

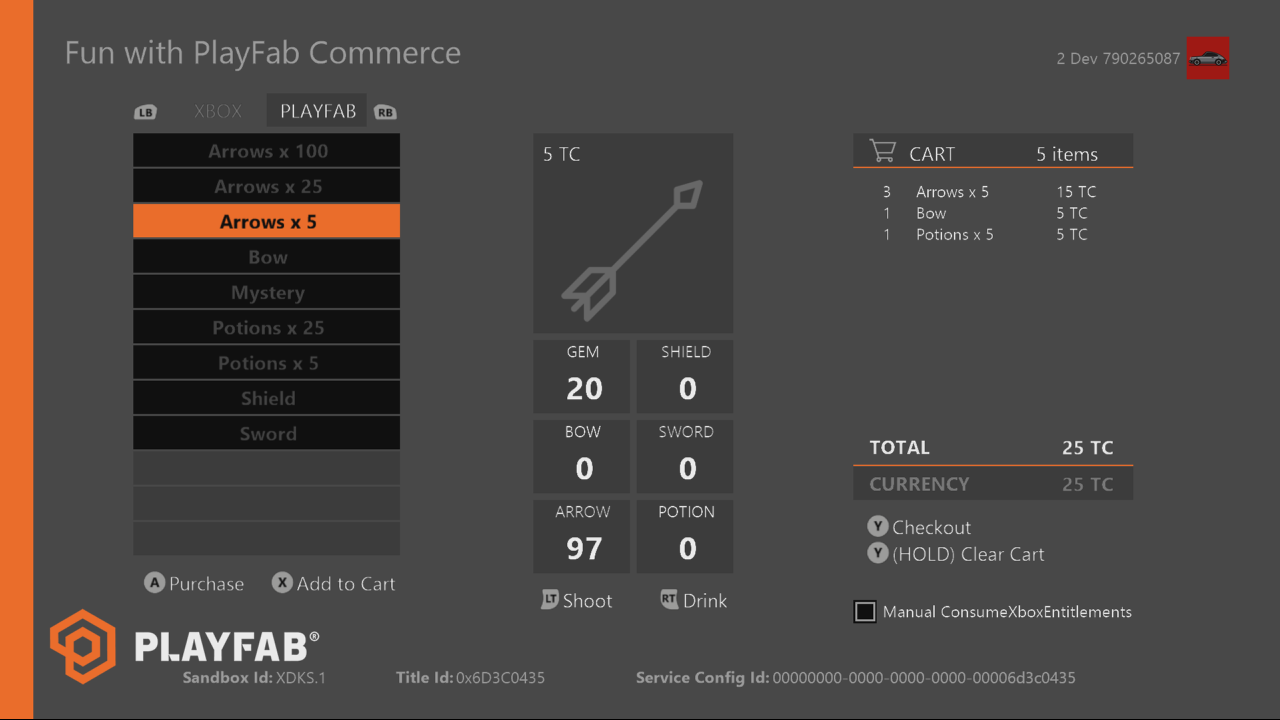
PlayFab Commerce

*\* This sample is compatible with the June 2018 XDK*

# Description

This sample demonstrates how to interact with both the Microsoft Store and PlayFab Economy to implement an in-game catalog of virtual items and currency that is mediated by PlayFab services.

