braincloud

Upgrading to brainCloud from PlayFab

Economy and Catalogs

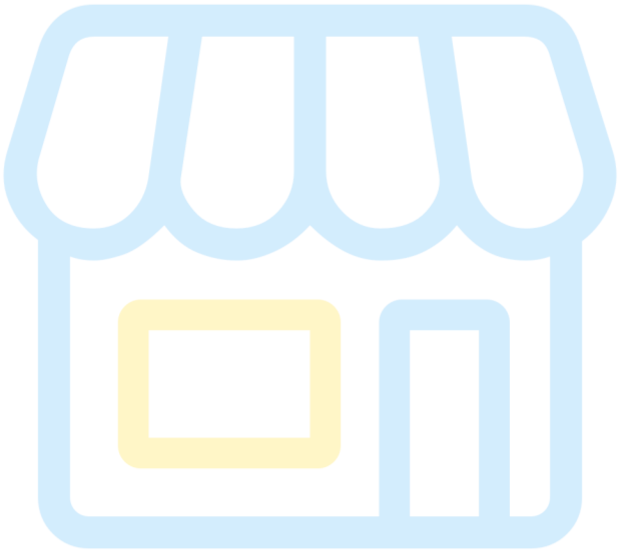


Table of Contents

Introduction

Both PlayFab and brainCloud offer extensive features for Items, Bundles, In-App Purchases (IAPs), and Virtual Currency. PlayFab groups these under the "Economy" section, which includes both v2 and legacy catalogs. The new catalog system differs significantly from the legacy system, though they share some fundamental features.

This guide examines how brainCloud's features compare to both v2 and legacy PlayFab features required for migration, and how brainCloud features can be tailored to suit your game's specific needs.

Note

brainCloud's Virtual Currency is covered in its own guide <link>, as it can be used with other brainCloud features such as rewards and level progression.

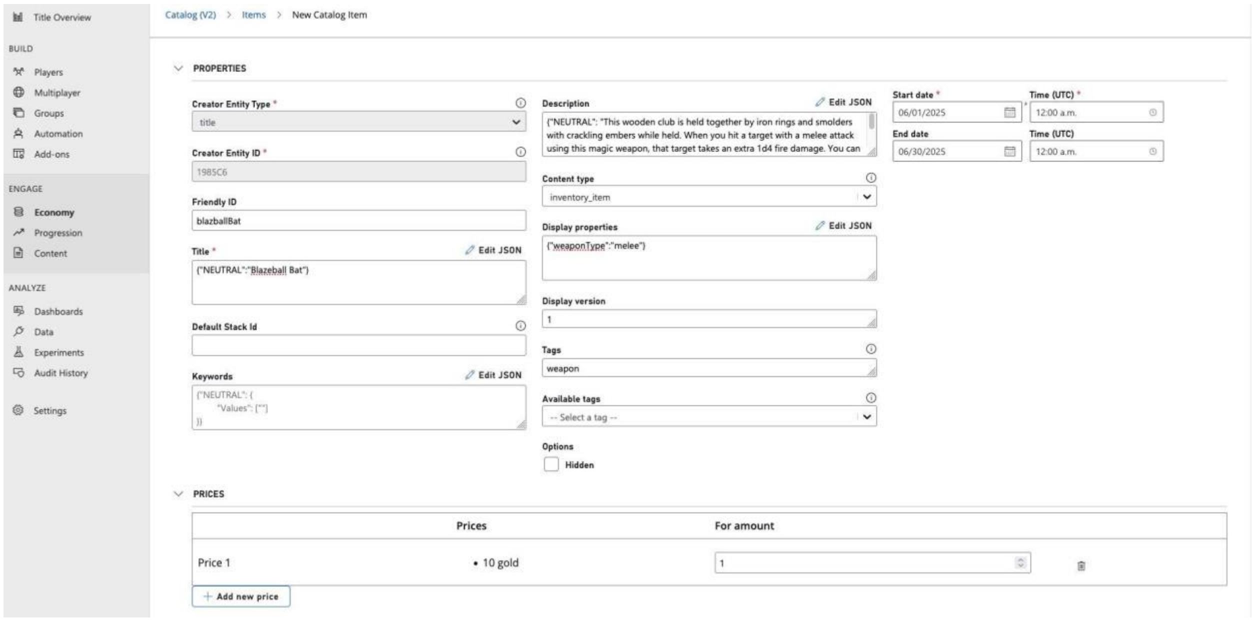
Items & Bundles

The first step is to recreate your PlayFab Items and Bundles in brainCloud. Advance to the Item Catalog: **App > Design > Marketplace > Item Catalog**. The Item Catalog contains the collection of items that players can acquire through rewards, in-game events, or purchase. Of particular note, Bundles can include quantities of Items, *and quantities of virtual currency*. A Virtual Currency Price can be set for Items and Bundles (or limit an Item to Real-Money sale only). For the purpose of cash buys, in brainCloud the operator will attach Items and Bundles to **Products (**for real-money purchase). Items in the catalog are configurable with properties like virtual currency price, virtual currency buyback price, description, category, tags and more.

Due to the similarities between brainCloud and PlayFab bundles and items, we provide an API comparison table later in this guide for quick reference.

Creating Items

Playfab Catalog v2 Items have a standard set of definition parameters common to all Catalog v2 features.


Catalog (V2) > Items > New Catalog Item

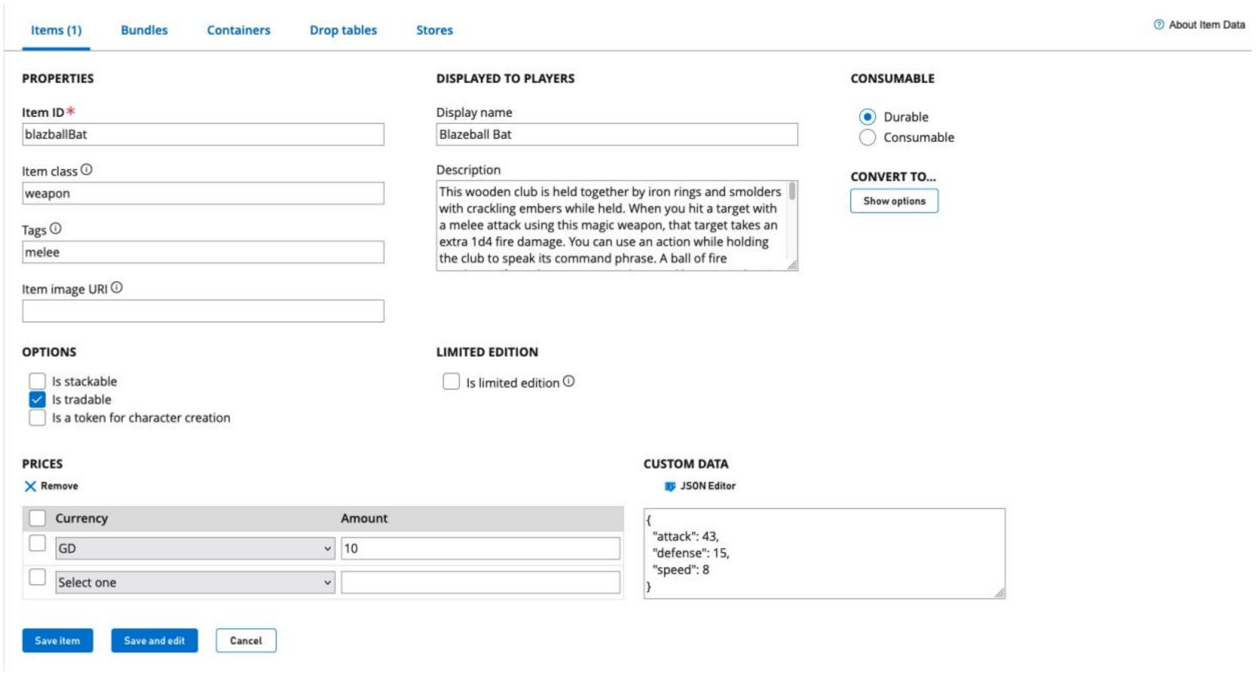
PROPERTIES

Creator Entity Type *	title
Creator Entity ID *	1985C6
Friendly ID	blazballBat
Title *	{"NEUTRAL":"Blazeball Bat"}
Default Stack Id	
Keywords	{"NEUTRAL": { "Values": [""] }}
Description	{"NEUTRAL": "This wooden club is held together by iron rings and smolders with crackling embers while held. When you hit a target with a melee attack using this magic weapon, that target takes an extra 1d4 fire damage. You can
Content type	inventory_item
Display properties	{"weaponType":"melee"}
Display version	1
Tags	weapon
Available tags	-- Select a tag --
Options	☐ Hidden
Start date *	06/01/2025
Time (UTC) *	12:00 a.m.
End date	06/30/2025
Time (UTC)	12:00 a.m.

PRICES

	Prices	For amount
Price 1	• 10 gold	1

Playfab Legacy catalog items share most options with v2 catalog items but offer extra functionality for durable, consumable, tradable, and stackable items. Legacy items are more similar to brainCloud items, as we will see.


Items (1) | Bundles | Containers | Drop tables | Stores

PROPERTIES

Item ID*	blazballBat
Item class	weapon
Tags	melee
Item image URI	

DISPLAYED TO PLAYERS

Display name	Blazeball Bat
Description	This wooden club is held together by iron rings and smolders with crackling embers while held. When you hit a target with a melee attack using this magic weapon, that target takes an extra 1d4 fire damage. You can use an action while holding the club to speak its command phrase. A ball of fire

CONSUMABLE

● Durable
○ Consumable

CONVERT TO...

