Ownership Verification Token is a JWT signed using RS512 (RSA PKCS#1 signature with SHA-512, RSA key size 2048). The token contains the following claims:
Claim	Description
`jti`	A unique identifier for the token.
`sub`	The account ID that was used to request the token.
`clid`	The client ID used to request the token.
`ent`	An array of Entitlements that were verified for this token. If the value is empty, the account is not entitled to any of the requested Entitlements for a given sandboxId.
`iat`	The token expiration.

Below is the flow diagram:



Testing

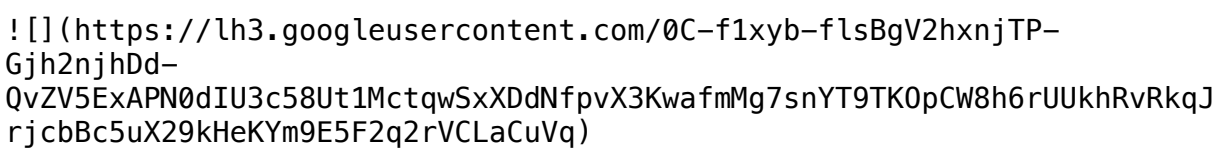
The Service Delivery team will create codes that grant redeemers a single `DevAudience` with access to all content for a particular title. This will allow most developers to gain access to the base game and all additional content.

-   Use your Epic Games account to log in to the Knowledge Base and refer to the article, [How to test purchase flow](https://epicsupport.force.com/epicgamesstoreknowledgebase/s/article/How-can-I-test-purchases-before-launch).

-   Additionally, Service Delivery can provide QA codes that have the base game and individual add-on content broken out to individual codes. This will allow test teams to validate the process of starting from the base game and adding additional content individually.

How Offers, Audiences, and Artifacts Work in the Game Store

The example below shows how Audiences, Artifacts, and Entitlements work in the Epic Games Store, starting with a user purchase.



Catalog Configuration

-   At this time, catalog configuration is performed by the EGS Service Delivery team. At some point in the future this will move to self-management via the Developer Portal.

-   Engage with your Service Delivery team through normal channels to set up Offers, Audiences, and Catalog Items.

Ownership vs. Entitlements

Topic	Ownership	Entitlements
User purchases Offer		When a Catalog Offer is purchased, an Entitlement is granted for each of the Items in the Offer to the purchasing user. An Entitlement record is unique for each user, per the Items granted.
Consumable		In the case of a Consumable, when a user makes the same purchase three times, they receive three Entitlement records, each with distinct Entitlement IDs. Think of Entitlements as receipts.
Durable		In the case of a Durable, the user receives an Entitlement for an Audience that is granted by the Offer. If a user purchases a Deluxe Edition of the game, the Deluxe Edition Audience Entitlement does NOT describe any other Items that are contained as part of the Deluxe Edition.
APIs	Because Entitlement queries only examine things there were directly acquired by users, the Ownership APIs and services were introduced. Ownership queries walk the full tree of Catalog Items to definitively answer whether or not a user should have access to a particular piece of content. For example, if a user purchases a deluxe

edition, which contains a season pass, which contains access to the first DLC, then an Ownership query will return TRUE on a query for the first DLC's Catalog Item. An Entitlement query will not say anything about the first DLC, only that the user purchased the deluxe edition.|
|
Ownership queries	Ownership queries are recommended for Durable Items. Ownership queries let developers know if a user should have access to a specific piece of content, whether they purchased it directly or received it through other means.	
Entitlement queries	Entitlement queries are recommended for Consumable Items. Entitlement queries return multiple unique instances of Consumable purchases so the game developer knows which ones to redeem/consume and how much to increment inventory to buy.	
Validate outstanding entitlement claims	Validate outstanding entitlement claims. Either the game's back end calls EOS directly (and the results are implicitly trusted) or the game client passes an entitlement token to the game's back end, and the token is validated using public keys.	
Award the user with in-game Items and mark the entitlements as redeemed	This option requires relating an outstanding entitlement to a specific in-game inventory item. This is done via the `entitlementName` or the `catalogItemId` field. These fields represent the immutable Offer or Item that the entitlement is tied to, versus the `entilementId`, which is the per-user / per-transaction unique ID. The game's back end stores a table which maps a particular `catalogItemId` to a specific set of in-game inventory Items.	