# Using the sample

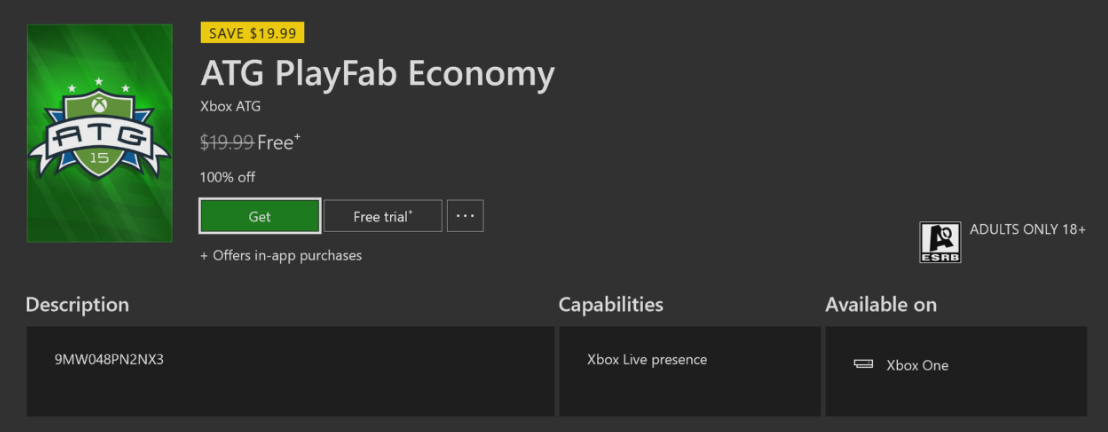


|  |  |
| --- | --- |
| Action | Gamepad |
| Toggle item category | Left/Right bumper |
| Purchase item | A button |
| Add to Cart (PlayFab items only) | X button |
| Checkout cart (when not empty) | Y button |
| Empty cart (when not empty) | Hold Y button |
| Consume arrow | Left trigger |
| Consume potion | Right trigger |
| Manually call ConsumeXboxEntitlements | Click Left Stick |
| Toggle debug output console | View button |

This sample is configured to work in the XDKS.1 sandbox. Before it can be run, the test account must be entitled for the published product.

To launch the sample, do one of the following on the devkit:

* Search for ATG PlayFab Economy in the Store, or
* From Xbox One XDK Command Prompt: xbapp launch ms-windows-store://pdp/?productid=9NKMD8HL90BN



Select Get to obtain the license; there is no need to allow the download to complete.

Once launched, there are two categories of items. The default category shows the items that are configured in the PlayFab catalog in GameManager. Each of these items have a cost in TC, the virtual currency configured for this title. The other category offers products configured for real money purchase and set up in Partner Center, including quantities of TC.

Virtual (PlayFab) items can be purchased individually but also added to a cart to checkout in a single transaction.

See [Implementation Notes](#_API_usage) for an explanation of the “Manual ConsumeXboxEntitlements” toggle.

**Note**: [Partner Center](https://partner.microsoft.com/) was formerly known as [Dev Center](https://developer.microsoft.com/) or UDC

# Implementation notes

The PlayFab title ID must be set in the sample initialization code. This sample uses the PlayFab title ID “4E29” which is assigned to the **PlayFabSettings::titleId** static field. This title ID is configured in Partner Center with full tier Xbox Live services despite the title not interfacing with any Xbox Live client API. This is needed to configure Single Sign On so it can use the PlayFab service as the relying party. The relying party business partner certificate is specific to the PlayFab title, so you must contact PlayFab developer relations ([devrel@playfab.com](mailto:devrel@playfab.com)) to configure the certificate.

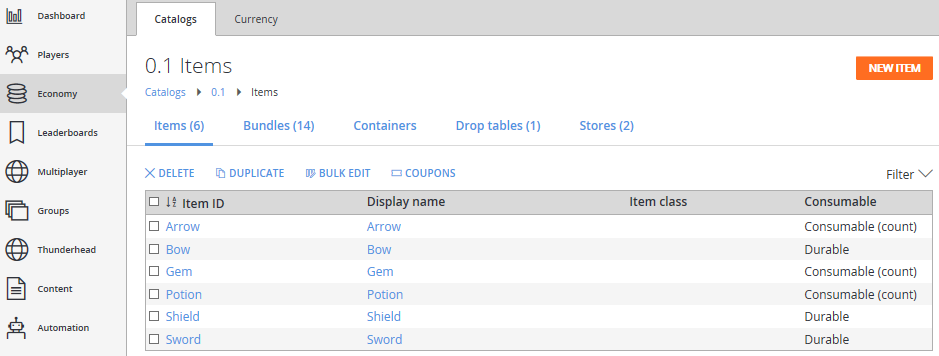
This sample also uses the Windows::Services::Store API to access store features, which is recommended for products natively set up in Partner Center. As such, it requires the account to be entitled for launch as described above. For debugging from VS, it also requires the debug\_licensing\_overrides.xml file. This is already included and configured with the required IDs. More information on the override can be found in the XDK documentation under [*Testing with a License Override*](https://developer.microsoft.com/en-us/games/xbox/docs/xdk/testing-license-override).