OPTIONS

☐ Is stackable
☑ Is tradable
☐ Is a token for character creation

LIMITED EDITION

☐ Is limited edition

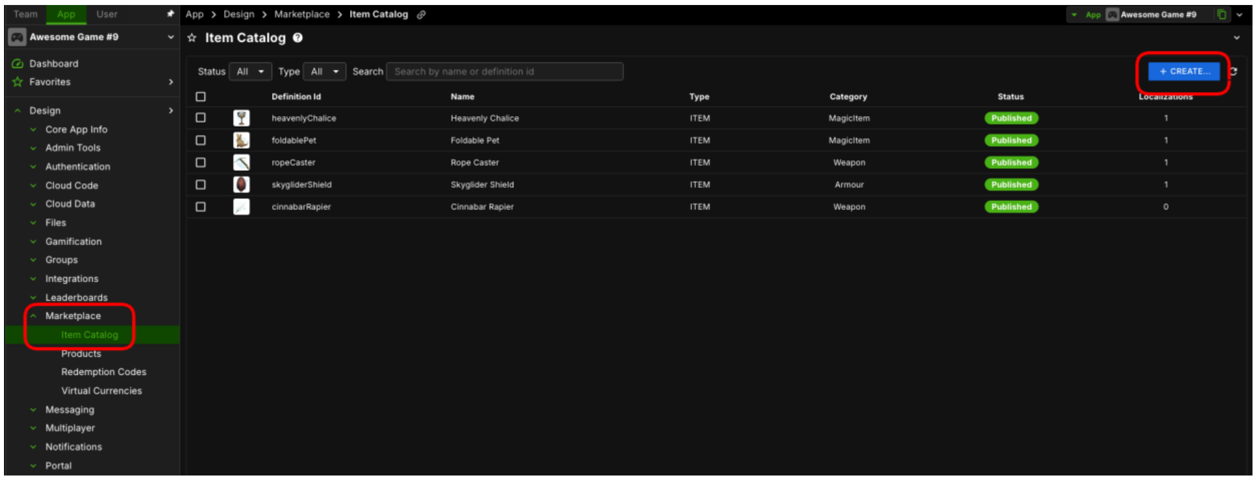
PRICES

Currency	Amount
GD	10
Select one	

CUSTOM DATA

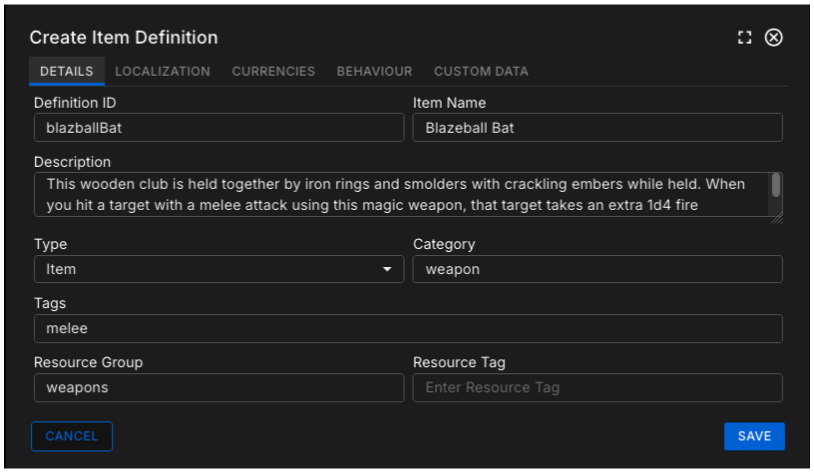
```
{
 "attack": 43,
 "defense": 15,
 "speed": 8
}
```

Let's look at creating the item above with brainCloud. The Item Catalog is found in the **Design → Marketplace → Item Catalog** menu. Click the **Create** button in the top left corner of the dashboard to add a new item. You will be presented with the option to create an **Item** or a **Bundle**. Select Item for now.



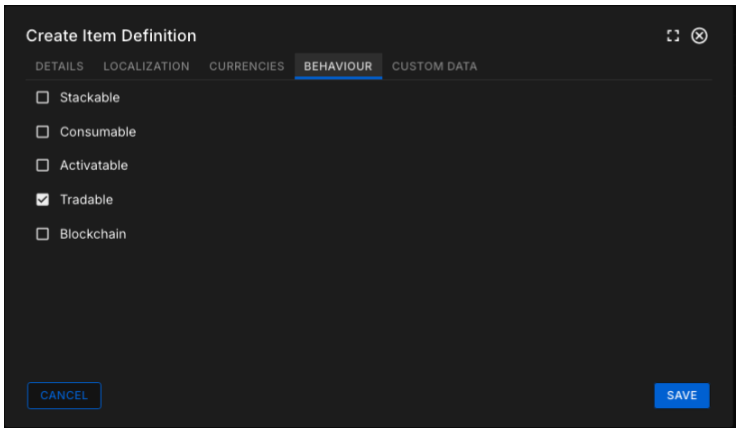
	Definition Id	Name	Type	Category	Status	Localizations
	heavenlyChalice	Heavenly Chalice	ITEM	MagicItem	Published	1
	foldablePet	Foldable Pet	ITEM	MagicItem	Published	1
	ropeCaster	Rope Caster	ITEM	Weapon	Published	1
	skygliderShield	Skyglider Shield	ITEM	Armour	Published	1
	cinnabarRapier	Cinnabar Rapier	ITEM	Weapon	Published	0

You can fill in the item's basic details such as ID, name, description, and tags here. There are multiple tabs that allow you to add virtual price and extra custom data, which can be used for adding stats to your items for example.



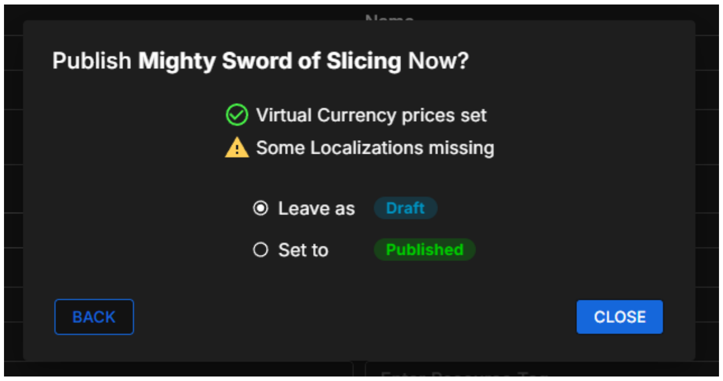
Definition ID: blazballBat	Item Name: Blazeball Bat
Description: This wooden club is held together by iron rings and smolders with crackling embers while held. When you hit a target with a melee attack using this magic weapon, that target takes an extra 1d4 fire	
Type: Item	Category: weapon
Tags: melee	
Resource Group: weapons	Resource Tag: Enter Resource Tag

The behavior tab allows you to configure how an item functions in a player's inventory or if it is tradable between players.



- [ ] Stackable
- [ ] Consumable
- [ ] Activatable
- [✓] Tradable
- [ ] Blockchain

Once you save the item, you will be prompted to choose if you want to Publish it. Draft items are hidden from players until you Publish. This allows you to work on it across multiple sessions until you are ready.



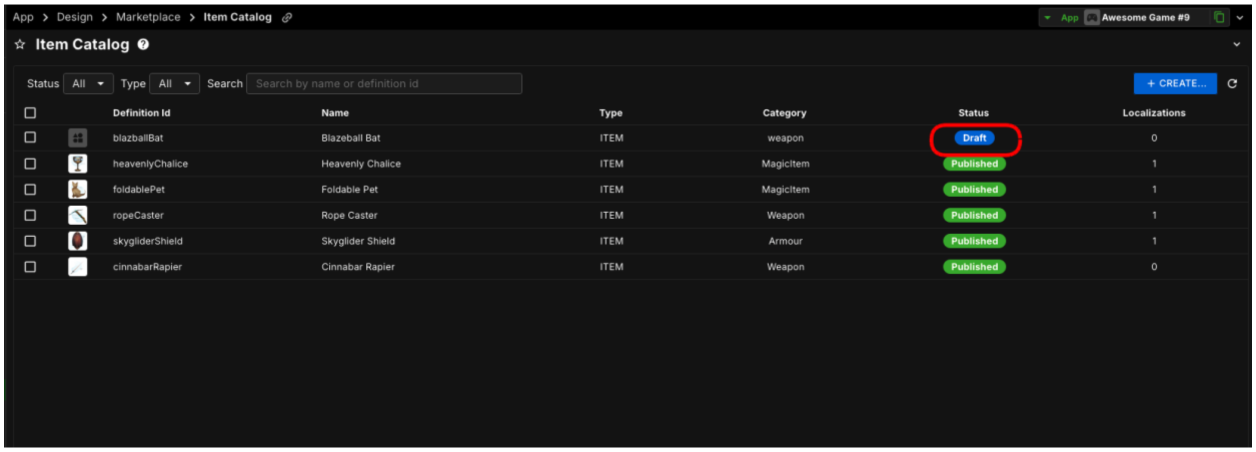
Publish **Mighty Sword of Slicing** Now?

✓ Virtual Currency prices set
⚠ Some Localizations missing

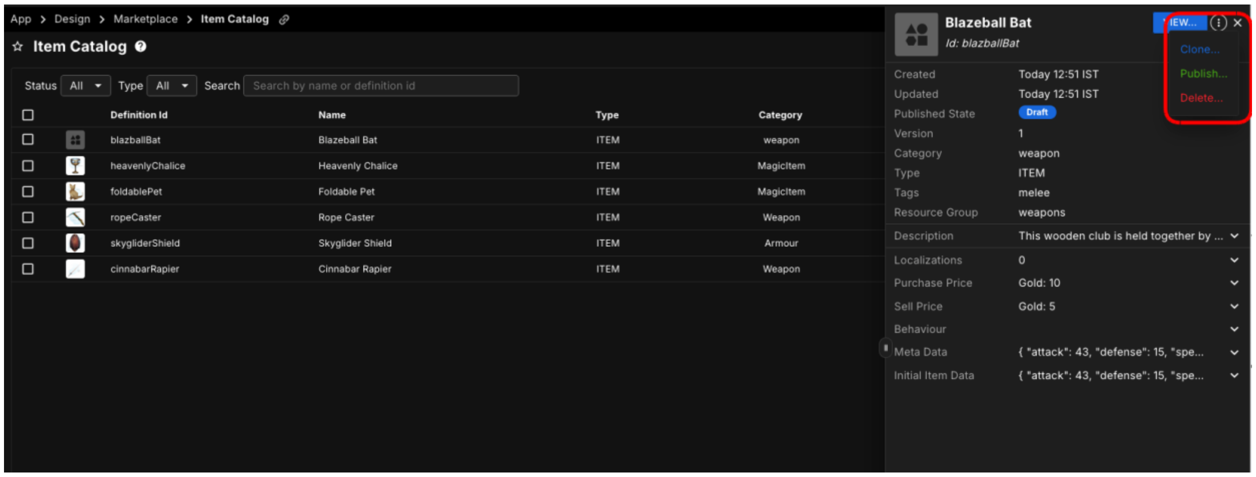
[✓ Leave as Draft] Set to Published

Publishing Later…

Items will appear in the catalog list. You can publish an item by clicking on it and then selecting the “Publish…” option in the panel's menu.


