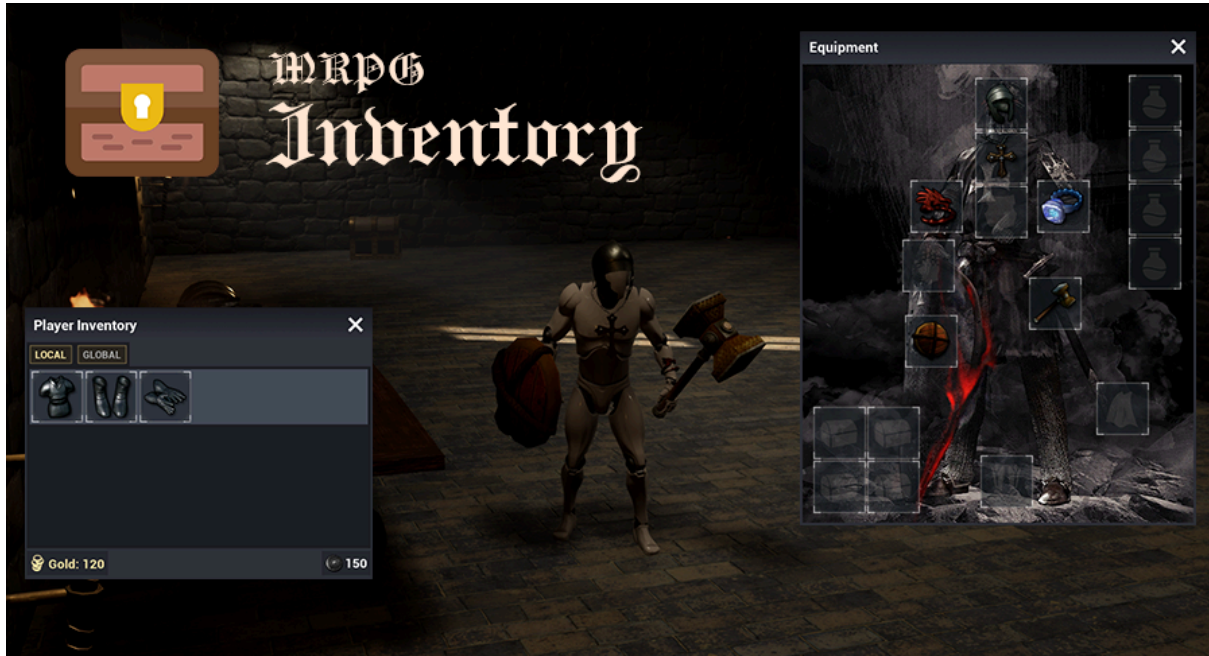


Multiplayer RPG Inventory System

now with XSolla Online services!



Version 2.0!

20.04.2021

Unreal Engine Plugin

Author: Tefel

Table of Contents

[Introduction](#)

[Plugin Installation](#)

[Integration](#)

[Basic Example](#)

[Testing Multiplayer](#)

[Integrating into existing project](#)

[Advanced Example with Xsolla](#)

[Custom XSolla Store](#)

[Creating new Project \(Store\)](#)

[Items](#)

[Virtual Currency](#)

[Bundles](#)

[World Actors](#)

[Pickable Items](#)

[StorageLoot](#)

[Multiplayer](#)

[InventoryPlayerController](#)

[InventoryManager](#)

[InventoryCharacter](#)

[ContainerComponent](#)

[Equipable Items](#)

[Draggable Windows](#)

[Plugin Structure](#)

Introduction

Multiplayer RPG Inventory System is complete and fully functional inventory management system. Very strong side of the plugin is very clean and simple to follow code. In fact it is extremely easy to build a game based on the system or mixed it with an already working prototype of the game. Regardless of use case - RPG / Survival / Multiplayer / Single Player is the perfect choice being always reliable.

UPDATE 2.0

After updating MRPG Inventory System becomes the most robust inventory system in the Marketplace! MRPG inventory provides Online Services, Registration, Login, Progression, Friends, Credit Card transactions and reliable Client-Server communication for all your players!

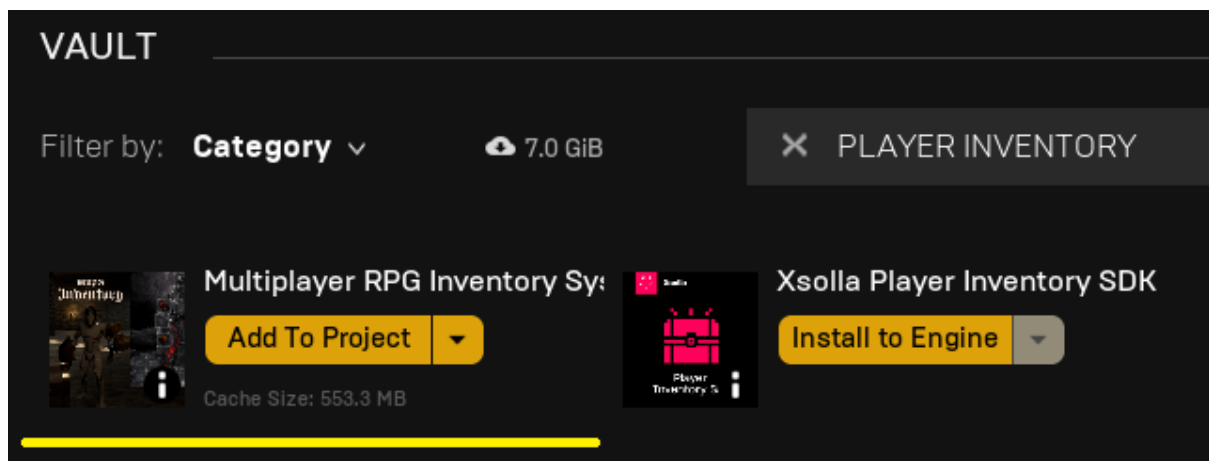
Have any questions before purchase?

- Join Discord and ask me directly
<https://discord.com/invite/SQ6JEkQ>

Plugin Installation

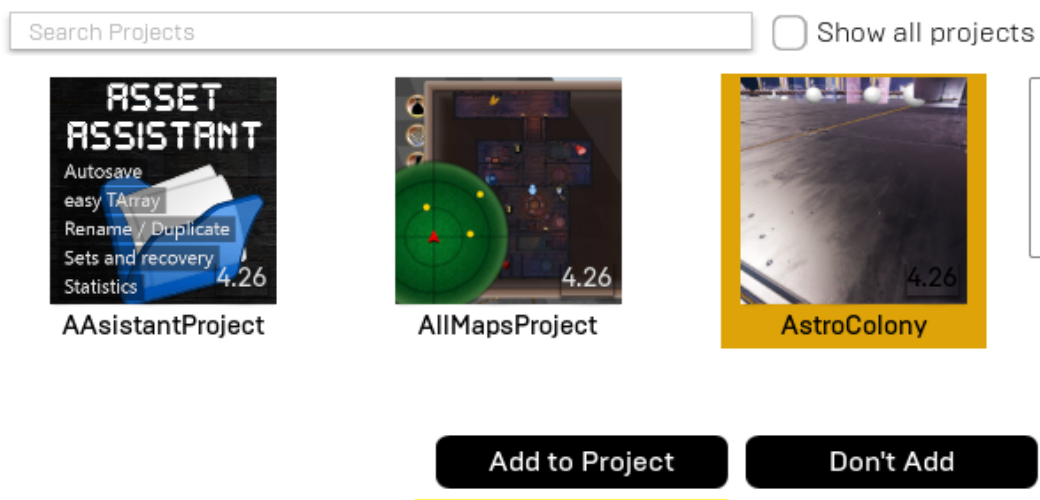
Thank you for purchasing the **Multiplayer RPG Inventory System 2.0**! Before you start reading step-by-step instructions you may want to check related video tutorials here: <https://www.youtube.com/watch?v=WY3Nfse5ykw>

In this section, you will learn how to install the plugin using the Epic Games Launcher. Once you've purchased the MRPG Inventory System, please ensure that you restart the launcher. This will ensure that your Vault refreshes, and it should show the plugin. As for the Vault, this is located at the bottom of your Library tab (UNREAL ENGINE > Library > Scroll Down).



Once you're there, search for Inventory and press the "Add to project" button. This will bring up a popup asking you which project you want to add it to.

Select the Project to Add the Asset to



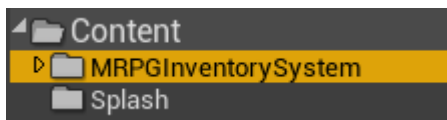
The plugin is compatible with any Unreal Engine 4 template. It creates some binding if necessary and uses its own Third Person Character.

Integration

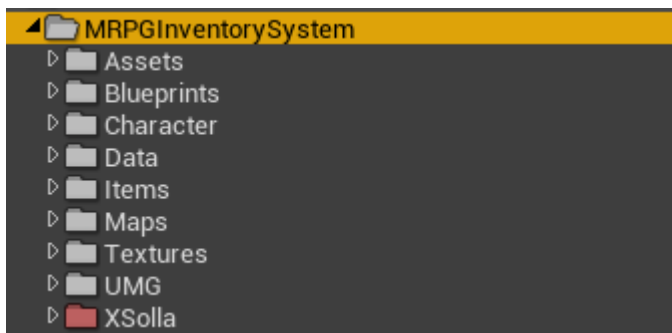
When you open the project for the first time you should see a proper Splash Image, it means that plugin is available!