## PlayFab items configuration

See the official PlayFab documentation for full details of how to use GameManager to configure items. This is done under the **Economy** section, which contains several sub-sections, including Currency, Items, Bundles, and Stores.

Currency is simply the virtual currencies that can be purchased, earned, or granted by the title. For simplicity, only one currency, TC, is used throughout the sample.

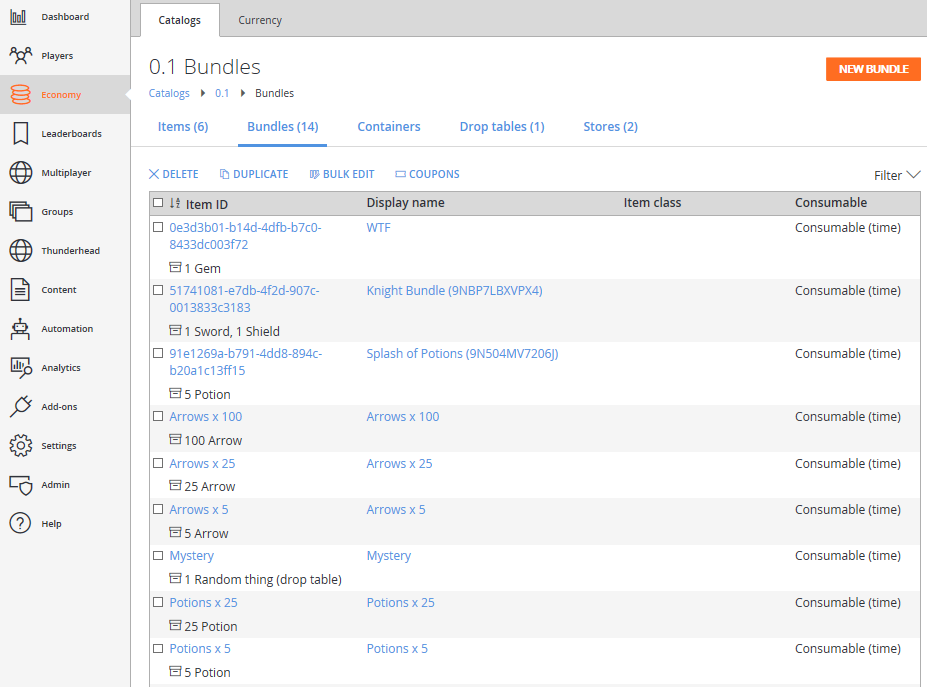
Items are the primitives that can be granted, and these can either be consumable or durable.



This sample uses consumables which are intended to be decremented upon use, as well as permanent durable items. Distinctions between these items and Microsoft products (as configurable in Partner Center) are as follows:

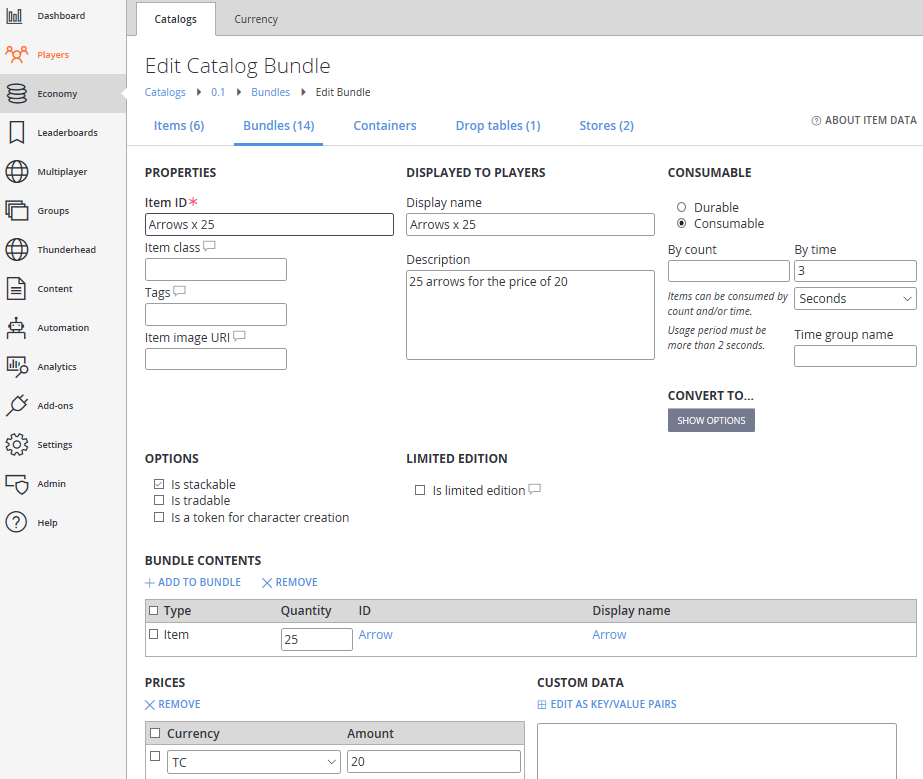
* Consumables can also be time limited (set to expire after a certain amount of time).
* Items are returned in inventory as separate instances. They can be set to be stackable if this is not desired; however, be wary when creating bundles with large amounts of individual items.

The items listed in the PlayFab category in the sample are all consumable bundles set up under the Bundles tab (as shown in the following figure). These bundles consist of various amounts and combinations of items and virtual currencies.



While the GetUserInventory API returns all items and bundles, the sample filters for bundle consumable items that contain a non-zero number of bundled items (see the Quantity field under Bundle Contents in the following figure).

Note also that these bundles are consumables themselves, but instead of being of Count type, they are set to expire after 3 seconds. This is so that the bundle instances that are purchased do not also return in the inventory as it is most likely that only the cumulative amounts of items are interesting to see in the inventory.



The other bundle items with the GUID identifiers are how PlayFab bridges Microsoft inventory items purchased with real money and the virtual items configured in GameManager. This will be described in the API section below.

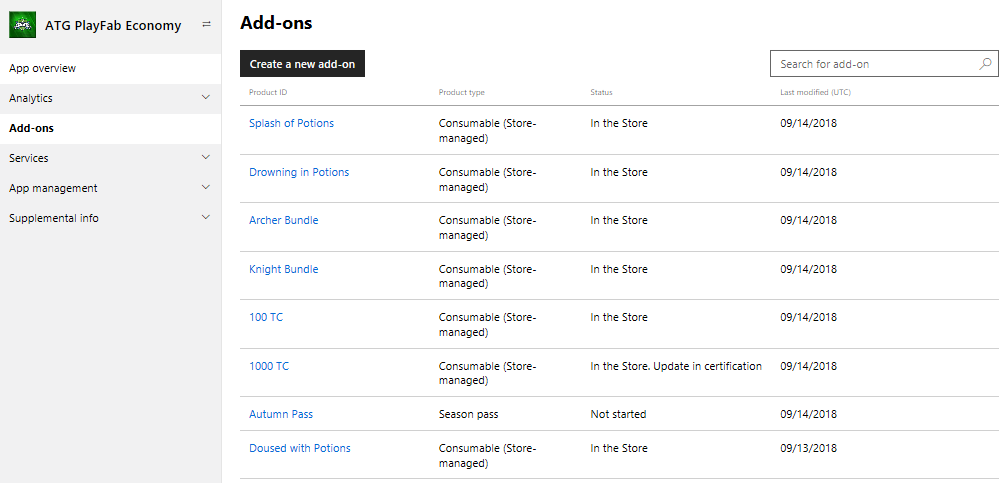
Other aspects employed by this sample’s catalog include:

* Multiple Stores that switch prices based on player segment. In this sample, one segment is defined such that if the user has greater than 400TC, prices of some items will be discounted.
* An item that is configured with a Drop Table that will grant a random item based on defined percentages.

Again, please see the PlayFab documentation for more detailed treatment of these and other aspects of Economy.