App > Design > Marketplace > Item Catalog
App Awesome Game #9
Item Catalog
Status All Type All Search Search by name or definition id
+ CREATE...

Definition Id	Name	Type	Category	Status	Localizations
blazballBat	Blazeball Bat	ITEM	weapon	Draft	0
heavenlyChalice	Heavenly Chalice	ITEM	MagicItem	Published	1
foldablePet	Foldable Pet	ITEM	MagicItem	Published	1
ropeCaster	Rope Caster	ITEM	Weapon	Published	1
skygliderShield	Skyglider Shield	ITEM	Armour	Published	1
cinnabarRapier	Cinnabar Rapier	ITEM	Weapon	Published	0


App > Design > Marketplace > Item Catalog
Item Catalog
Status All Type All Search Search by name or definition id
VIEW...
Clone...
Publish...
Delete...

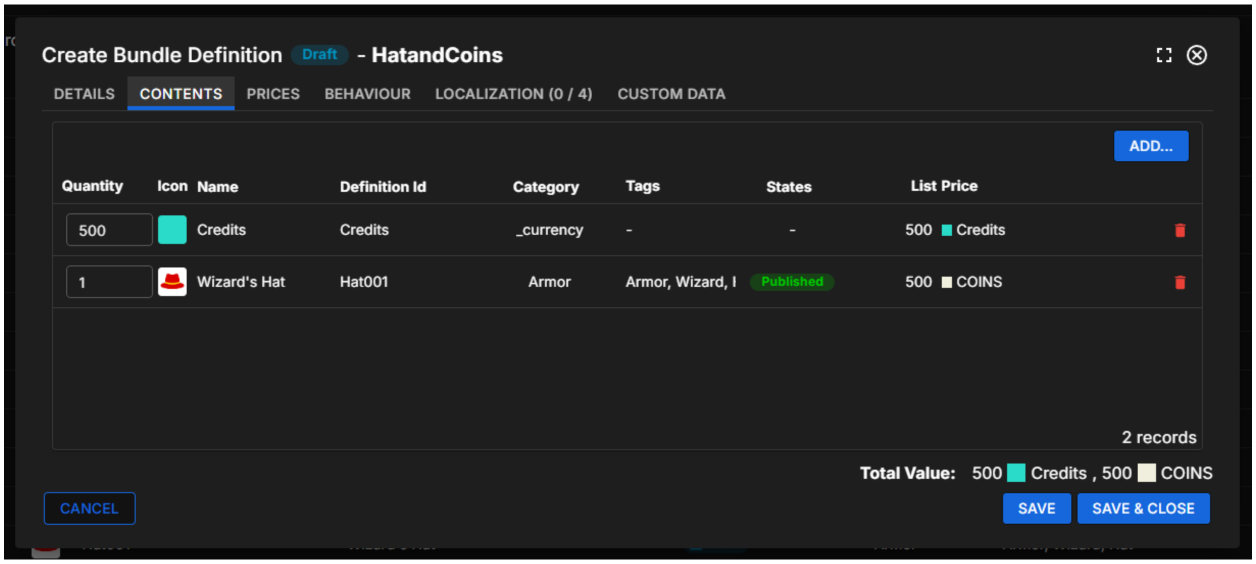
Definition Id	Name	Type	Category
blazballBat	Blazeball Bat	ITEM	weapon
heavenlyChalice	Heavenly Chalice	ITEM	MagicItem
foldablePet	Foldable Pet	ITEM	MagicItem
ropeCaster	Rope Caster	ITEM	Weapon
skygliderShield	Skyglider Shield	ITEM	Armour
cinnabarRapier	Cinnabar Rapier	ITEM	Weapon

Blazeball Bat
Id: blazballBat

Created	Today 12:51 IST
Updated	Today 12:51 IST
Published State	Draft
Version	1
Category	weapon
Type	ITEM
Tags	melee
Resource Group	weapons
Description	This wooden club is held together by ...
Localizations	0
Purchase Price	Gold: 10
Sell Price	Gold: 5
Behaviour	
Meta Data	{ "attack": 43, "defense": 15, "spe...
Initial Item Data	{ "attack": 43, "defense": 15, "spe...

Bundles

Bundles are simply containers of Items and Virtual Currency. Just like Items, a buy price in Virtual Currency may be set, and the Bundle can also be attached to a Cash Product. It is also allowed for a Bundle to contain *only* Virtual Currency, this is a good way to set up Virtual Currency swaps in your game.



Create Bundle Definition Draft - **HatandCoins**

Quantity	Icon	Name	Definition Id	Category	Tags	States	List Price
500	🟦	Credits	Credits	_currency	-	-	500 🟦 Credits
1		Wizard's Hat	Hat001	Armor	Armor, Wizard, I	Published	500 ⬜ COINS

2 records

Total Value: 500 🟦 Credits , 500 ⬜ COINS

Note that by default, Bundles will filter the Item picker to only Published items as a convenience to you.

Stores, Products & Item Lists

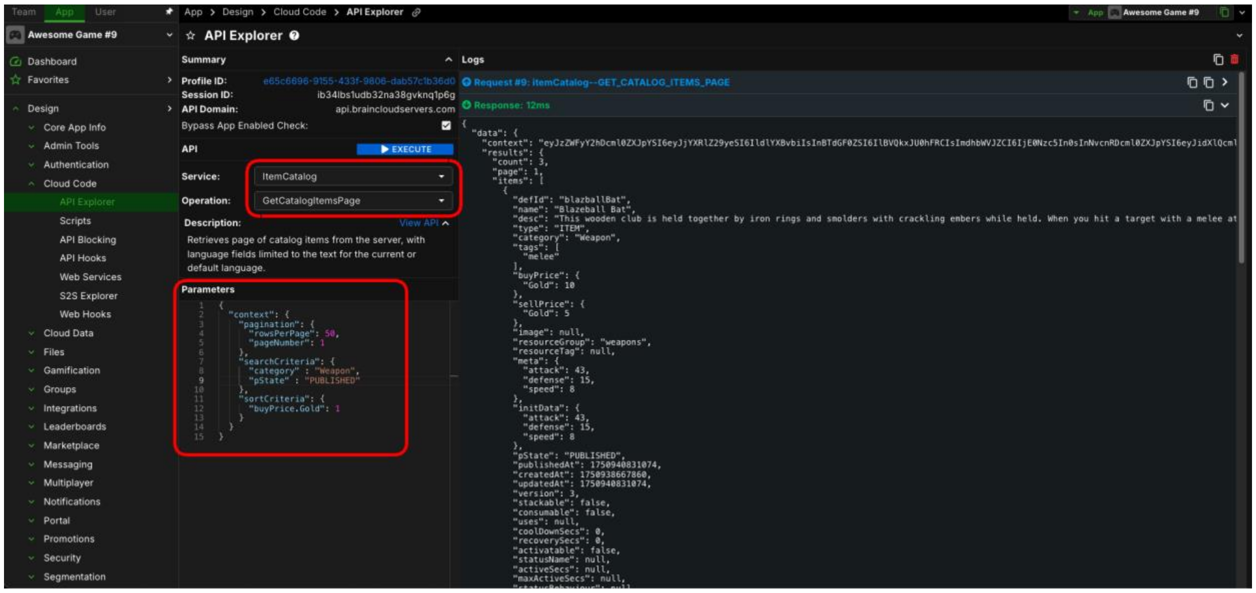
While Items and Bundles are similar between PlayFab and brainCloud, the acquisition methods differ. Items can be granted to players using APIs similar to both platforms. However, presenting players with information on stores and item lists varies, as it also differs between PlayFab's v2 and legacy catalog systems.

First, let's look at listing items for a storefront or NPC store. These stores list items that are purchasable using soft-currency, not from a third-party storefront.

Listing Items

To list or filter items, use the GetCatalogItemsPage request. This is similar to the GetStoreItems legacy request or the SearchItems v2 request with PlayFab. This request allows searches using a custom query, providing flexibility for listing specific categories or item states.

We'll use the API Explorer to quickly demonstrate filtering all published weapons in the catalog, sorted by price. Find more information about the API Explorer in the Cloud Scripts Document.



Summary

Profile ID: e65c6696-9155-433f-9806-dab57c1b36d0
Session ID: ib34lbs1udb32na38gvknq1p6g
API Domain: api.braincloudservers.com
Bypass App Enabled Check: ✓

API

Service: ItemCatalog
Operation: GetCatalogItemsPage

Description:
Retrieves page of catalog items from the server, with language fields limited to the text for the current or default language.

Parameters

```
{
  "context": {
    "pagination": {
      "rowsPerPage": 50,
      "pageNumber": 1
    },
    "searchCriteria": {
      "category" : "Weapon",
      "pState" : "PUBLISHED"
    },
    "sortCriteria": {
      "buyPrice.Gold": 1
    }
  }
}
```

Request #9: itemCatalog--GET_CATALOG_ITEMS_PAGE

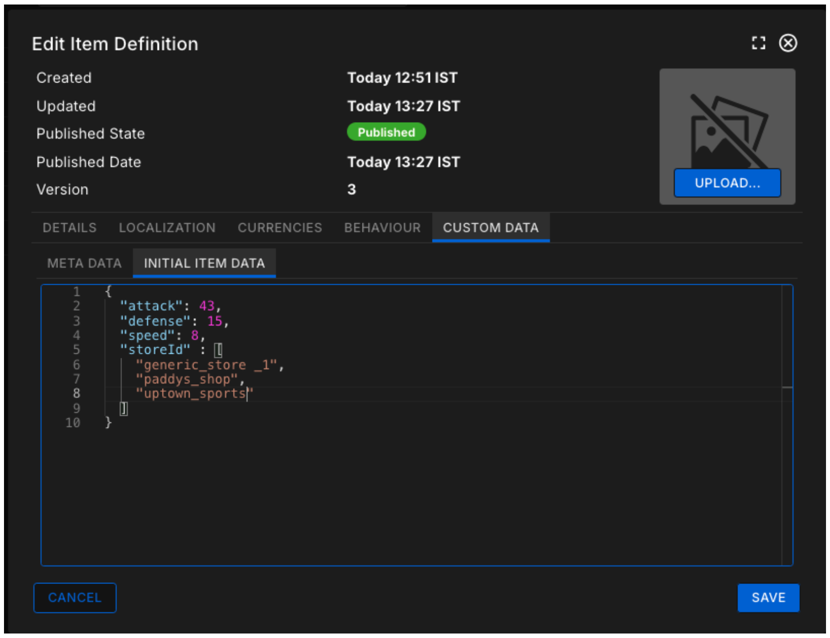
Response: 12ms

```
{
  "data": {
    "context": "eyJzZWFyY2hDcml0ZXJpYSI6eyJjYXRlZ29yeSI6IldlYXBvbiIsInBTdGF0ZSI6IlBVQkxJU0hFRCIsImdhbWVJZCI6IjE0Nzc5In0sInNvcnRDcml0ZXJpYSI6eyJidXlQcml
    "results": {
      "count": 3,
      "page": 1,
      "items": [
        {
          "defId": "blazballBat",
          "name": "Blazeball Bat",
          "desc": "This wooden club is held together by iron rings and smolders with crackling embers while held. When you hit a target with a melee at
          "type": "ITEM",
          "category": "Weapon",
          "tags": [
            "melee"
          ],
          "buyPrice": {
            "Gold": 10
          },
          "sellPrice": {
            "Gold": 5
          },
          "image": null,
          "resourceGroup": "weapons",
          "resourceTag": null,
          "meta": {
            "attack": 43,
            "defense": 15,
            "speed": 8
          },
          "initData": {
            "attack": 43,
            "defense": 15,
            "speed": 8
          },
          "pState": "PUBLISHED",
          "publishedAt": 1750940831074,
          "createdAt": 1750938667860,
          "updatedAt": 1750940831074,
          "version": 3,
          "stackable": false,
          "consumable": false,
          "uses": null,
          "coolDownSecs": 0,
          "recoverySecs": 0,
          "activatable": false,
          "statusName": null,
          "activeSecs": null,
          "maxActiveSecs": null,
```