Go to the MRPGInventorySystem content directory and Open it



It's You will see that is divided into multiple folders

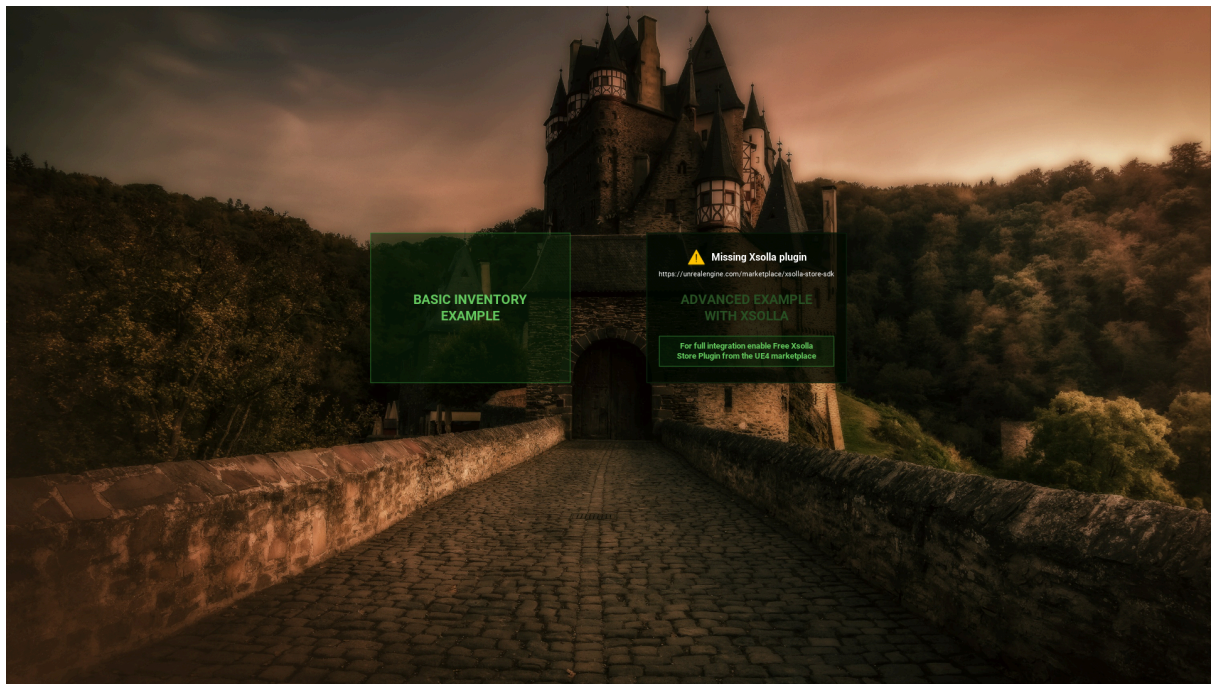


- Assets - contains non-relevant assets / environments, fonts and meshes
- **Blueprints** - Multiplayer Inventory logic - Actors, Components, Libraries, Game relevant classes
- Character - default UE4 Third Person Character
- **Data** - Enum, Structs and Data Tables
- Items - unique meshes for inventory items
- **Maps** - Demo maps, you should start **THERE**
- **XSolla** - files required for integration Online Services

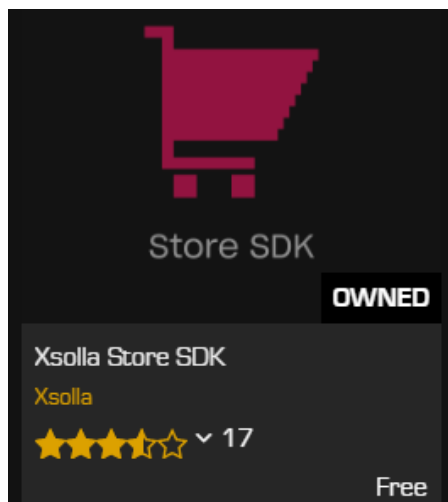
First go to the **MRPGInventorySystem / Maps /** and open **L_MainMenu**.

This map is completely black and shows you a simple selection menu when you play it.

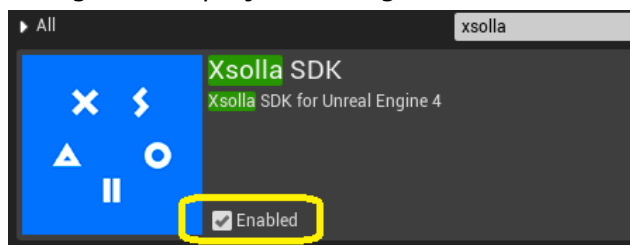
It should only allow you to select “Basic Inventory Example”



To enable advanced example you need to install **Free XSolla Store SDK** plugin



Install it to the Engine exactly the same like all other Engine plugins.
Now go to the project settings and enable Xsolla SDK



After this operation you will see that both options in the **L_MainMenu** map will become available.

Basic Example



If you are looking into Multiplayer Inventory without additional features like players registration and online transactions this example is for you.

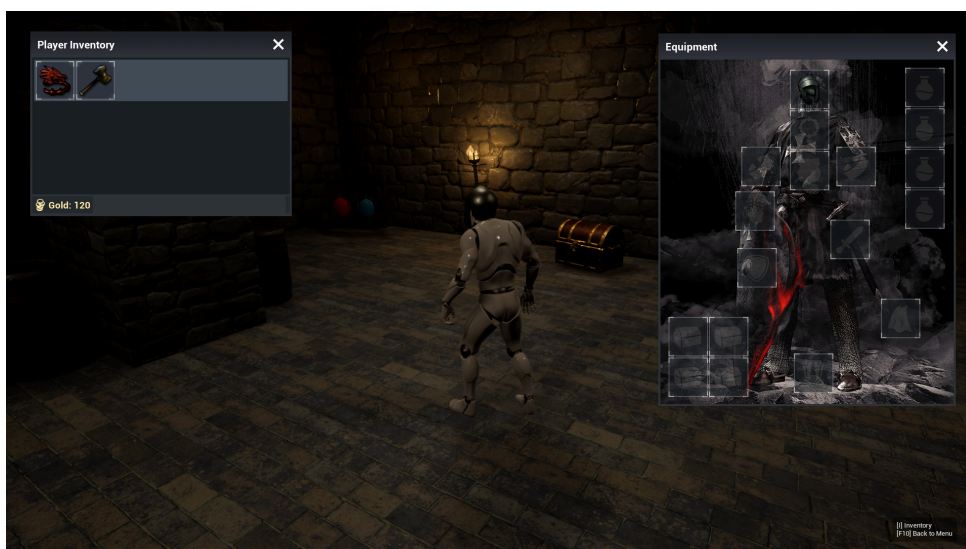
It presents server-client communication so you can test it with multiple players joining the session.

Basic Inventory Example launches **L_MRPGInventory** located in **Maps/**, but you want to open it directly.



Example starts in the scene where you can move your character using **WASD** and rotate the camera using the pressed **right mouse button** with **mouse rotation**.

It presents basic functionality like opening the chest on the server, getting information about the storage container from the server and presenting it inside your **HUD**.



Press **I** to open your players inventory and Equipment.

You can freely move items by dragging them. When you start dragging all compatible slots will become highlighted. This helps you to find the right spot for your equipment. With Inventory plugin every possible drag/drop operation is supported!

- Drop item **on the ground** will place it in the World, so players can pick it up
- Drop item to **Equiptement** and observe how that your character starts wearing it
- Drop item into **another chest** will transfer it there
- Drop item into **shop** will sell the item



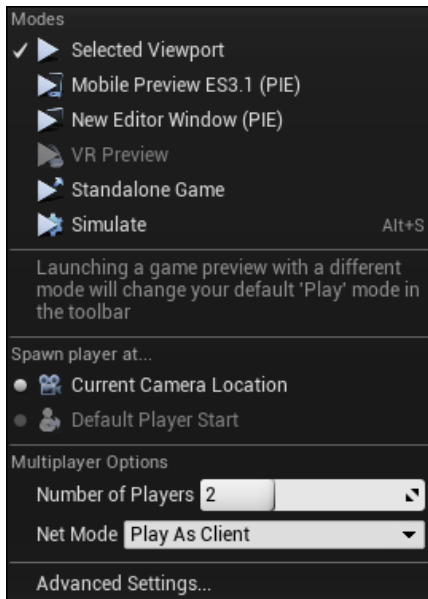
Picking items / gold - every picked and goal item goes into your player inventory



