

## Store Offers Example



### Description

This Demo shows how to create Store for In-Apps and other in-game items. The store is using Balancy to manage all the data. It has not only ordinary offers, but also conditional offers. Such offers are available for players only after certain conditions are complete.

There are 2 types conditions:

1. Reach certain level
2. Make N purchases

You can can as many other conditions as you want to.

### How to start

1. Clone [Git Project](#)
2. Open Project in Unity
3. Launch Scene Assets/StoreDemo/Scenes/SampleScene.unity
4. Press Play
5. Click on offers to purchase items. Please note that once you make 5 purchases, new offers appear.








## Create your own Balance

1. Open Balancy [Dashboard](#)
2. Click on Examples button
3. Find **Store Offers** example
4. Click on Clone and confirm

You'll get your own version of Store Offers app on Balancy. Follow our documentation and replace in Unity Project Gameld from our project to your newly created one.

## Brief description of Balancy setup for Store Offers

1. **Item Model** is the template all items. In my example I've created Gems and Gold as Items.
2. All items, including resources are stored in Inventories. Thus we have **Inventory Config**. We'll speak more about Inventory in our next Example Project with Inventories.
3. **Store Offer** is the main Template for this Example Project. It looks like this

Id ▾ ▲	Icon ▾ ▲	Conditions ▾	Items ▾ ▲	Price ▾ ▲
53	 Assets/Icon/chest4.png ▾	Purchases Count (Inverse: false; Min Purchases: 5)	Item: [14] Gold; Count: 400	Item: [13] Gems; Count: 300
56	 Assets/Icon/chest5.png ▾	Purchases Count (Inverse: false; Min Purchases: 5)	Item: [14] Gold; Count: 750	Item: [13] Gems; Count: 400
50	 Assets/Icon/chest6.png ▾	Purchases Count (Inverse: false; Min Purchases: 10), Player Level (Level: 5; Inverse: false)	Item: [14] Gold; Count: 1000	Item: [13] Gems; Count: 500
27	 Assets/Icon/donat1_image.png ✓	<SELECT>	Item: [13] Gems; Count: 50	Count: 0
32	 Assets/Icon/donat2_image.png ✓	<SELECT>	Item: [13] Gems; Count: 100	Count: 0
35	 Assets/Icon/donat3_image.png ✓	<SELECT>	Item: [13] Gems; Count: 250	Count: 0
38	 Assets/Icon/donat4_image.png ✓	<SELECT>	Item: [13] Gems; Count: 500	Count: 0

1. The first parameter is **Icon**. This is the image, which is used in the game store. We synchronized Unity Addressables with Balancy to make it work. You can find how it works [here](#).
2. **Conditions** is the most tricky part. I'll explain it in the next section
3. **Items** list of items you get when you purchase the offer. I'm using component **ItemWithAmount** here - you can find it in the Templates list.
4. **Price** are the items, which are taken during the purchase.
5. There are other parameters, please check them by yourself.

## Conditions

Conditions are often used to hide some parts of a game from a player before he is ready.

**Condition** can be anything:

- player's level
- find a specific item
- killed a certain enemy
- etc...

You should implement Conditions logic only once in Balancy and in your code and after that you can use it everywhere: Tutorials, Quests, Dialogs,... I'm using them in Store Offers.

- To see offers with Ids **53** and **56** player must make 5 purchases
- For offer **50** player must have at least level 5 and make 10 purchases

You can see this information in my table (screenshot above). **ConditionChecker.cs** is the class, which works with conditions. You should check it's implementation. If you decide any new types of Conditions, you'll need to make changes only there.