Listing Store Items

In the response above, you'll see detailed information about the returned items, and any item field can be used as a query for filtering and listing catalog items.

While brainCloud doesn't have a "stores" concept like PlayFab's legacy API, you can achieve this in several ways. The simplest method is to add a store ID to the "Initial Item Data" of each item. Using an array allows an item to be present in multiple stores simultaneously.

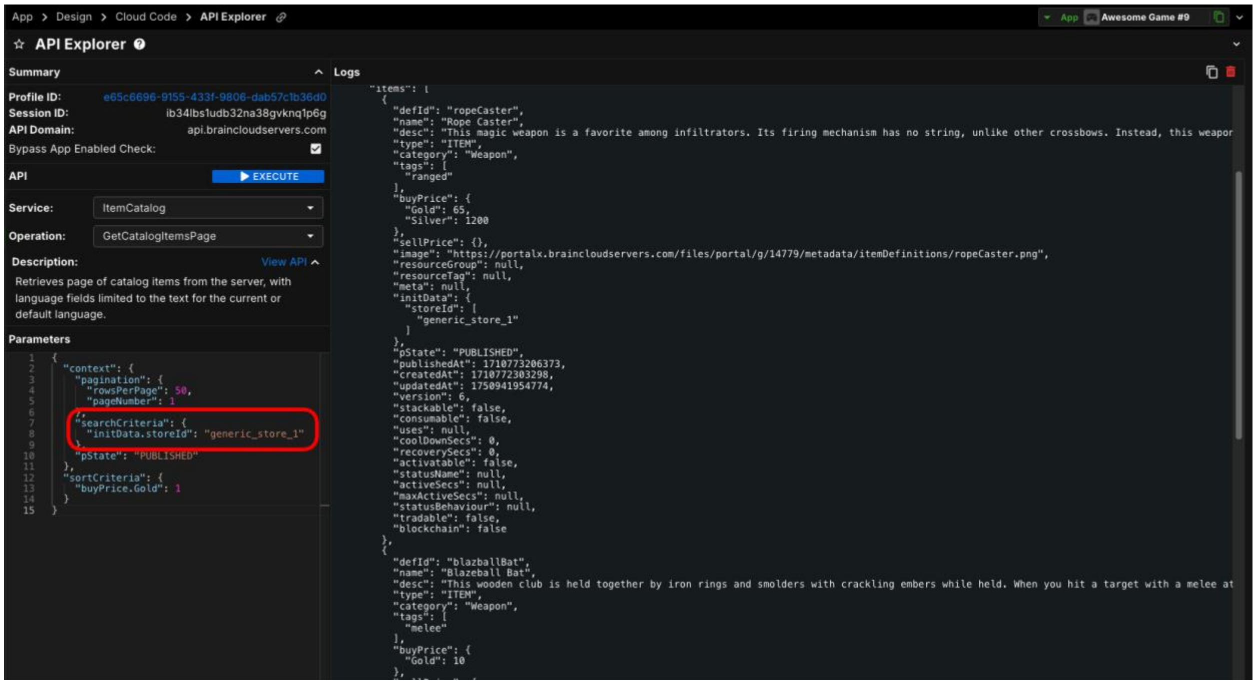


Edit Item Definition

Created	Today 12:51 IST
Updated	Today 13:27 IST
Published State	Published
Published Date	Today 13:27 IST
Version	3

```
{
  "attack": 43,
  "defense": 15,
  "speed": 8,
  "storeId" : [
    "generic_store _1",
    "paddys_shop",
    "uptown_sports"
  ]
}
```

And then we can query the **initData.storeId** array in the **GetCatalogItemsPage** request.



Profile ID: e65c6696-9155-433f-9806-dab57c1b36d0
Session ID: ib34lbs1udb32na38gvknq1p6g
API Domain: api.braincloudservers.com
Bypass App Enabled Check: ✓

Service: ItemCatalog
Operation: GetCatalogItemsPage
Description: Retrieves page of catalog items from the server, with language fields limited to the text for the current or default language.

```
{
  "context": {
    "pagination": {
      "rowsPerPage": 50,
      "pageNumber": 1
    },
    "searchCriteria": {
      "initData.storeId": "generic_store_1"
    },
    "pState": "PUBLISHED"
  },
  "sortCriteria": {
    "buyPrice.Gold": 1
  }
}
```

```
      "items": [
        {
          "defId": "ropeCaster",
          "name": "Rope Caster",
          "desc": "This magic weapon is a favorite among infiltrators. Its firing mechanism has no string, unlike other crossbows. Instead, this weapor
          "type": "ITEM",
          "category": "Weapon",
          "tags": [
            "ranged"
          ],
          "buyPrice": {
            "Gold": 65,
            "Silver": 1200
          },
          "sellPrice": {},
          "image": "https://portalx.braincloudservers.com/files/portal/g/14779/metadata/itemDefinitions/ropeCaster.png",
          "resourceGroup": null,
          "resourceTag": null,
          "meta": null,
          "initData": {
            "storeId": [
              "generic_store_1"
            ]
          },
          "pState": "PUBLISHED",
          "publishedAt": 1710773206373,
          "createdAt": 1710772303298,
          "updatedAt": 1750941954774,
          "version": 6,
          "stackable": false,
          "consumable": false,
          "uses": null,
          "coolDownSecs": 0,
          "recoverySecs": 0,
          "activatable": false,
          "statusName": null,
          "activeSecs": null,
          "maxActiveSecs": null,
          "statusBehaviour": null,
          "tradable": false,
          "blockchain": false
        },
        {
          "defId": "blazballBat",
          "name": "Blazeball Bat",
          "desc": "This wooden club is held together by iron rings and smolders with crackling embers while held. When you hit a target with a melee at
          "type": "ITEM",
          "category": "Weapon",
          "tags": [
            "melee"
          ],
          "buyPrice": {
            "Gold": 10
          },
```

Purchasing Items

brainCloud provides the ability to purchase an item using the PurchaseUserItem request, which checks if the player has the required currency balance. This request will debit the player's balance once the item is delivered.

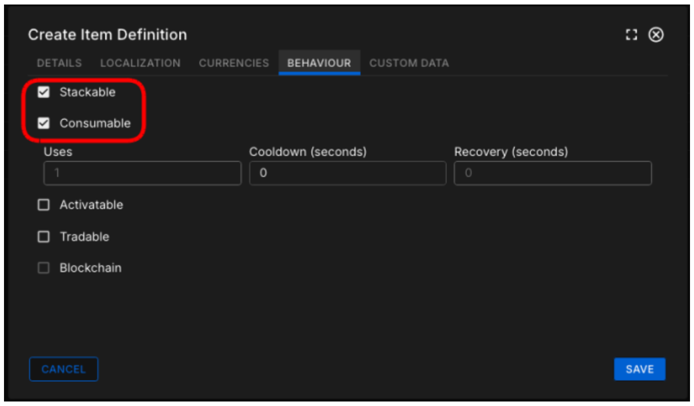
There is also an AwardUserItem, similar to PlayFab's GrantItemToUser API, which won't process the cost of items.

Bundles & Containers

Bundles and containers in PlayFab are similar, both representing an item which contains a set of items delivered to the player. The main difference is that bundle items are delivered immediately upon purchase or delivery, while containers can be kept in the player's inventory and opened later. Upon opening, items are delivered and the container is consumed. PlayFab also allows containers to be opened only if the player owns a specific item, like a key.

While brainCloud allows bundles to be created for IAPs (see examples in IAP & Products section below), items cannot contain a subset of items. However, items can be set as consumable or stackable, and metadata can be associated with items, enabling us to reproduce these features using custom cloud-code.

Let's explore creating bundles and containers in brainCloud through custom features. In the first example, I created two items: a quiver of arrows and an arrow. The quiver will be our bundle, containing 10 arrows . The arrow's behavior should be set as stackable and consumable.

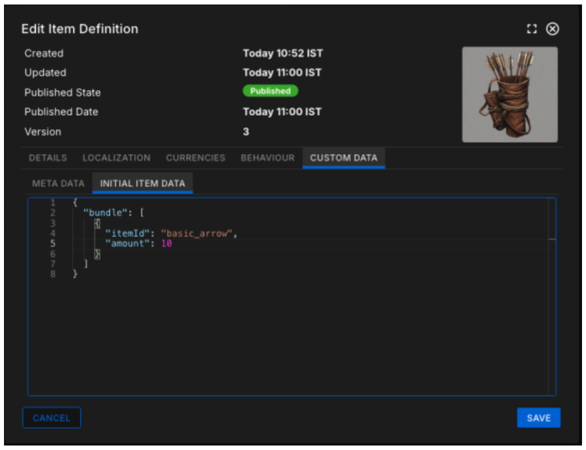

Create Item Definition

- [✓] Stackable
- [✓] Consumable

Uses	Cooldown (seconds)	Recovery (seconds)
1	0	0

- [ ] Activatable
- [ ] Tradable
- [ ] Blockchain

The quiver should have custom data set on the initial Item Data object.