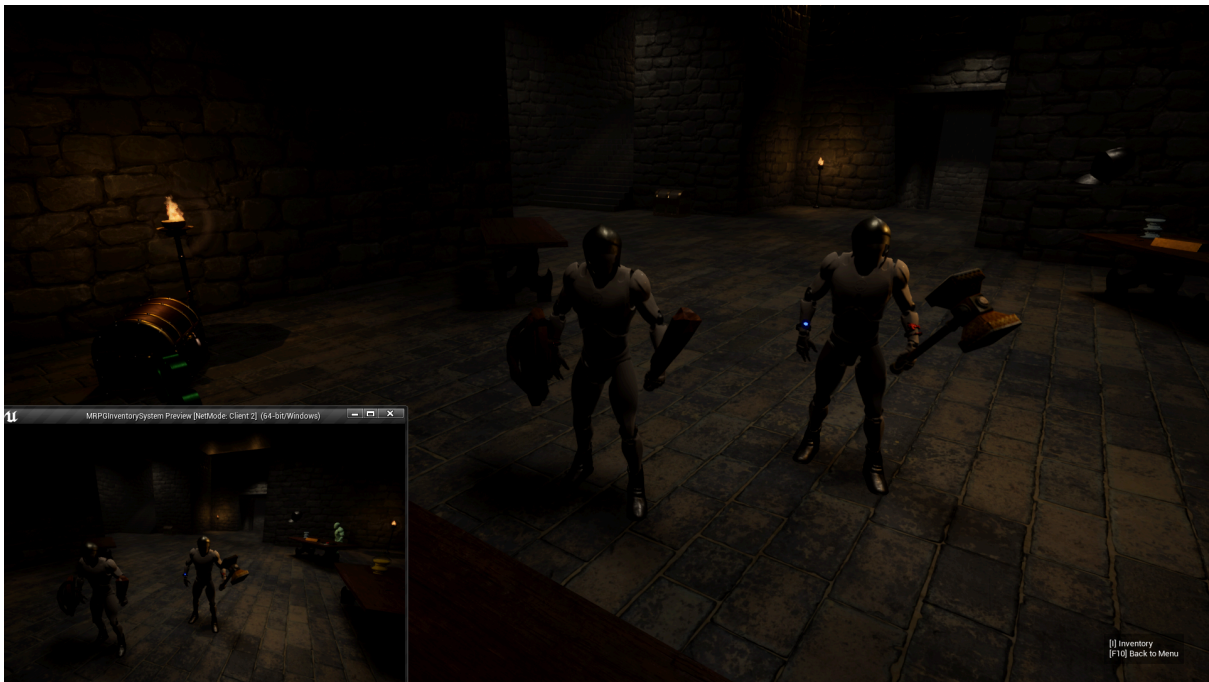
You can trade items with **Local Vendors**. They offer predefined items. Every transaction is validated on the server. After every transaction your gold amount is going to change.

- **Testing Multiplayer**

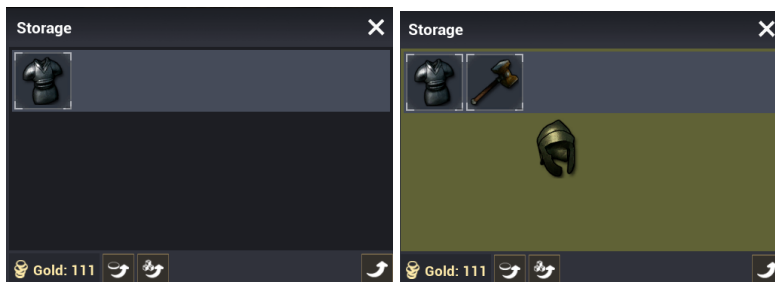
To test all multiplayer features you can simulate them starting **L_MRPGInventory** with multiple Clients:



This way two Players will play with each other. They will see when another player moves, equips something, picks the item or put something on the ground.



Additionally, the player will get feedback when the opened chest changes its contents.

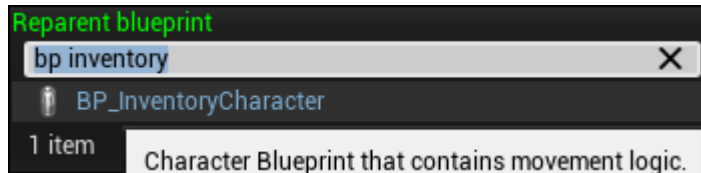


- **Integrating into existing project**

If your project already contains gameplay logic and all gameplay classes like PlayerControllers & PlayerCharacters you may want to follow steps from this section.

Method A

Reparent blueprints so they use default inventory classes. Open your Character, go to Class settings and select **BP_InventoryClass** as a Parent Class

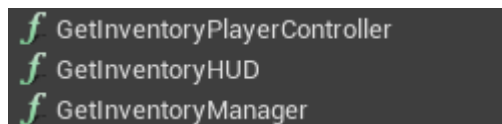


Do exactly the same for your PlayerController using **InventoryPlayerController**

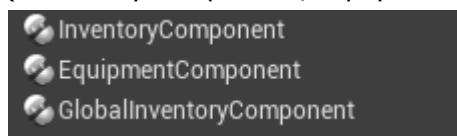
Method B

If for some reason you cannot reparent your blueprint classes you can still copy logic from them.

StepA - make sure that InvFunctionLibrary is referencing proper Gameplay classes:

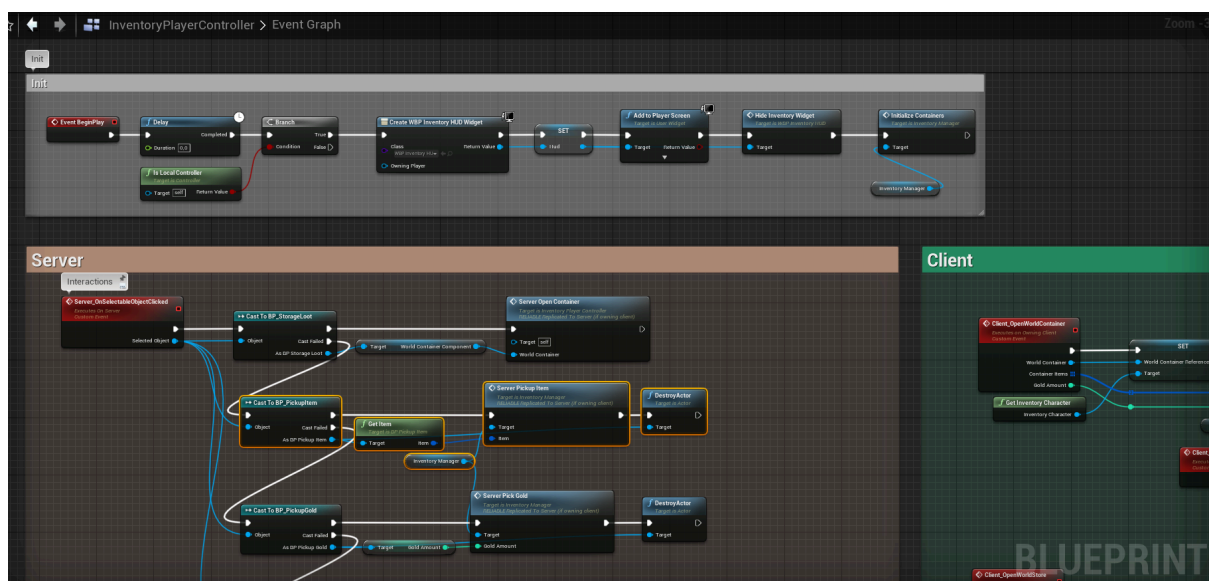


StepB - add MRPG Components into your Inventory Character class (InventoryComponent, EquipmentComponent, GlobalInventoryComponent)



Add missing logic from the blueprint itself, OnRep for replication of meshes and Equip/Unequip functions.

StepC - copy main blueprint functions into your player controller



Currently, an instructional video is being prepared showing how to accurately integrate the plugin into your project. <https://www.youtube.com/unrealtefel>
Link will appear here as soon as it's available.

Advanced Example with Xsolla



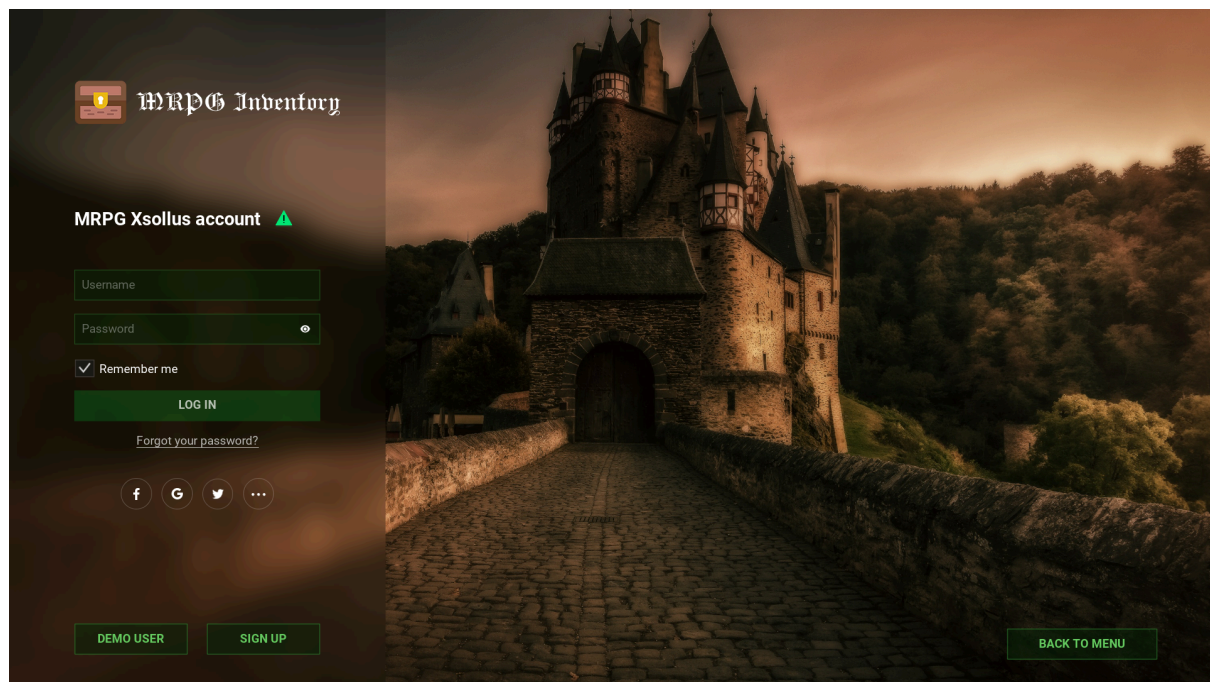
Advanced Example provides full integration of all Online Services to your game. It means that players in your game will be able to register and log-in, change their avatar and profile details. Their character will progress and reach higher levels. Players will be able to purchase custom items and pay for them with Credit Card or Virtual in-game currency!