Award Example
Let's say `catalogItemId` `X` is a 100 coins offer and `catalogItemId` `Y` is a 500 coins offer. The game's back must call Redeem (either via [SDK API](https://dev.epicgames.com/docs/services/en-US/API/Members/Functions/Ecom/EOS_Ecom_RedeemEntitlements/index.html) or [REST](https://dev.epicgames.com/docs/services/en-US/Interfaces/Ecom/OwnershipVerificationREST/#redeem/consumeentitlements)) and, upon successful redemption, add the appropriate in-game inventory Items to the user's account.

 - Verifying Ownership/Entitlements In Epic Account Services (EAS), third-party applications require that you authorize consent for data access.

 - There are multiple methods for game developers to verify content the user has rights to, regardless of whether they are using Ownership or Entitlement queries.

Below are a few options for you to consider:
Option	Description
Use EOS APIs	The EOS SDK provides for doing local client verification of owned content.

|Using verification tokens |The EOS SDK provides APIs are very similar to the non-token versions of the APIs, except the results are stored in a verifiable token. These tokens can be cached locally on the user’s PC for a little while, or passed to a backend service for verification. A public key is used to inspect and verify the authenticity of the token, and by extension any ownership or entitlement contained therein can be trusted. |
|Service-to-service |Backend services can make the same API calls that game clients make, using the same SDK. Alternatively, in cases where the SDK is not appropriate, backends can make direct REST service calls to accomplish the same operations. Details on using REST are documented [here](https://dev.epicgames.com/docs/services/en-US/Interfaces/Ecom/OwnershipVerificationREST/index.html). |

Redeem Consumables

Multiple solutions exist for managing Consumable Entitlements, often depending on the mix of client logic versus back end logic. Regardless, the steps are generally the same:

Query Offer
You query for a list of Catalog Offers defined with Epic Online Services. This data is cached for a limited time and retrieved again from the backend when necessary.

Example
Let's walk through an example. We'll use these parameters to query a list of catalog Offers:
Parameter	Value to use	Description
`ApiVersion`	`EOS_ECOM_QUERYOFFERS_API_LATEST`	API version in the call.
`EOS_HEcom`	`XXXXXX`	EOS handle.
`Options`	`EOS_Ecom_QueryOffersOptions`	Pointer containing filter criteria.
`ClientData`	`XXXXXX`	Data sent to you in `CompletionDelegate`.
`CompletionDelegate`	`XXXXXX`	Callback after async completes.
`LocalUserId`	`XXXXXX`	The local user Epic account ID.
`CatalogItemIds`	`XXXXXX`	The number of elements in `EntitlementIds`.
`CatalogItemIdCount`	`XXXXXX`	The number of Catalog Items.

> The EOS SDK returns your callback (the `CompletionDelegate` parameter) regardless of success or error. Use the `void*` parameter to pass any contextual information the callback needs. Relevant information can be copied from the EOS SDK's cache while the callback function is running. Feel free to keep the callback function active as long as you like.

Make Call

So now that we have our call parameters, let's go ahead and make our call request. Use the request code below:
```

**EXAMPLE REQUEST CODE NEEDED PLEASE**
**EXAMPLE REQUEST CODE NEEDED PLEASE**
**EXAMPLE REQUEST CODE NEEDED PLEASE**

```
Example Response
The successful response code is below:

```

**EXAMPLE REQUEST CODE NEEDED PLEASE**
**EXAMPLE REQUEST CODE NEEDED PLEASE**
**EXAMPLE REQUEST CODE NEEDED PLEASE**

```

  ## REST Tips
For developers using REST the following tips are mighty useful:

-   A Consumable will always be `true` for all Consumable Items, regardless of the redemption state.

-   A status will change from `active` pre-redemption to `redeemed` post-redemption.

- The `active` status will change from `true` pre-redemption to `false` post-redemption.