Edit Item Definition

Created	Today 10:52 IST
Updated	Today 11:00 IST
Published State	Published
Published Date	Today 11:00 IST
Version	3

```
{
  "bundle": [
    {
      "itemId": "basic_arrow",
      "amount": 10
    }
  ]
}
```

And finally, make sure to publish these items before testing our custom feature.

For this example, we'll rely on brainCloud's API Hooks (similar to PlayFab's Automation Rules) to trigger custom cloud-code whenever an item is purchased. Check out our guide here Cloud Scripts Document for more information.

To start, create a new cloud-code script called **processItemBundles**. Ensure that client access to this script is disabled. Add the following code to your script.

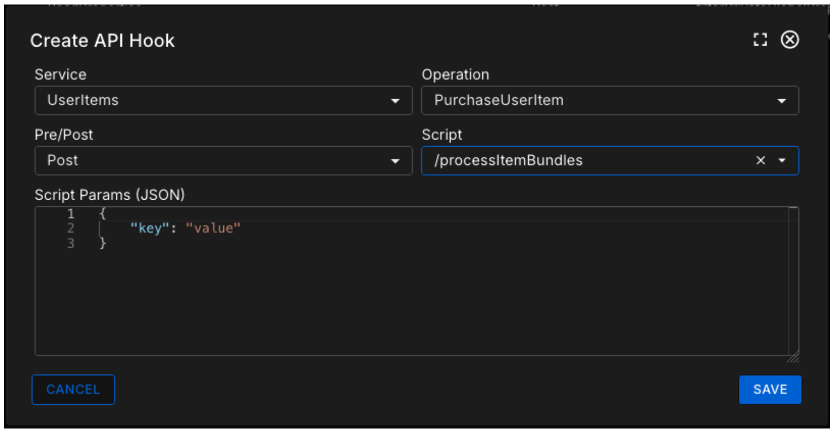
```
"use strict";


function main() {

 var response = {};


 // [1] - Check if this item is a bundle //

 const items = data.message.items;

 const uniqueItemId = Object.keys(items)[0];

 const itemData = items[uniqueItemId].itemData;

 if(itemData && itemData.bundle){

   const userItemsProxy = bridge.getUserItemsServiceProxy();

   // [2] - Loop through the bundle items //

   itemData.bundle.forEach(item => {

     const isItemValid = true; // validate the item if needed //

     if(isItemValid){

       // [3] - Award Item

       let awardItemResp = userItemsProxy.awardUserItem(item.itemId, item.amount, false);

       if (awardItemResp.status == 200) {

           // Success!

       }

       else {

         // - your error here //

       }

     }

     else {

       // - your error here - invalid bundle item id //

     }

   });

   // [4] - Remove item from inventory //

   const dropItemResp = userItemsProxy.dropUserItem(uniqueItemId,
data.callingMessage.quantity, false);
```

```
    if (dropItemResp.status != 200) {

      // - your error here //

    }

  }


  return response;

}


main();
```

This code follows these steps:

1. Checks if the purchased item contains the “bundle” object in its item data.
2. Loops through each object in the bundle array.
3. Awards the item.
4. Removes the original bundle item.

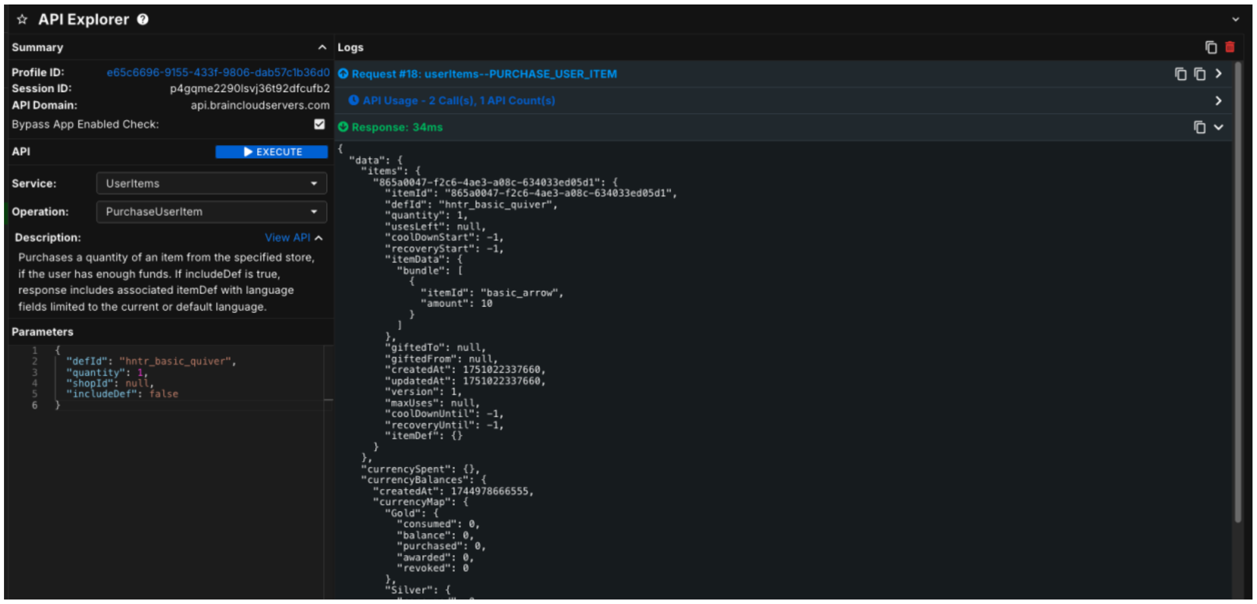
Next, assign this script as a *Post* API Hook on the UserItems service and PurchaseUserItem operation.


Create API Hook

Service: UserItems
Operation: PurchaseUserItem
Pre/Post: Post
Script: /processItemBundles
Script Params (JSON):

```
{
    "key": "value"
}
```

Now we can test this out using the API Explorer.



Profile ID: e65c6696-9155-433f-9806-dab57c1b36d0
Session ID: p4gqme2290lsvj36t92dfcufb2
API Domain: api.braincloudservers.com
Bypass App Enabled Check: ✓

Service: UserItems
Operation: PurchaseUserItem

Description:
Purchases a quantity of an item from the specified store, if the user has enough funds. If includeDef is true, response includes associated itemDef with language fields limited to the current or default language.

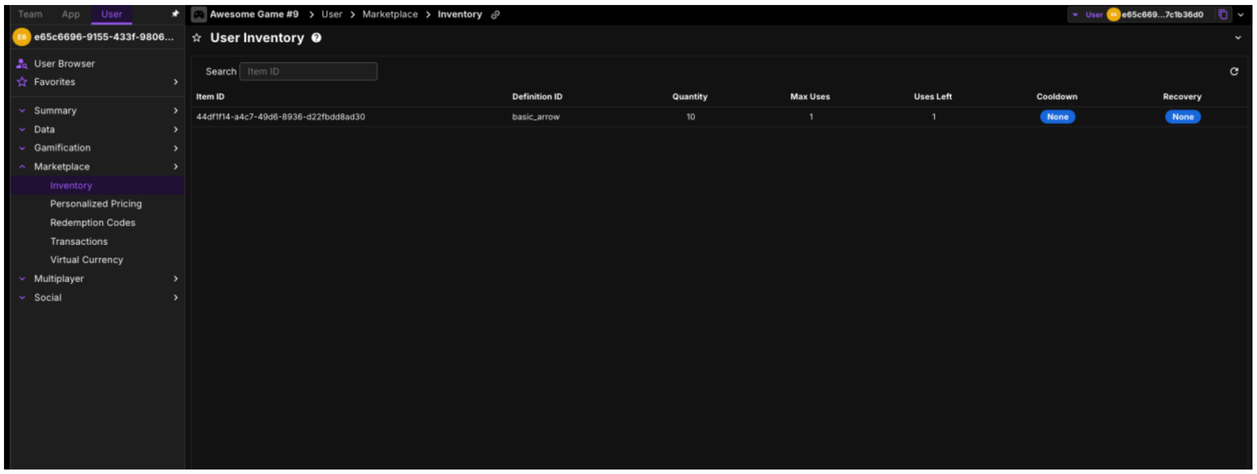
Parameters

```
{
  "defId": "hntr_basic_quiver",
  "quantity": 1,
  "shopId": null,
  "includeDef": false
}
```

Request #18: userItems--PURCHASE_USER_ITEM
API Usage - 2 Call(s), 1 API Count(s)
Response: 34ms

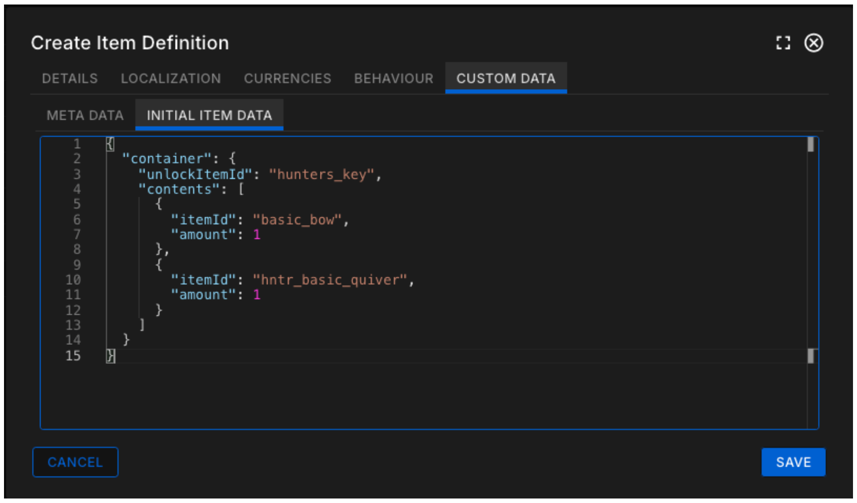
```
{
  "data": {
    "items": {
      "865a0047-f2c6-4ae3-a08c-634033ed05d1": {
        "itemId": "865a0047-f2c6-4ae3-a08c-634033ed05d1",
        "defId": "hntr_basic_quiver",
        "quantity": 1,
        "usesLeft": null,
        "coolDownStart": -1,
        "recoveryStart": -1,
        "itemData": {
          "bundle": [
            {
              "itemId": "basic_arrow",
              "amount": 10
            }
          ]
        },
        "giftedTo": null,
        "giftedFrom": null,
        "createdAt": 1751022337660,
        "updatedAt": 1751022337660,
        "version": 1,
        "maxUses": null,
        "coolDownUntil": -1,
        "recoveryUntil": -1,
        "itemDef": {}
      }
    },
    "currencySpent": {},
    "currencyBalances": {
      "createdAt": 1744978666555,
      "currencyMap": {
        "Gold": {
          "consumed": 0,
          "balance": 0,
          "purchased": 0,
          "awarded": 0,
          "revoked": 0
        },
        "Silver": {
```

Now, when you load the player’s inventory, you'll see the arrows delivered, but the "Hunter's Basic Quiver" item won't be present, as it served only as the holding item for the bundle.