Advanced Inventory Example launches **L_MRPGInventoryXsolla** located in Maps/, but you may want to open it directly.

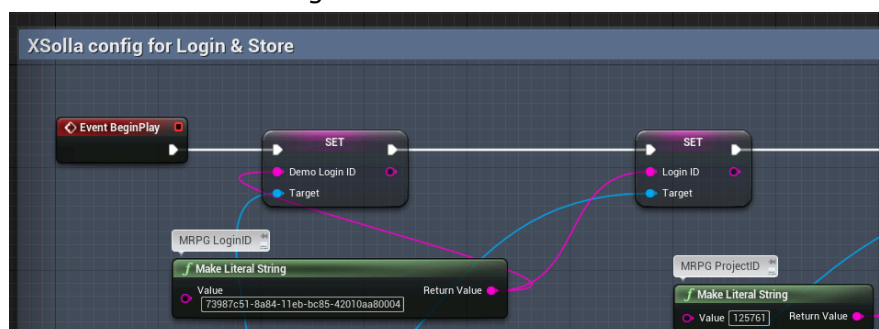
Before going further you may want to check tutorial videos for Advanced Example:

<https://www.youtube.com/watch?v=JGbKiXdsmKE>

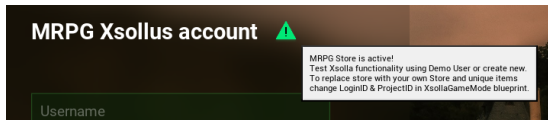
When you pick an advanced example you will need to provide login and password to connect to the Server. You can obviously create your own new Account in MRPG store, but you may also use **Demo user**: login **MRPGUser**, password **MRPGUser**



By default the plugin automatically configures MRPG Shop. This means you can use **MRPGUser**. This configuration is located in **XSollaGameMode**.

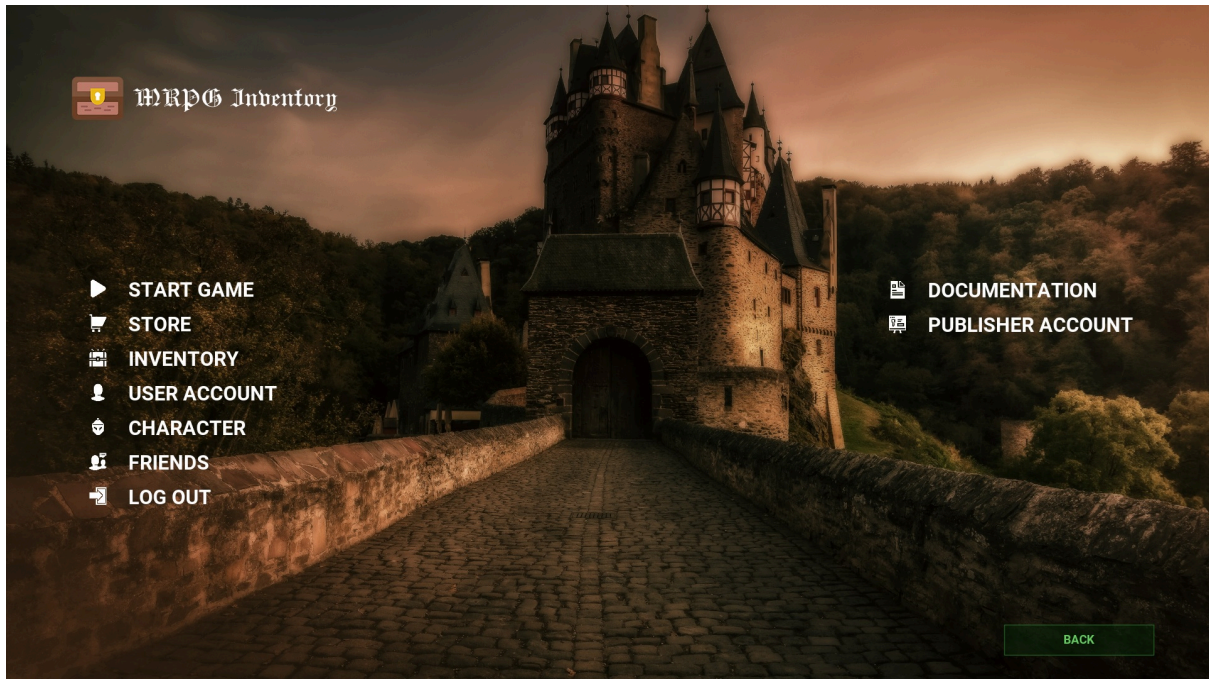


To configure your own store go to the [Custom XSolla Store section](#)



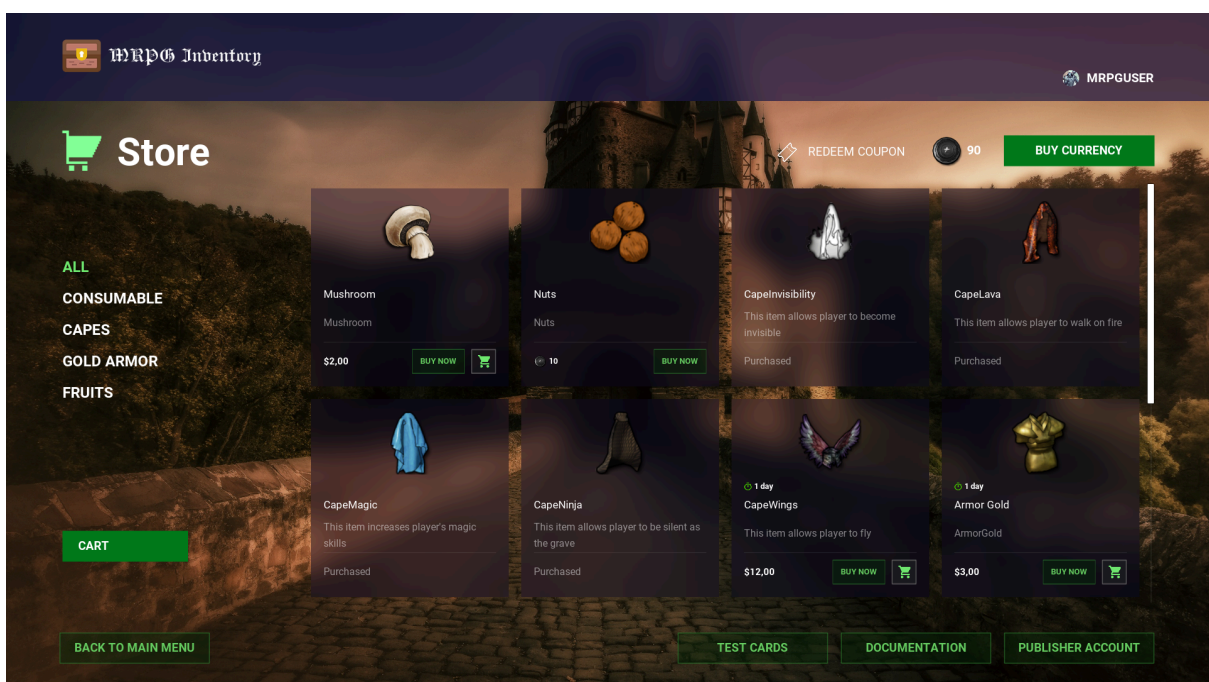
MRPG detects if it's properly configured showing a green question mark icon. Letting you know if the default MRPG Inventory is used or any other XSollus Store.

After successfully login to MRPG Inventory you will see menu with options:



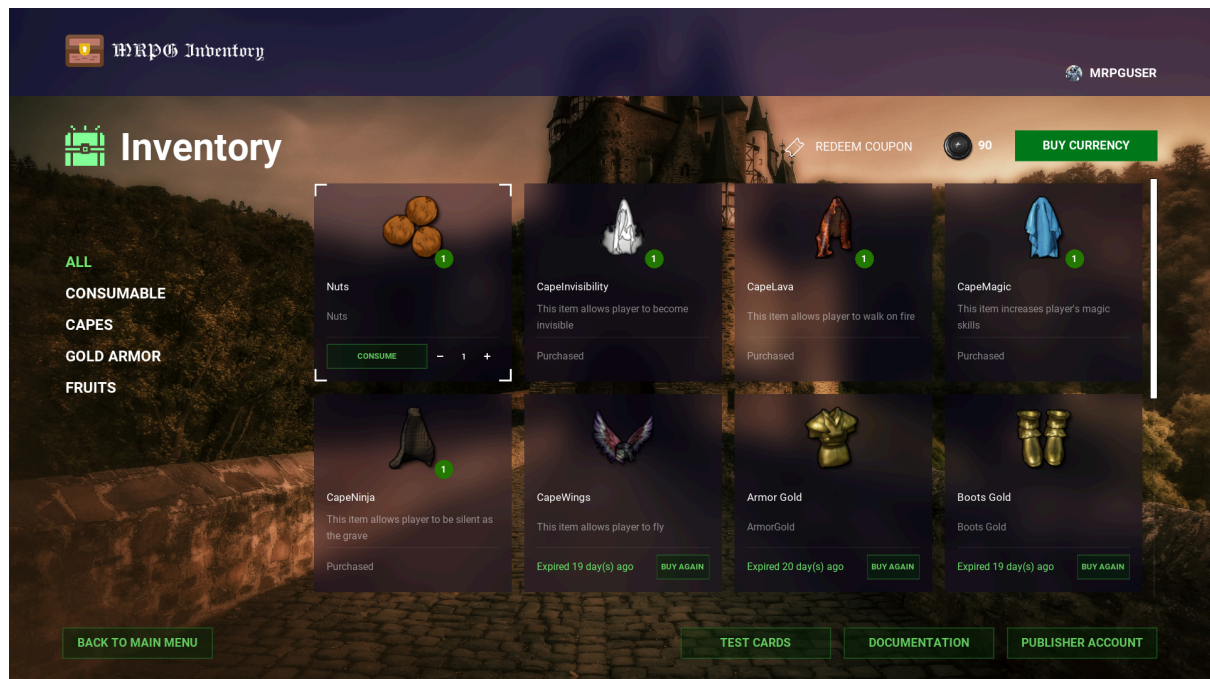
You may just simply experiment with all options and check online links on the right side!

