ASM for NGO

This example was built with NGO(Netcode for gameobjects) 1.5.1, things may have changed.

Intro

There are 2 ways to work with ASM and NGOs.

First is to make use of the Netcode built-in scene manager, this will give us NetworkManager, Events, Scene, and object synchronization, and all nice things.

The second would be to disable the built-in scene manager, this would give you a lot more freedom and flexibility, but at the cost of having to implement everything yourself. That means you need to make a new NetworkManager, NetworkSceneManager, Network sync, and NetworkObject.

This example is for the first option with the netcode plugin from ASM, If you are going for option 2, I wish you luck.

Rules to follow

1. In NGO the client syncs the scenes, meaning the ASM workflow will not be applied on the client side. It's the NGO workflow that loads the scene, you will utilize ASM on the server side.

2. ASM still works if you wish to work with offline scenes on the client side. Like loading a UI pause screen during gameplay for example. Or return to the main menu. It's just the netcode scenes that will be loaded by netcode and not ASM.

3. Preload and loading screens will not be synchronized, you have to create the sequence yourself, an example of this can be found in "ConnectionManager.cs" where we have a sequence for client connect and host connect. And it's simply because the client syncs the scene not load it themself.

4. This plugin only syncs the scenes marked as Netcode, if you wish to change this rule, assign a new validator to "networkManager.SceneManager.VerifySceneBeforeLoading", this is the main method for handling which scenes sync.

5. Method marked as netcode cannot be loaded when offline.

This project is a good start, but just an example. Feel free to use it. Feel free to experiment, this is not a "must-do" guide.

Things to consider:

1. You might want to send a signal to the clients before loading a scene, like… "we are about to load a new scene, prepare your loading screens and report back when ready." then the client can use "LoadingScreenUtility.OpenLoadingScreen(loadingScene);" on the client before the sync starts.
2. This project used client-side auth, simply to be easier to make, you do not want that. So treat all players as a client.
3. This project is in no way the right way to do things, you may have to figure out the gameplay yourself. This is but a showcase that it works and an idea of where to go next.

Good luck.