Item ID	Definition ID	Quantity	Max Uses	Uses Left	Cooldown	Recovery
44df1f14-a4c7-49d6-8936-d22fbdd8ad30	basic_arrow	10	1	1	None	None

This is a good solution for bundles, but it won't work for containers that persist in the player’s inventory until consumed. Next, let's look at an example for containers. I created two new items: the "Archer's Supply Chest" (the container) and a bow. The chest will contain the bow and 20 arrows. The chest needs custom item data, similar to what we used for the bundle, but it will also include a unlock item.



```
{
  "container": {
    "unlockItemId": "hunters_key",
    "contents": [
      {
        "itemId": "basic_bow",
        "amount": 1
      },
      {
        "itemId": "hntr_basic_quiver",
        "amount": 1
      }
    ]
  }
}
```

In this case, the container will be purchased first, and then we'll use an API Hook and custom cloud-code script on the UserUserItem to deliver the contents. The unlockItemId is optional, but you'll need to create and award that item to the player for testing.

First, create a new cloud-code script called **processContainerItems** and ensure it is not callable by the client. Add the following code to the script.

```
"use strict";


function main() {

 var response = {};


 // [1] - Check it item is a container //

 const containerData = data.message.item.itemData? data.message.item.itemData.container :
null;

 if(containerData){

   const userItemsProxy = bridge.getUserItemsServiceProxy();

   // [2] - Check if player has the unlock item //

   if(containerData.unlockItemId){

     const context = {

       pagination: { rowsPerPage: 1, pageNumber: 1 },

       searchCriteria: { defId: containerData.unlockItemId },

       sortCriteria: { createdAt: 1 }
```

```
      };

      let getUserItemResp = userItemsProxy.getUserItemsPage(context, false);

      if (getUserItemResp.status == 200) {

        // [3] - Check if we found the item //

        if(getUserItemResp.data.results.count > 0){

          // we need the id of the key later so we can consume it //

          const keyItemId = getUserItemResp.data.results.items[0].itemId;

          // [4] - Iterate through the items //

          containerData.contents.forEach(item => {

            // [5] - Process the items //

            let awardItemResp = userItemsProxy.awardUserItem(item.itemId, item.amount,
false);

            if (awardItemResp.status == 200) {

                // Success!

            }

            else {

              // - your error here //

            }

          });

        }

        // [6] - Remove the container //

        const dropItemResp = userItemsProxy.dropUserItem(data.message.item.itemId, 1,
false);

        if (dropItemResp.status != 200) {

          // - your error here //

        }

        // [7] - Remove the key //

        const dropKeyResp = userItemsProxy.dropUserItem(keyItemId, 1, false);

        if (dropKeyResp.status != 200) {

          // - your error here //

        }

      }
      else {

        // - your error here //

      }

    }

  }


  return response;

}


main();
```

Again, you can test this using the API Explorer. You should see that the chest and key are removed from the player's inventory and a new bow and 20 arrows have been added.

IAPs & Products

Now that we've covered how to recreate items and stores on brainCloud, let's explore purchasing items and currency from third-party stores.

Store Integration

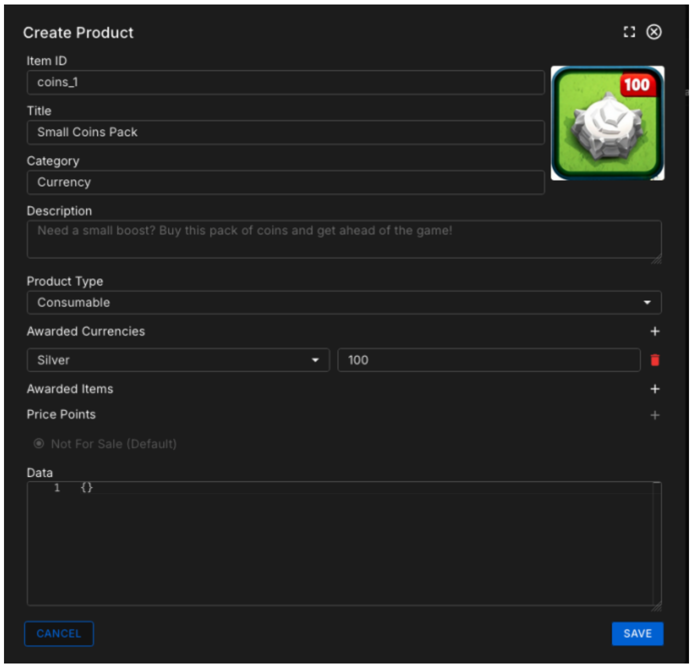
To enable purchases from third-party stores, first add integration details to your brainCloud app. This process varies by platform, so refer to the guide here for setup instructions.

Products

A key difference is that brainCloud items listed in the Item Catalog are separate from items purchasable from third-party apps. In brainCloud, everything purchased with real money are called Products. These Products are located in the **Marketplace → Products** menu.

Create and save a new product

In this example, we're creating a currency pack with the product type "Consumable," indicating it will be converted into coins upon receipt.


Create Product

Item ID: coins_1

Title: Small Coins Pack

Category: Currency

Description: Need a small boost? Buy this pack of coins and get ahead of the game!

Product Type: Consumable

Awarded Currencies: Silver 100

Awarded Items

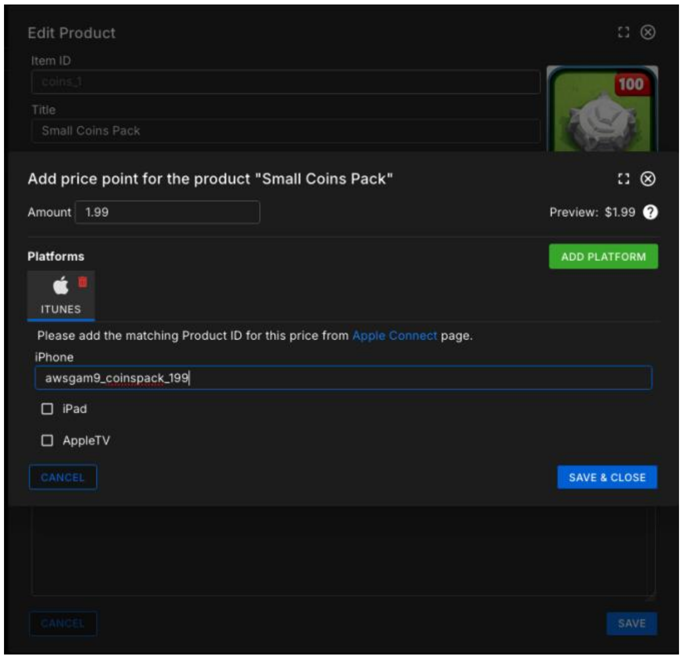
Price Points

Not For Sale (Default)

Data

```
{}
```

When you click save, a popup will prompt you to set a price point. Add a price and an item ID valid for your platform.


Edit Product

Item ID: coins_1

Title: Small Coins Pack

Add price point for the product "Small Coins Pack"

Amount: 1.99

Preview: $1.99

Platforms

ITUNES

Please add the matching Product ID for this price from Apple Connect page.

iPhone: awsgam9_coinspack_199

☐ iPad

☐ AppleTV

If you want to create an item the player can use in their inventory, change the product type and add items from your catalog to the "Awarded Items" field.

The Awarded Items field also lists Bundles so you can sell those for real money.

You can also configure subscriptions by selecting the "Subscription" product type.

Offer Versatility

Products allow you to add multiple Awarded Items or Bundles. You can also choose to combine multiple items in a Bundle, as well as Virtual Currency, and then Award the Bundle through the Awarded items field. This gives the developer maximum control to manage offers as they wish.

Listing Products

We've covered listing items and stores with brainCloud, but there can be ambiguity between products and catalog items, which isn't an issue in PlayFab.

We can list products using the GetSalesInventory or GetSalesInventoryByCategory. These are specific to the platform you're using and will return products for that platform only.

Validating Purchases

In PlayFab's legacy system, purchase processing happens between the client and the third-party service, then a receipt is passed to PlayFab's store validation APIs. This validation API confirms the purchase, but doesn't grant items to the player. Developers must deliver items manually or via rules and cloud-scripts.

Version 2 item purchases differ for some platforms by using a single API call that validates a store receipt and grants the item to the user simultaneously.

brainCloud similarly varies by store, with some requiring additional integration steps. The Economy API Comparison table at the end of this guide, outlines these differences for each third-party store.

Fulfilling Purchases & Delivering Items

When a purchase is performed or validated correctly with brainCloud, the associated items or currency are delivered to the player automatically. There's no need to send additional client requests or use extra cloud-code for item or currency delivery.

Coupons/Redemption Codes

PlayFab allows developers to generate coupons for items, which players can redeem using the RedeemCode request to receive the item.

brainCloud has a similar feature called Redemption Codes, found in the **Marketplace → Redemption Codes** menu. Unlike PlayFab's coupons, brainCloud lets developers set up campaigns based on these codes. Redemptions are tracked, and campaigns can be scheduled. These codes can also trigger a custom cloud-code Fulfillment Script to perform any required action. For this example, we'll create a cloud-code script to deliver the item configured in the campaign.