- Store - offers you online transactions with your credit card
- Inventory - shows current items purchased
- User Account - your personal details
- Character - player progression system
- Friends - invites and player communication
- Log Out - if you don't want to stay logged in simply log out

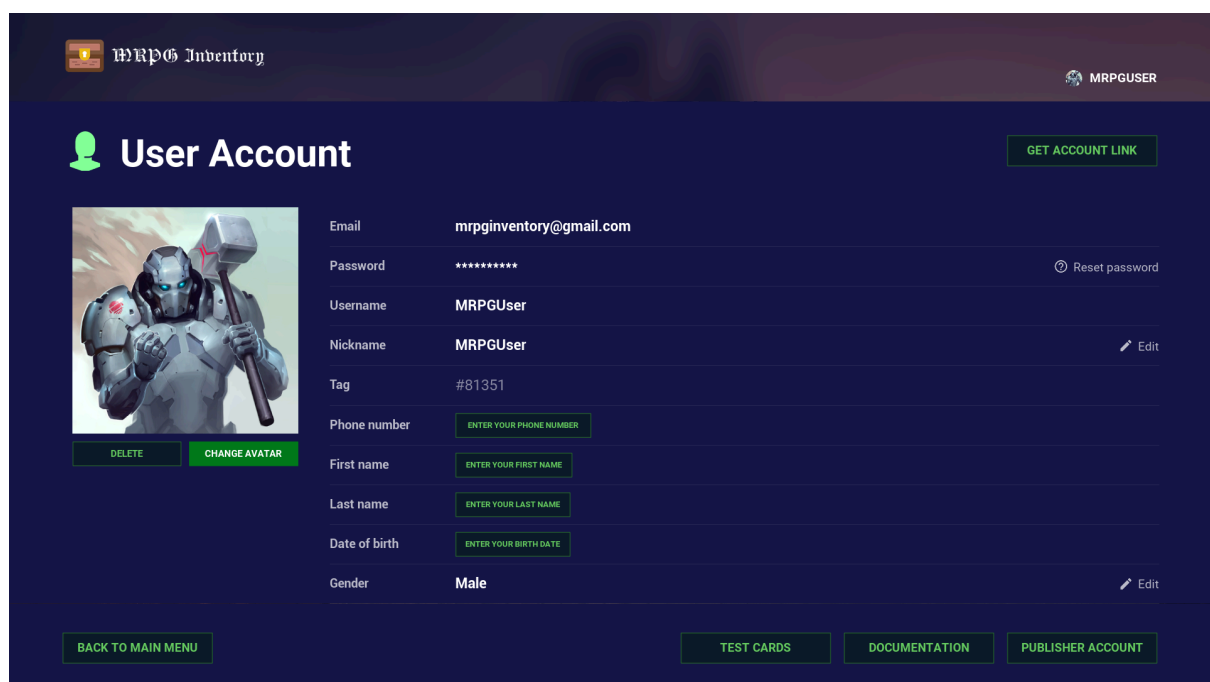
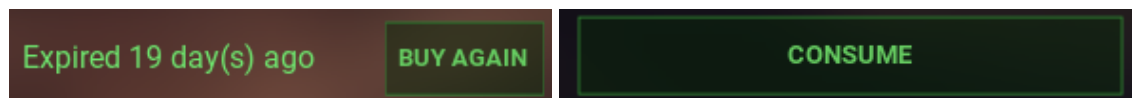


In the store you can preview all items which can be purchased. They are splitted into different categories. Some of them can be purchased multiple times like consumables and items which expire. You may purchase items using testing credit card number:

<https://developers.xsolla.com/doc/pay-station/references/test-cards/>



All purchased items are visible in the Inventory Section. You can preview the expiration time and consume items which are consumable.



User account provides basic information about you, **Nickname, Username, Tag, Phone Number, First Name, Last Name, Date of Birth** and your **Gender**. In the section you can also **reset the password** and change your **Avatar**!

Custom XSolla Store



To create your own XSolla store with custom items you need to create your own **XSolla Publisher Account** here:

https://publisher.xsolla.com/?utm_source=blueprints&utm_medium=mrpg-inventory&utm_campaign=in-game-store

• Creating new Project (Store)



ID 197098

Tefel

⚙️ Company settings

MAIN MENU

Dashboard

Accounting

Support

PROJECTS



ID 125761

MRPG Inventory

⚙️ Project settings

🛒 Store

📄 Subscriptions

🕒 Partner Network

👤 Login

🎮 Launcher

🎮 Pay Station

🛡️ Anti-fraud

🏗️ Site Builder

💰 Funding

To create a new Project and allow your Players to log-in directly there and having access to unique content created by press **PROJECT +**

Fill project name, supported languages and your website URL. Then press **CREATE**

CREATE

Your project should appear on the left side and allow you to select it.

After selecting your project you will have access to **Project settings** and other sections where you can configure your project.

To activate it in Unreal Engine 4 you need to provide two IDs:

- **LoginID** - is located in Company Settings -> API key

ID 197098

Tefel

⚙️ Company settings

MAIN MENU

Company settings

Company

Users

API key

- **ProjectID** - is located on the left top corner in the project section

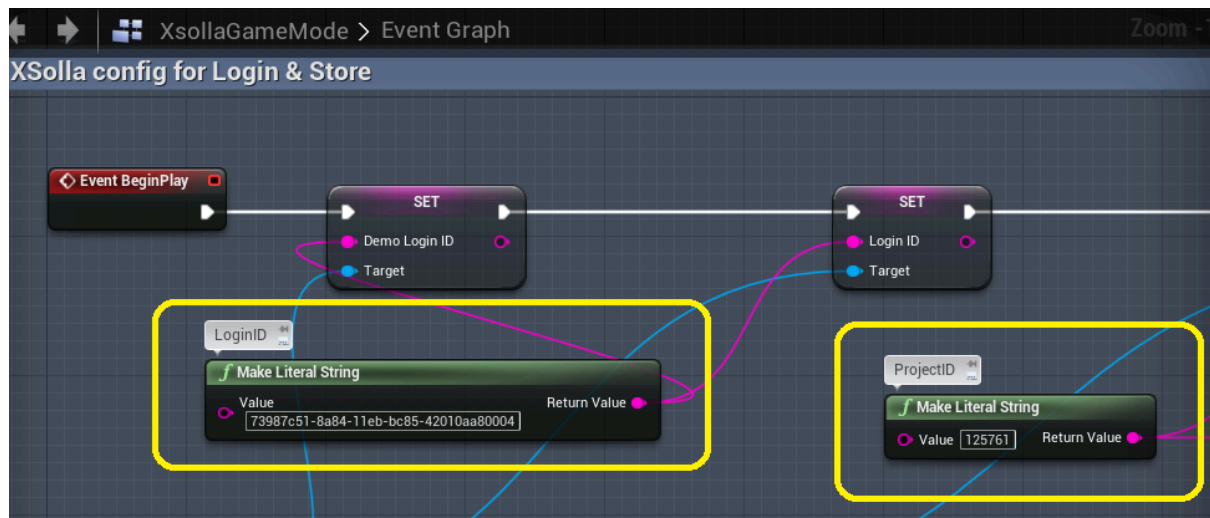
ID 125761

MRPG Inventory

⚙️ Project settings

After having them prepared, go back to UE4 for the final setup.

XSolla ProjectID & LoginID is part of the plugin configuration but MRPG Inventory is overriding it in **XSollaGameMode**



To log-in for the first time simply create a new player account. After verification through email you should be ready to go and access the main manu with new details and custom items!

The screenshot shows a 'SIGN UP' button on the left. To its right is a form with fields for 'Username', 'Email', and 'Password'. Below the 'Password' field, it says 'Minimum length: 6 symbols'. At the bottom, there is a checkbox for 'By continuing, you agree to be bound by our Privacy Policy' and a 'CREATE ACCOUNT' button.

• Items

To create your items in the store go to **Store Section**



Virtual Items

● READY

Sell in-game items that players purchase with virtual or real currency.

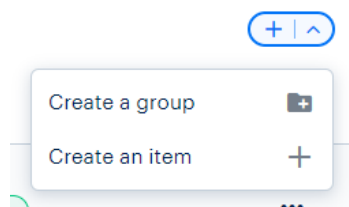
[API documentation](#)

[SDK documentation](#)

[Documentation](#)

CONFIGURE

Configure Store Virtual Items



Add new items or groups with a press button on the right side.

After finishing your items should be grouped into categories, where each category has some items in it.