## Microsoft Items configuration

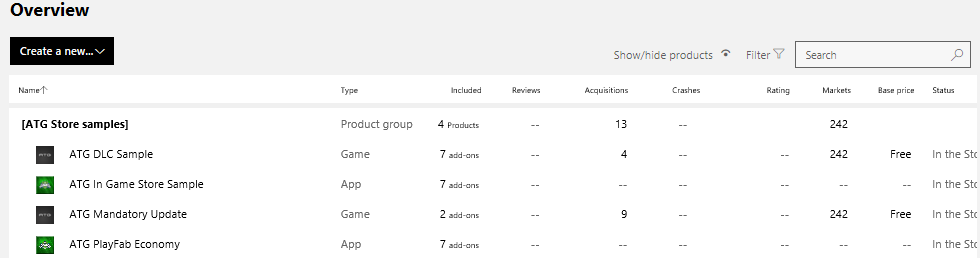
Each of the bundle items with GUIDs have a 1:1 corresponding item configured in Partner Center.



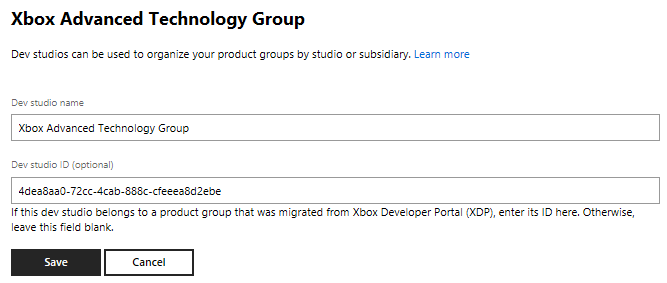
These are otherwise unremarkable consumables which can be set with the desired attributes. They must be store-managed to allow the b2b consume API to function. It is not necessary for these consumables to be set with the visibility that has its own Store Listing. It is highly recommended to set the granted quantity for each of these to be 1 (see below).

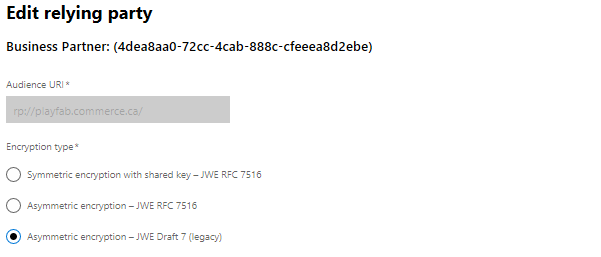
**Important!** For the key b2b aspect of the sample to work, several aspects must be configured properly:

1. The Partner Center base game product must live in a product group:



This product group must be associated with a Dev Studio which in turn needs to be associated with a Dev Studio ID that matches the Relying Party Business Partner ID.





Configuration of relying party, web services, and full tier Xbox Live is beyond the scope of this ReadMe. Please reach out to account managers and the forums for assistance on these aspects.

1. Each product intended to be consumed b2b by ConsumableXboxEntitlements (see below) must be set to be associated with a legacy Xbox product ID, which is a GUID. This can only be done by account managers who will need to reach out to the store ingestion team (SMP > Addon > Details > Legacy XBox Linkage Kind: InventoryCallOnly).

## API usage

The following documentation provides more detailed information about the APIs used in this sample:

Windows Store API: <https://docs.microsoft.com/en-us/uwp/api/Windows.Services.Store.StoreContext>