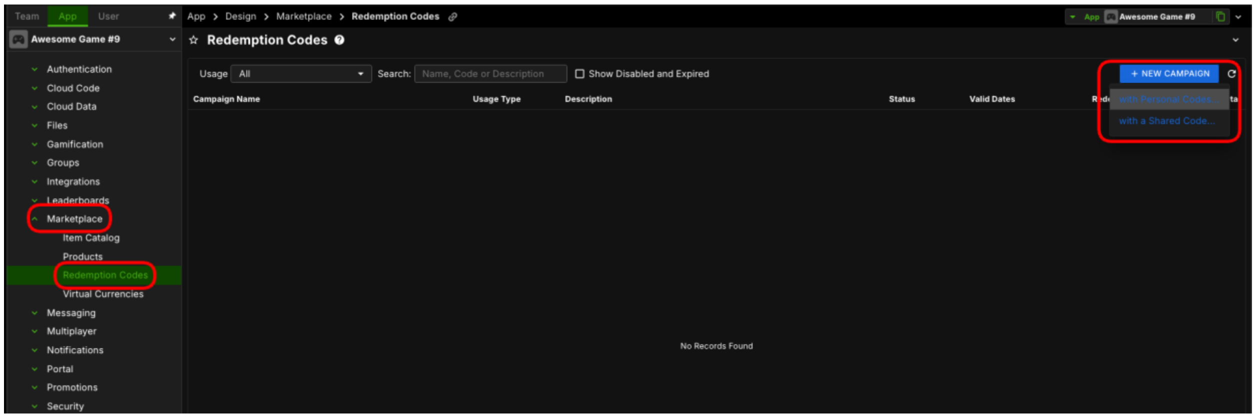
Create a new cloud-code script called **redemptionCodeProcessItem** and ensure it’s not callable from the client. Add the following code to the script.

```
"use strict";


function main() {

 var response = {};


 const itemId = data.customCodeInfo.itemId;

 const amount = data.customCodeInfo.amount;

 const userItemsProxy = bridge.getUserItemsServiceProxy();

 const postResult = userItemsProxy.awardUserItem(itemId, amount, false);

 if (postResult.status == 200) {

   response = {

       "success": true,

       "complete": true

   };

   return response;

 }


 return response;

}
```

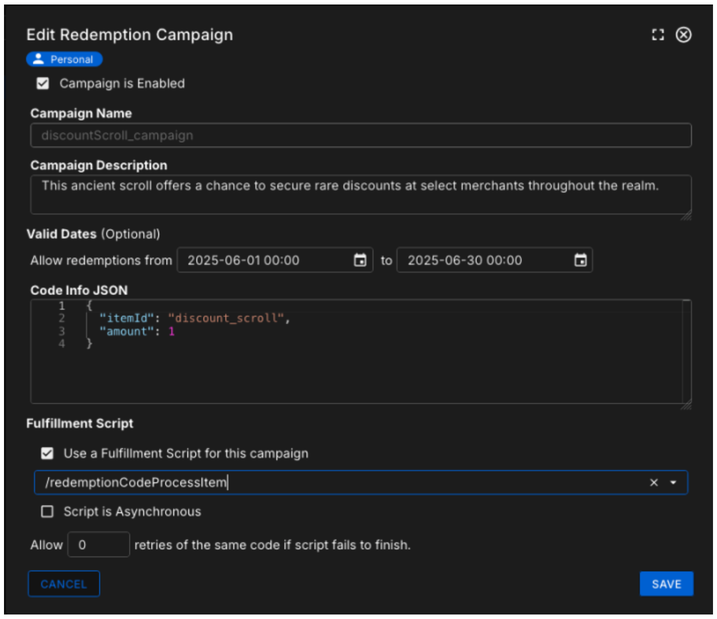
```
main();
```

Personal Codes

The closest equivalent to PlayFab's coupons in brainCloud are Personal Codes. To create a new personal code campaign, click the "New Campaign" button on the Redemption Codes dashboard.



Next, set up your campaign details. In this example, we add the reward-item details to the info JSON field. Be sure to include the fulfillment script as well.


Edit Redemption Campaign

Personal

☑ Campaign is Enabled

Campaign Name
discountScroll_campaign

Campaign Description
This ancient scroll offers a chance to secure rare discounts at select merchants throughout the realm.

Valid Dates (Optional)
Allow redemptions from 2025-06-01 00:00 to 2025-06-30 00:00

Code Info JSON

```
{
  "itemId": "discount_scroll",
  "amount": 1
}
```

Fulfillment Script

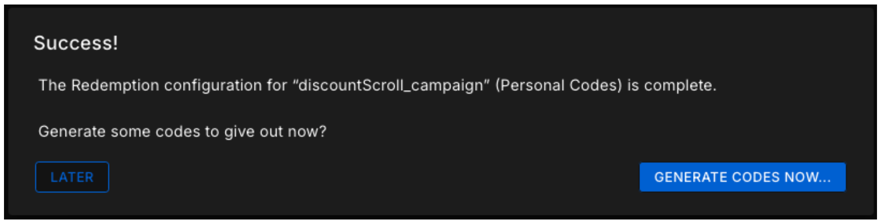
☑ Use a Fulfillment Script for this campaign

/redemptionCodeProcessItem

☐ Script is Asynchronous

Allow 0 retries of the same code if script fails to finish.

After clicking the "Save" button, a popup will indicate the campaign was created and ask if you want to generate codes. Click "Generate Codes Now."

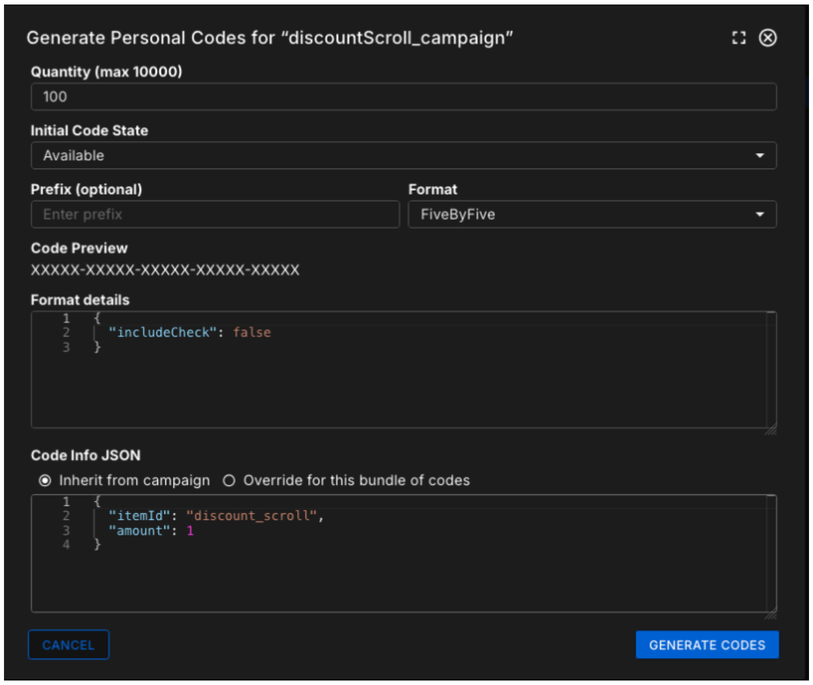


Success!

The Redemption configuration for “discountScroll_campaign” (Personal Codes) is complete.

Generate some codes to give out now?

The next panel lets you choose options for generating codes, including the number of codes for the campaign. For this example, leave these as defaults.



Generate Personal Codes for “discountScroll_campaign”

Quantity (max 10000): 100

Initial Code State: Available

Prefix (optional): Enter prefix

Format: FiveByFive

Code Preview: XXXXX-XXXXX-XXXXX-XXXXX-XXXXX

Format details

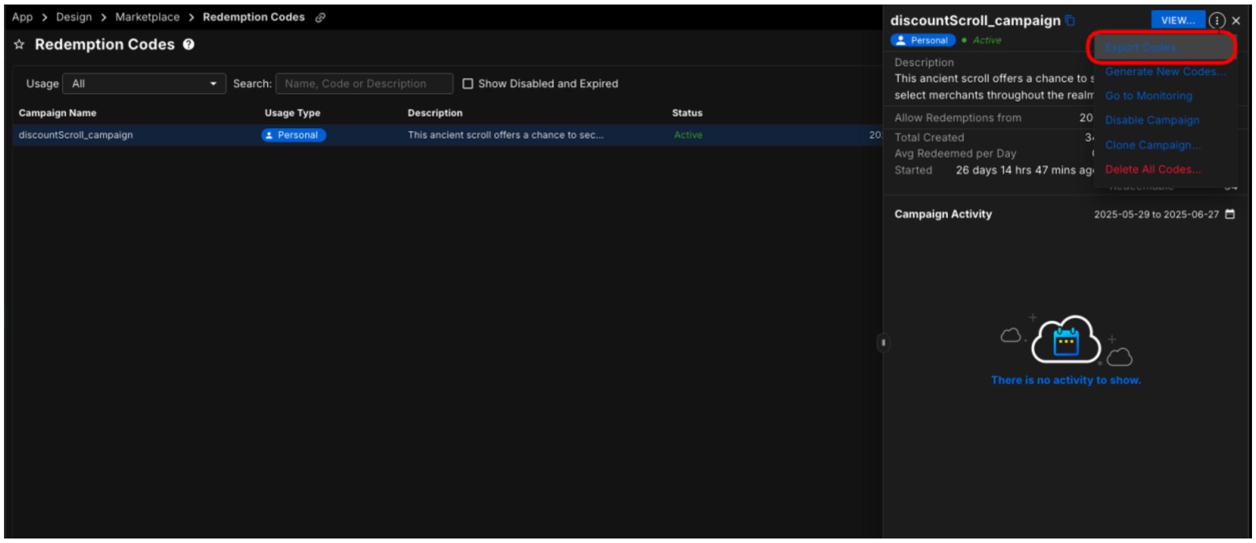
```
{
  "includeCheck": false
}
```

Code Info JSON

[✓ Inherit from campaign] Override for this bundle of codes

```
{
  "itemId": "discount_scroll",
  "amount": 1
}
```

Now our campaign is set up. To test it, download some codes by clicking on the campaign in the Redemption Codes dashboard and selecting "Export Codes" from the panel on the left side of the dashboard.