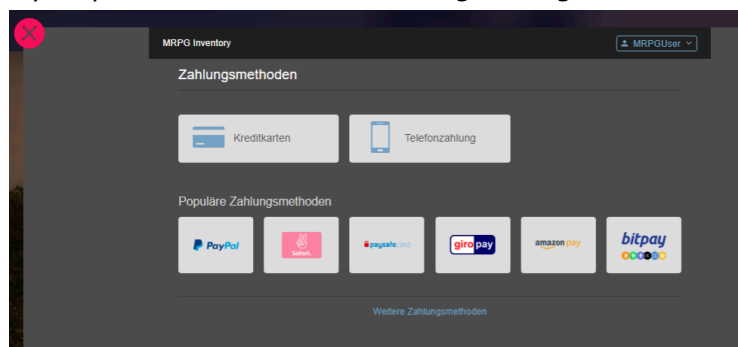
Virtual Items

Order	Name	External ID	Status
1	 Consumable Total items: 2	Consumable	Enabled v
2	 Capes Total items: 5	Capes	Enabled v
3	 Gold Armor Total items: 4	GoldArmor	Enabled v
4	 Fruits Total items: 5	Fruits	Enabled v

Only main currency is visible as default so pick wisely if Item should use Virtual or credit currency by default:

Order	Name	Price	Status
1	 Mushroom Mushroom	\$2.00	Enabled v
2	 Nuts Nuts	\$1.00 10 MRPGCoin	Enabled v

Try to purchase virtual items using testing credit card numbers.



Zahlung mit Kreditkarten

Kartennummer

4111 1111 1111 1111 ✓

Ablaufdatum

12 ✓ 40 ✓

Prüfnummer

1234 5678 ✓

☒ Diese Kreditkarte für zukünftige Zahlungen speichern

Mit der Übermittlung der Zahlungsdaten bestätigst, dass du mindestens 18 Jahre alt bist bzw. das je nach den Gesetzen in deinem Land geltende Mindestalter erreicht hast. Du bestätigst außerdem, dass du die Endbenutzer-Lizenzvereinbarung von Xsolla und sonstige geltende rechtliche Vereinbarungen gelesen und verstanden hast und diesen zustimmst. Bitte beachte, dass der auf der Abrechnung erscheinende Händlename wie folgt lautet: "Xsolla".

[< Zurück](#)

Summe €1,72

Jetzt zahlen

Bestellübersicht

 **Mushroom**
Mushroom

US\$2,00

Zwischensumme

US\$2,00 = €1,72

Summe

€1,72

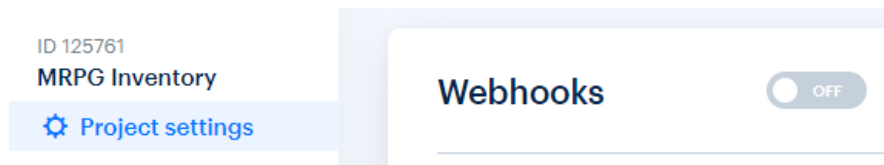
Der Gesamtbetrag enthält eventuell Wechselkursgebühren, damit die Transaktion vor Ort abgewickelt werden kann.

Testing card number 4111 1111 1111 1111 12/40 123

More test cards: <https://developers.xsolla.com/doc/pay-station/references/test-cards/>

If a transaction is successful everything works as intended.

If you cannot purchase the item it may require additional step - disable Webhooks



it's located in your project settings

- Virtual Currency

Virtual Currency

● READY

Sell packages of virtual currency that players use to make in-game purchases.

[API documentation](#)

[SDK documentation](#)

[Documentation](#)

CONFIGURE

Virtual currency is one option to buy virtual items. Whenever a player decides to buy something with virtual currency, the number of currency points is reduced.

You can decide how your players are getting virtual currency. If they have to purchase it or they are rewarded for some in-game events!

To define your own Virtual Currency press Configure button.

Virtual Currency

[Prices](#) [Packages](#)

[+ New virtual currency](#)

Name	Price	
 MRPGCoin	\$5.00	...

After defining **virtual currency** one should be available on the list. Remember that MRPG Inventory is fully supporting one currency and only one is visible when purchasing items from **Global Vendors**!

Virtual Currency

[Prices](#) [Packages](#)

[+ New package](#)

Name	Price	Status	
 SmallSilverPack (100 coins) 100 MRPGCoin	\$3.00	Enabled v	...
 BigSilverPack (500 coins) 500 MRPGCoin	\$10.00	Enabled v	...

Virtual **currency packages** are like normal items. They can be defined in the Packages section and purchased from **Global Vendors**.

● Promotions

Promotions

[+ Create promotion](#)

Status

All statuses

SEARCH

Found 3 promotions

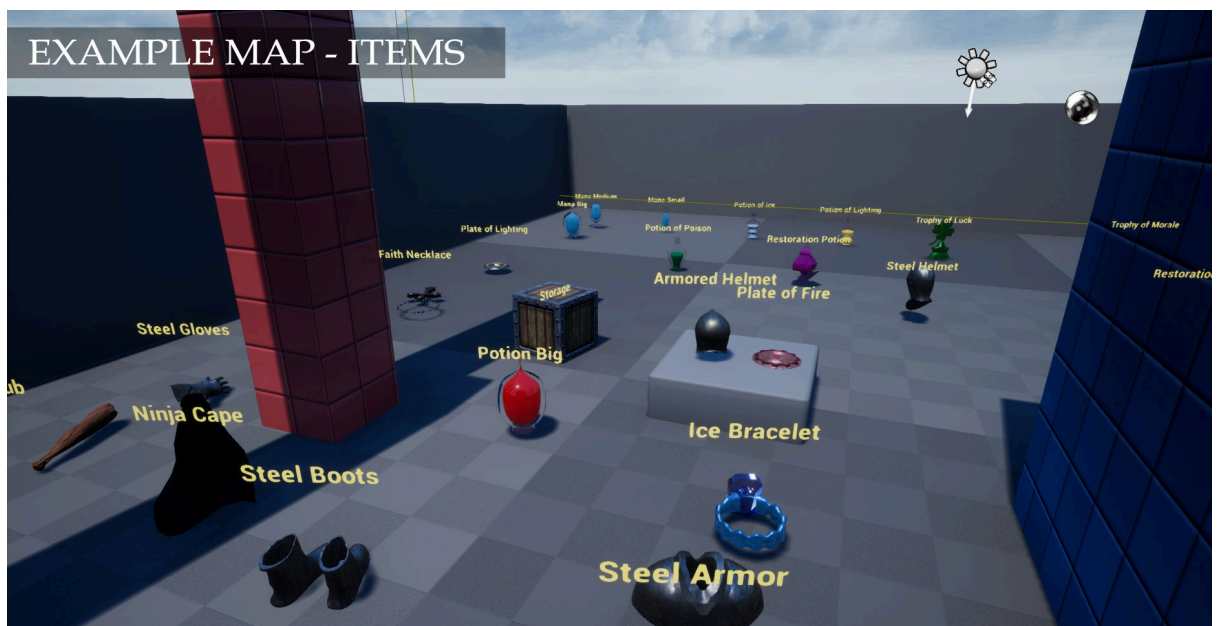
Promotion name	Type	From	To	Details	Status	
ID 26024 MRPG10DISCOUNT	Promo code	3/30/2021 12:00 PM	12/31/2021 11:00 AM	Discount amount 10%	Active	***
ID 26023 GoldPromo	Discount	3/30/2021 12:00 PM	12/31/2021 11:00 AM	Discount amount 10%	Not active	***
ID 26022 FruitPromo	Coupon	3/30/2021 12:00 PM	12/31/2025 11:00 AM	Good type Virtual items	Active	***

XSolla store allows you to control prices within the store. There are three ways of reducing prices:

- Discounts - temporary reducing prices of selected items in entire Shop
- Promo Codes - reduces prices of items just before purchasing them (entering code before doing transaction)
- Coupons - used from store menu, allow to receive some bonuses and additional discounts for items

World Actors

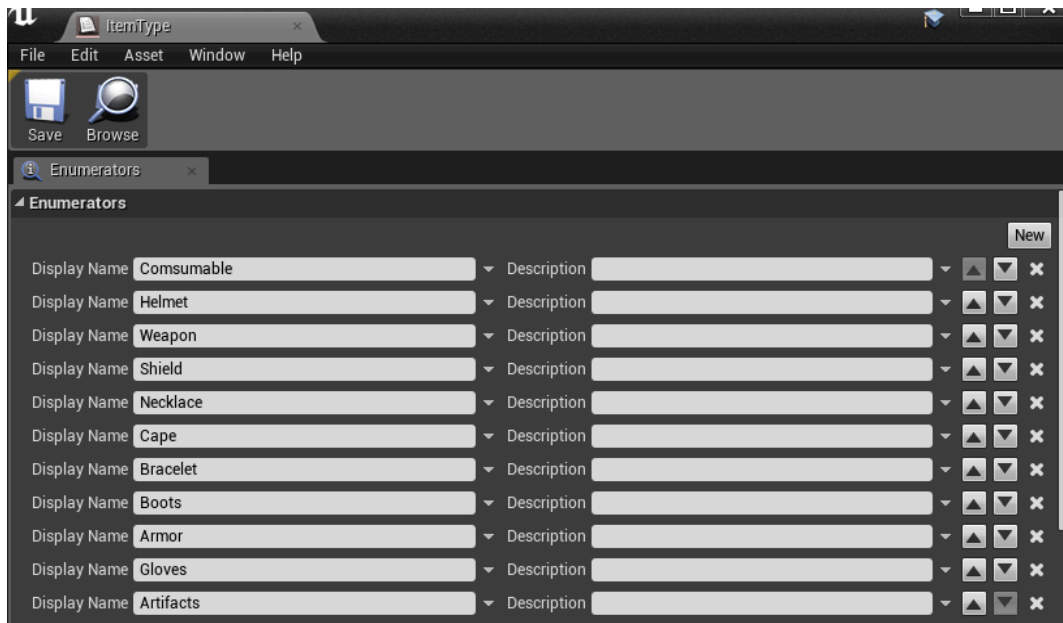
Every world actor is represented by StaticMesh and text.