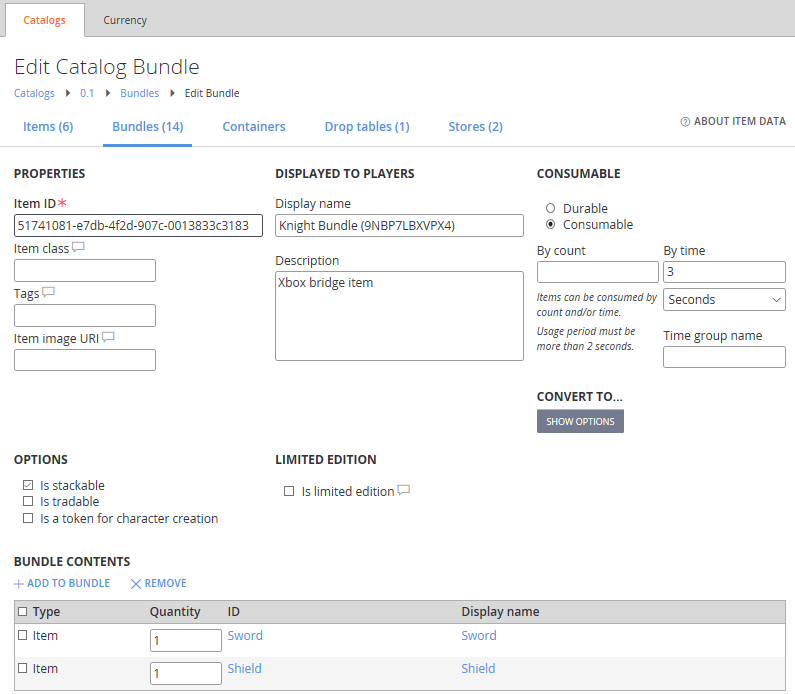
PlayFab API: <https://api.playfab.com/documentation/client>

The sample roughly divides operations in these areas into two source files: StoreOps.cpp and PlayFabOps.cpp respectively.

**ConsumeXboxEntitlements** is the key API that the title must call to reconcile real money purchases of Microsoft Products with corresponding PlayFab items. When this API is called, the PlayFab service will make a b2b call to Microsoft’s inventory service to query any owned consumables with non-zero quantity. It is essential for the b2b configuration steps in the previous section to be followed for this to work at all.

Due to the usage of the Microsoft inventory.xboxlive.com service, PlayFab will see products returned with legacy Xbox Product IDs. These items will be compared with any bundles that are configured in the PlayFab catalog, which is why there are items in a previous figure configured with GUID identifiers.

For every Microsoft inventory consumable that matches with a PlayFab bundle, the outstanding quantity is consumed one at a time, and one instance of the corresponding PlayFab bundle is granted. Like with the purchasable bundles, these bundles are also set to be time-limited consumables, with the bundle contents defining what is granted by the original real money purchase.



It is ultimately up to the developer to ensure that the description of the Partner Center product matches what is actually granted in terms of the in-game items.

**Note:** Microsoft consumable quantities are consumed by 1 until exhausted, so it is very important to not create consumables that give large quantities, as these will generate heavy traffic between Microsoft and PlayFab services that will get the title throttled or banned.

Observe that in normal operation, ConsumeXboxEntitlements is called upon success of the RequestPurchaseAsync of a Microsoft product. The “Manual ConsumeXboxEntitlements” toggle will decouple this so that the RequestPurchaseAsync will result in a quantity increase of the Microsoft consumable, and then **clicking Left Stick** will manually call the ConsumeXboxEntitlements so that the transition from Microsoft quantity to PlayFab inventory can be observed.

# Best practices

The following is a summary of the key gotchas mentioned in this document:

* Partner Center products must be under a product group, the product group must be associated with a Dev Studio with a Dev Studio ID matching the relying party business partner ID.
* Create PlayFab bundles as time-limited consumables with 3-5 second expiry if the quantity of bundle instances is not important for user inventory.
* Microsoft consumables that are intended to be linked with PlayFab bundles should only grant quantity 1 for each purchase, as outstanding quantity is consumed one-at-a-time by the PlayFab service.
* PlayFab bundles linked to Microsoft products require Item IDs set to Xbox product IDs; to find out what these IDs are for products configured in Partner Center, contact account manager so they can check them in SMP (internal tool).
* Microsoft consumables intended for linking must have their Legacy XBox Linkage Kind set to InventoryCallOnly. This can also be seen or set in SMP.

# Update history

**Initial Release:** October 2018

# Privacy Statement

When compiling and running a sample, the file name of the sample executable will be sent to Microsoft to help track sample usage. To opt-out of this data collection, you can remove the block of code in Main.cpp labeled “Sample Usage Telemetry”.

For more information about Microsoft’s privacy policies in general, see the [Microsoft Privacy Statement](https://privacy.microsoft.com/en-us/privacystatement/).