Campaign Name	Usage Type	Description	Status
discountScroll_campaign	Personal	This ancient scroll offers a chance to sec...	Active

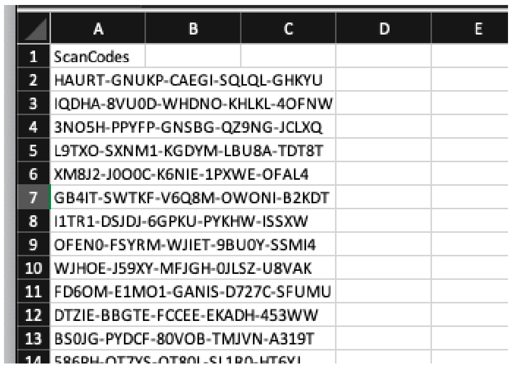
discountScroll_campaign
Personal • Active

Description
This ancient scroll offers a chance to s
select merchants throughout the realm

Allow Redemptions from 20
Total Created 34
Avg Redeemed per Day
Started 26 days 14 hrs 47 mins ag

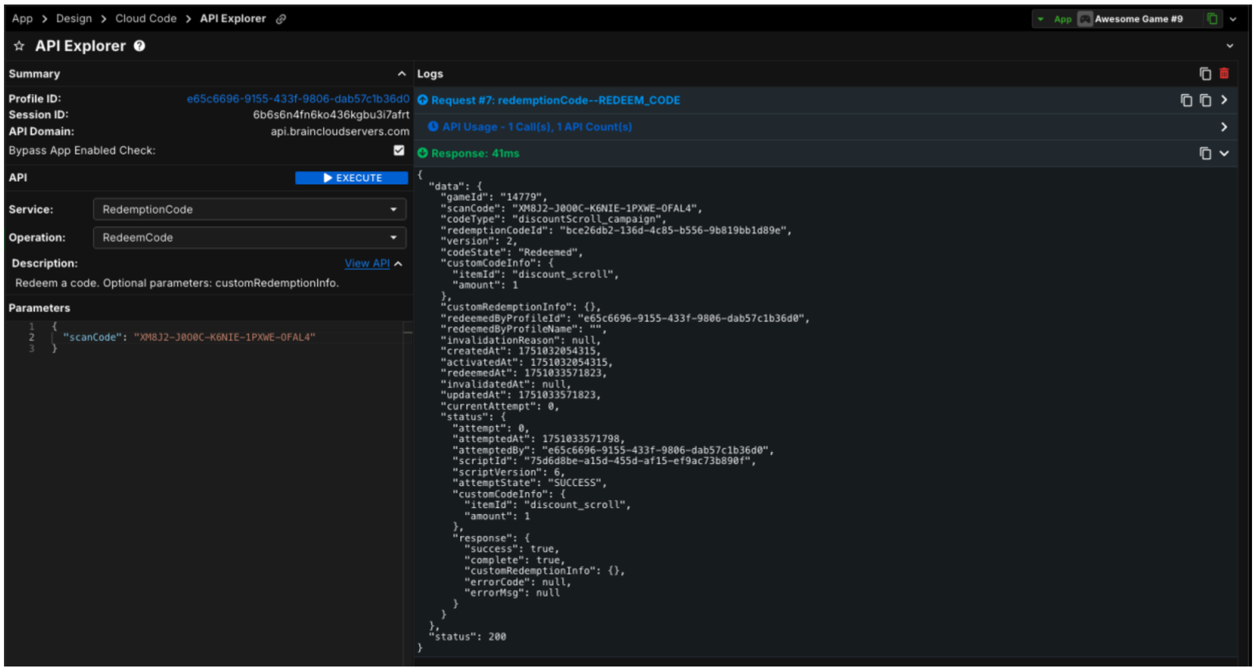
Campaign Activity 2025-05-29 to 2025-06-27

An excel sheet containing the codes will be email to you.



	A	B	C	D	E
1	ScanCodes				
2	HAURT-GNUKP-CAEGI-SQLQL-GHKYU				
3	IQDHA-8VU0D-WHDNO-KHLKL-4OFNW				
4	3NO5H-PPYFP-GNSBG-QZ9NG-JCLXQ				
5	L9TXO-SXNM1-KGDYM-LBU8A-TDT8T				
6	XM8J2-J0O0C-K6NIE-1PXWE-OFAL4				
7	GB4IT-SWTKF-V6Q8M-OWONI-B2KDT				
8	I1TR1-DSJDJ-6GPKU-PYKHW-ISSXW				
9	OFEN0-FSYRM-WJIET-9BU0Y-SSMI4				
10	WJHOE-J59XY-MFJGH-0JLSZ-U8VAK				
11	FD6OM-E1MO1-GANIS-D727C-SFUMU				
12	DTZIE-BBGTE-FCCEE-EKADH-453WW				
13	BS0JG-PYDCF-80VOB-TMJVN-A319T				
14	586PH-OT7YS-OT8OL-SL1RO-HT6XL				

Now we can test one of these codes out using the API Explorer.



Profile ID: e65c6696-9155-433f-9806-dab57c1b36d0
Session ID: 6b6s6n4fn6ko436kgbu3i7afrt
API Domain: api.braincloudservers.com
Bypass App Enabled Check: ✓

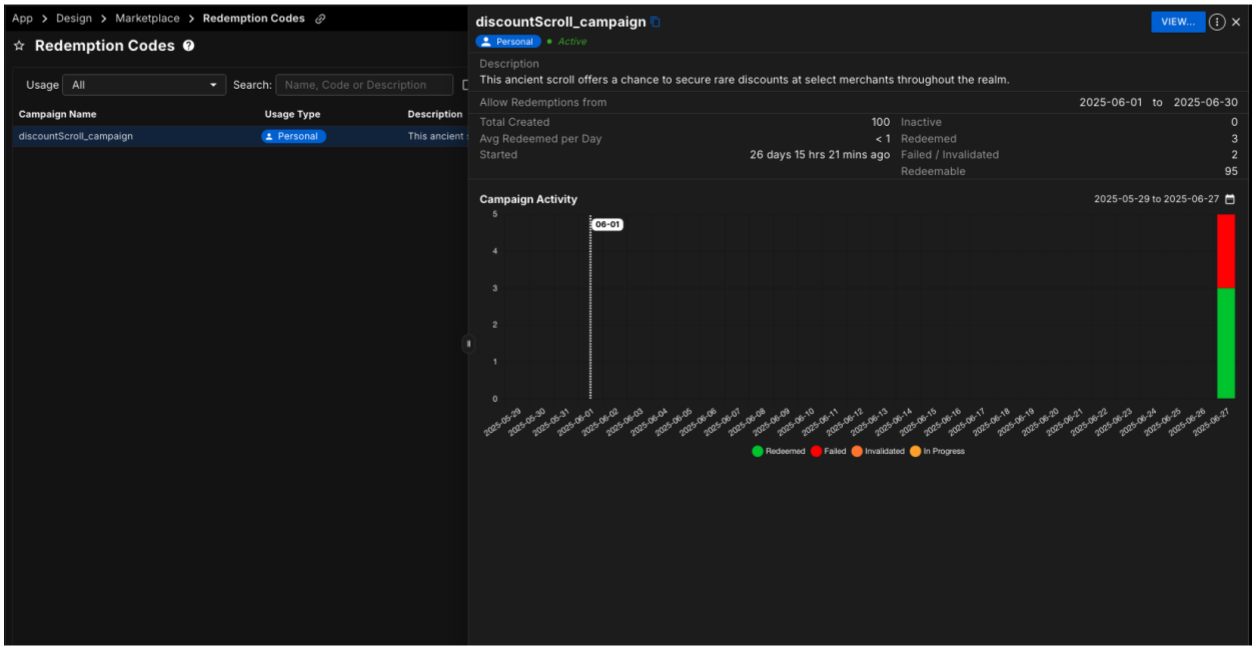
Service: RedemptionCode
Operation: RedeemCode
Description: Redeem a code. Optional parameters: customRedemptionInfo.

```
{
  "scanCode": "XM8J2-J000C-K6NIE-1PXWE-0FAL4"
}
```

Request #7: redemptionCode--REDEEM_CODE
API Usage - 1 Call(s), 1 API Count(s)
Response: 41ms

```
{
  "data": {
    "gameId": "14779",
    "scanCode": "XM8J2-J000C-K6NIE-1PXWE-0FAL4",
    "codeType": "discountScroll_campaign",
    "redemptionCodeId": "bce26db2-136d-4c85-b556-9b819bb1d89e",
    "version": 2,
    "codeState": "Redeemed",
    "customCodeInfo": {
      "itemId": "discount_scroll",
      "amount": 1
    },
    "customRedemptionInfo": {},
    "redeemedByProfileId": "e65c6696-9155-433f-9806-dab57c1b36d0",
    "redeemedByProfileName": "",
    "invalidationReason": null,
    "createdAt": 1751032054315,
    "activatedAt": 1751032054315,
    "redeemedAt": 1751033571823,
    "invalidatedAt": null,
    "updatedAt": 1751033571823,
    "currentAttempt": 0,
    "status": {
      "attempt": 0,
      "attemptedAt": 1751033571798,
      "attemptedBy": "e65c6696-9155-433f-9806-dab57c1b36d0",
      "scriptId": "75d6d8be-a15d-455d-af15-ef9ac73b890f",
      "scriptVersion": 6,
      "attemptState": "SUCCESS",
      "customCodeInfo": {
        "itemId": "discount_scroll",
        "amount": 1
      },
      "response": {
        "success": true,
        "complete": true,
        "customRedemptionInfo": {},
        "errorCode": null,
        "errorMsg": null
      }
    }
  },
  "status": 200
}
```

A successful response indicates the code was deactivated and the item delivered. You can confirm this by returning to the Redemption Codes dashboard and checking the analytics for your campaign, which shows the number of codes redeemed and on which days.



Usage: All

Campaign Name	Usage Type	Description
discountScroll_campaign	Personal	This ancient

discountScroll_campaign
Personal · Active

Description
This ancient scroll offers a chance to secure rare discounts at select merchants throughout the realm.

Allow Redemptions from 2025-06-01 to 2025-06-30

Total Created	100	Inactive	0
Avg Redeemed per Day	< 1	Redeemed	3
Started	26 days 15 hrs 21 mins ago	Failed / Invalidated	2
		Redeemable	95

Economy API Comparison

For quick reference, below is a list of PlayFab economy and store APIs, including purchase validation, redemption calls, and integration guides for setting them up on brainCloud. Note that player inventory calls are covered in another guide on players and player data found here Player Inventory.

PlayFab	brainCloud	
GetCatalogItems GetStoreItems GetItems SearchItems	GetCatalogItemsPage GetSalesInventory GetSalesInventoryByCategory	
GetItem	GetCatalogItemDefinition	
GrantItem(s)ToUser	PurchaseUserItem AwardUserItem	
RedeemCoupon	RedeemCode	
Purchase Validation		
ValidateAmazonIAPReceipt	VerifyPurchase	guide
ValidateGooglePlayPurchase RedeemGooglePlayInventoryItems	ConfirmGooglePlayPurchase	guide
ValidateIOSReceipt RedeemAppleAppStoreInventoryItems	VerifyItunesRecipt	guide

ValidateWindowsStoreReceipt RedeemMicrosoftStoreInventoryItems	VerifyMicrosoftReceipt	
RedeemSteamInventoryItems	StartSteamTransaction FinalizeSteamTransaction	