● Pickable Items

Data about items are defined in **Inventory Manager Class Defaults**:

Item Type is a simple Enum type defining specific behaviour / slot of the item:



Every item is based on unique ID which is replicated and more properties are obtained from InventoryManager -> InventoryData[]
Inventory Data is a simple TArray which contains all information about existing items in the system.

Simple element from the array is a struct:



It defines UI 2dTexture, StaticMesh used for pickable objects, maximum quantity of item, Type, Title, Equipable Mesh (Static or Skeletal).

- **StorageLoot**

Storage Loot is a specific actor which can be placed in the world. It allows players to look into it and collect items from it or store items into it.

It inherits from Selectable Object which can be selected / invoked like PickableItem.

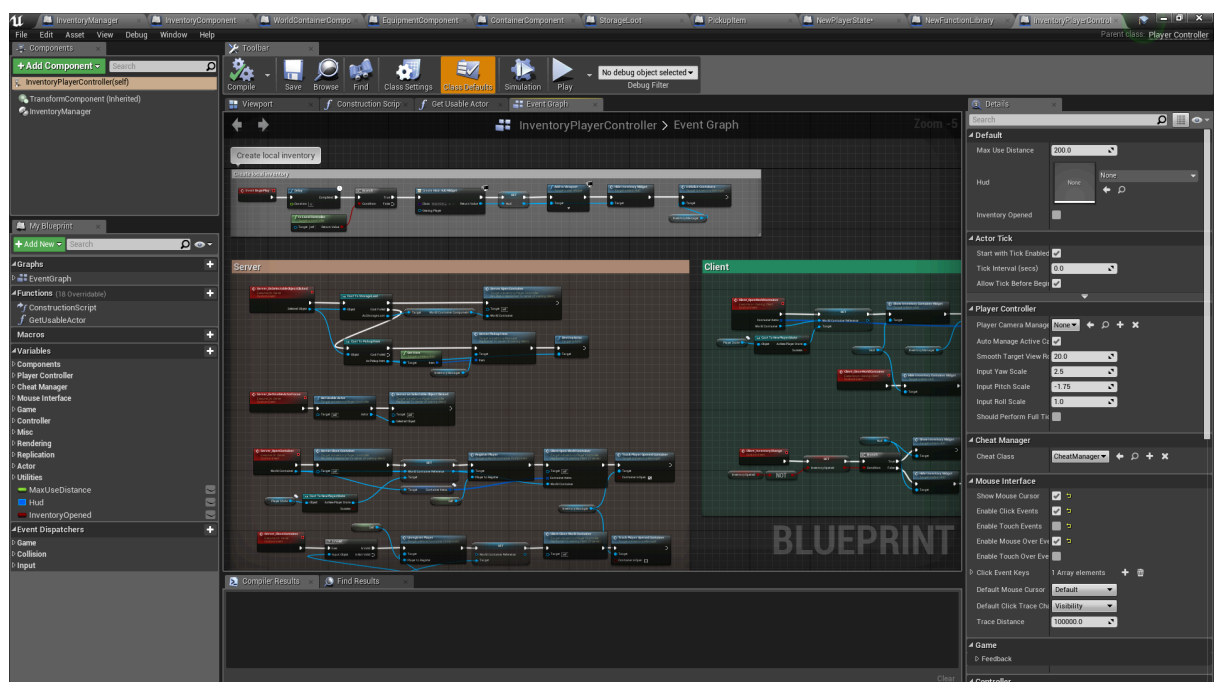
Multiplayer

The Inventory system was created to work in single player mode as well as multi player. At each step, the data being checked is synchronized between the server and clients. RPC calls tell clients what happened. An example of this is to pick-up an item from the ground, open a chest or move an item from one slot to another.

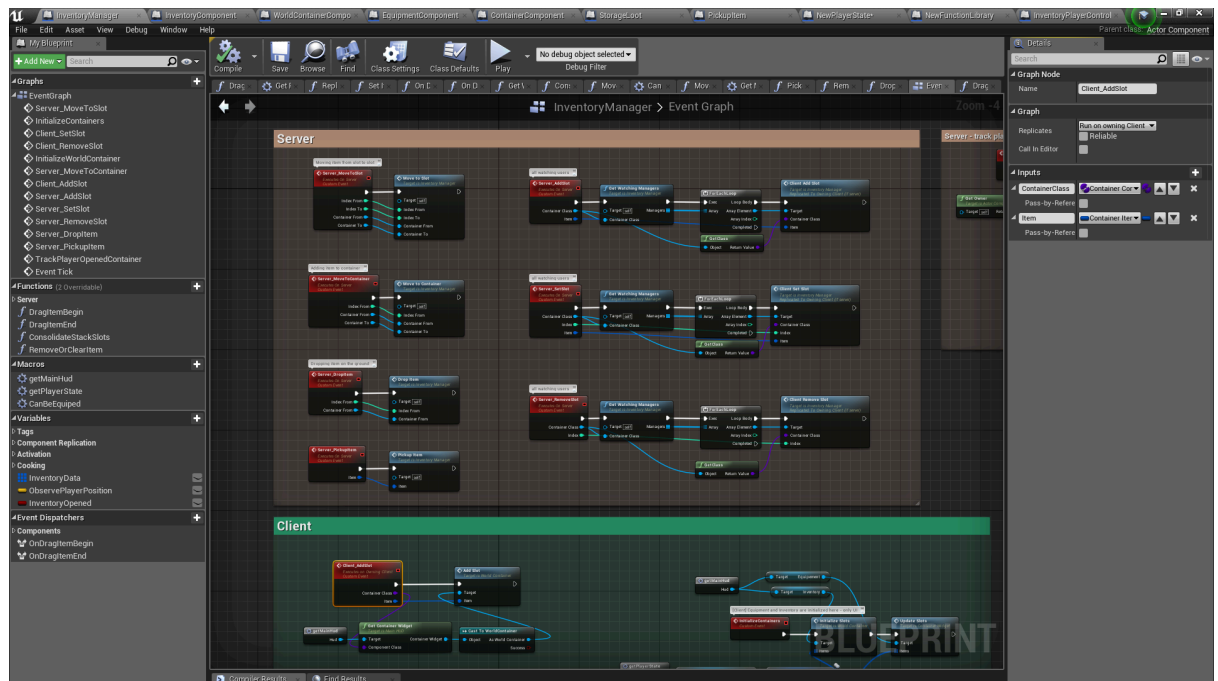
To replicate as little information as possible, a number of optimizations were used. Only observing clients will get information about the contents of the chest, while the others will not know what is in the chest until it is opened.

RPC calls were not enough to replicate the item-equipped state, so they were extended by OnRep calls.

- **InventoryPlayerController**



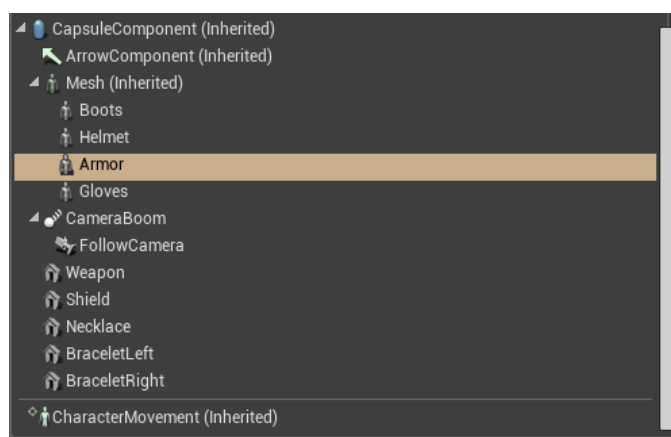
● InventoryManager



Inventory manager has all the logic for moving items between containers, picking them up from the ground or dropping them to the ground. It also manages slots and makes sure that slots which are modified are updated for all other clients.

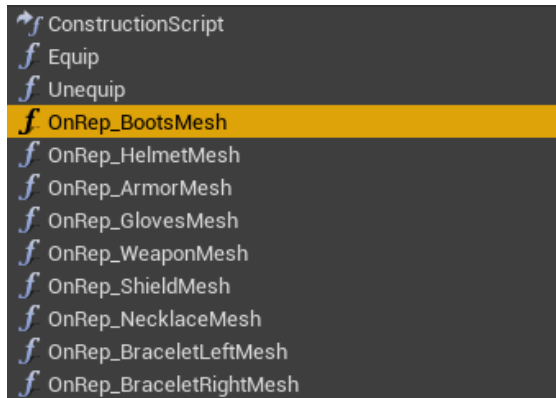
● InventoryCharacter

Specific type of Character which is controlled by User which contains logic for equipable items. It provides setup for Character (SkeletalMesh + AnimBP). It Also has specific StaticMesh and SkeletalMesh components for every item which is equipped.



Because the system has to work in Single and Multi Player modes, OnRep values are used. They let you know whenever one property was changed and the

skeletal mesh or static mesh of the character has to be updated. It happens in OnRep functions:

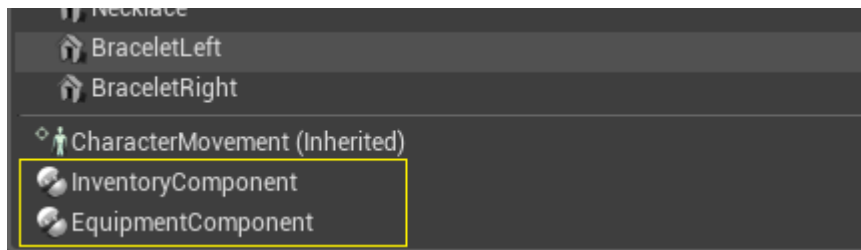


Functions OnRep just creates mapping between MeshComponent and Mesh variables - setting mesh for each.

- **ContainerComponent**

ContainerComponent is a specific ActorComponent which is replicated and contains items. It represents the state of items in specific containers like Character or StorageLoot.

Because Character requires two types of containers (Inventory and Equipment) that's why two components are added to InventoryCharacter:

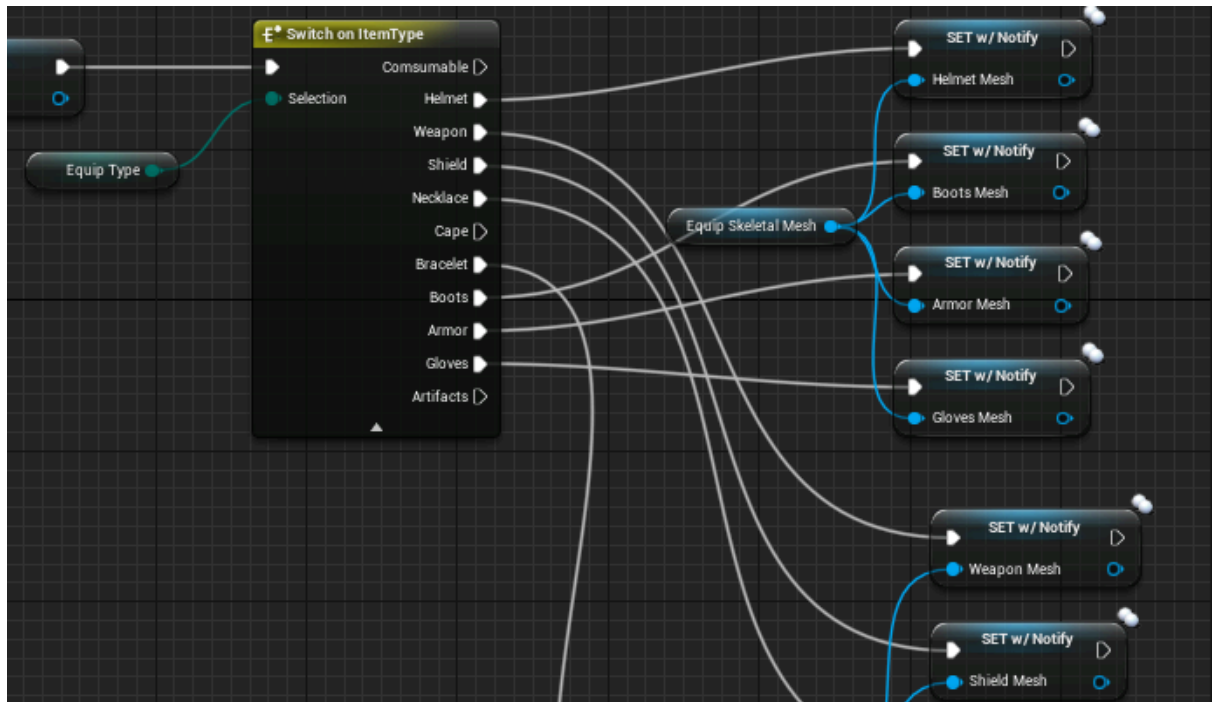


There are some Containers which derives from **ContainerComponent**:

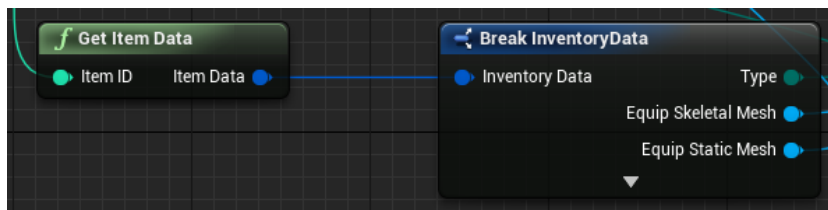
- **WorldContainerComponent** is a specific type of **ContainerComponent** which allows users to register and unregister watching players. Information is shared between other players over a Network. It is used in **StorageLoot**.
- **EquipmentComponent** used in Character (PlayerState) - keeps information about state of equipment and allows to initialize Equipment (equip all the items)
- **InventoryComponent** is a child of ContainerComponent which keeps the state of Character Inventory items.

Equippable Items

Equippable means that every item which is equipable (has specific EquipType)

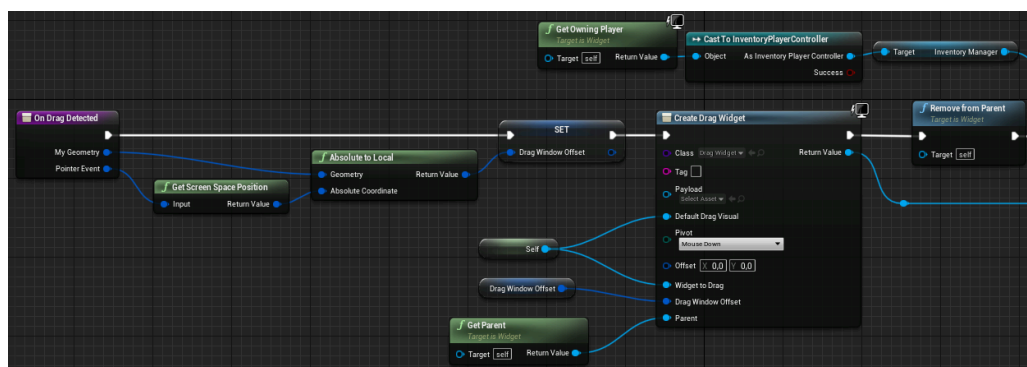


It also contains information about specific Mesh to equip - SkeletalMesh or StaticMesh. Based on that values item will be equipped or unequipped and OnRep function fired.

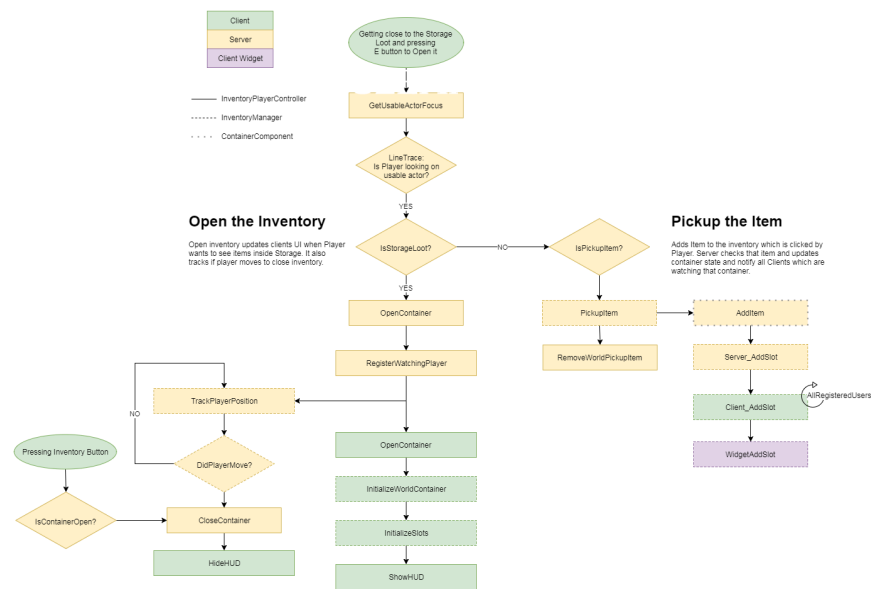


Draggable Windows

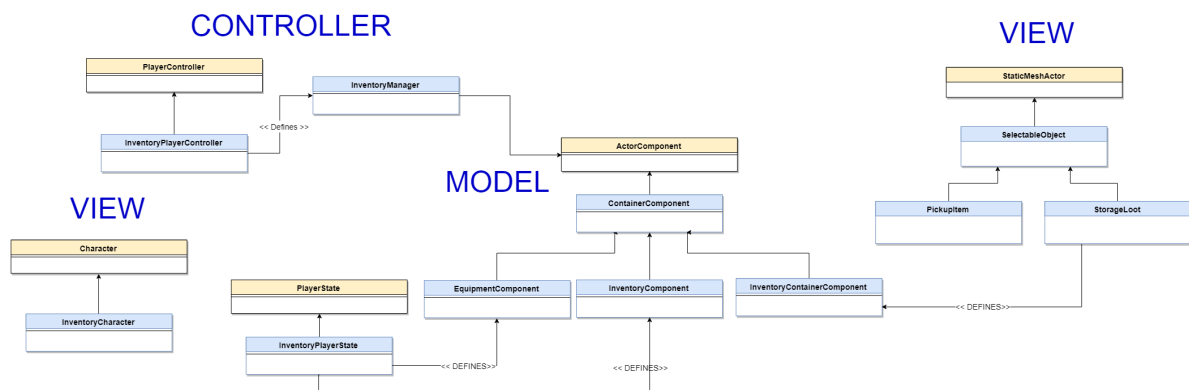
Every draggable window inherits from WBP_Draggable which implements OnDragDetected. This allows to move all windows on the screen.



Plugin Structure



The architecture of Inventory System represents classic MVC structure which is simply represented by **InventoryPlayerController**, **InventoryCharacter** and **UUserWidgets**. Model-view-controller (MVC) is a software architectural pattern commonly used for developing user interfaces that divides an application into three interconnected parts. This is done to separate internal representations of information from the ways information is presented to, and accepted from, the user. The MVC design pattern decouples these major components allowing for efficient code reuse and parallel development.



- **Model** – represents problems and implements logics for Inventory System
- **View** – defines User Interfaces (Widgets) and contains only references to the model but never implements any logic
- **Controller** – gets input data from the user and reacts to it. For example when user try to pick-up inventory Item, moves items inside inventory or drop Item on the ground