Innoactive Portal Access Control Plugin for Unreal Engine 4.23+

An Unreal Plugin that attempts to authorize the application use via the Innoactive Portal Desktop Client (formerly known as Hub Launcher).

Usage

Installation

1. Download the latest release.
2. Create your own project or navigate to your existing project
3. IMPORTANT: Make sure the project is a cpp project and has a Visual Studio solution, otherwise it will fail (You need to create a cpp class to force the creation of a Visual Studio solutions.
4. Make sure your project is currently closed
5. Go to your project root folder and create a Plugins folder
6. Unzip the InnoactivePortalAccessControl.zip into the newly created plugins folder. **Important**: The plugin's folder must be named *InnoactivePortalAccessControl*
7. Start the project. Wait for the plugin to compile.
8. If everything went well the plugin content should be displayed now.

Setup & Configuration

NOTE: If the Innoactive folders do not show up make sure you have set the view options to show plugins.

1. The plugin content contains an *AuthorizationScene* level that can be used as a starting point but most likely, you'll want to create a dedicated and customized scene to perform authorization. Either way, the functionality of the *AuthorizationScene* must be maintained.

In case the built in *AuthorizationScene* does not fit your needs, you can build your own like this:

1. Create a new level (Ideally choose an empty level)
2. Drag and drop the `InnoactivePortalAccessControlHandler` Blueprint into the level
3. Enter the Client Id and type the name of your main scene into “Loaded Level On Success”. Also, make sure to create a level in case of failed authorization to inform the user abut the problem (missing authorization) and reference the level's name as “Loaded Level On Error"

Building

No matter if the default or a customized *AuthorizationScene* is being used, the authorization scene must be the default map / start level for any build! To ensure this, go to the project settings (Project Settings -> Maps and Modes -> Default Maps).

Also make sure that if you are using any of the default scenes from the plugin, they are included in the "List of maps to include in a packaged build" list. You can check this by going to the "Project Settings" > "Packaging" section

The dlls found in the library folder of the plugin are copied over automatically but you should double-check after compiling that they have been copied over successfully.

API

Blueprint Methods

```
void ReleaseResources();
```

Releases all resources created by the plugin, should be called on closing the application

```
void StartAuthorization(FString clientId);
```

Type	Info
Description	Handles auth for user
Parameters	clientId -> The client id provided in the Innoactive Portal
Returns	

```
FString GetCurrentLogMessage();
```

Type	Info
Description	Returns a string that can be printed to a UI widget to inform the user about the current status of the auth process
Parameters	
Returns	Info string

```
void OnAuthorizationSuccessful();
```

Event that is called when the auth process has finished successfully

```
void OnAuthorizationFailed();
```

Event that is called when the auth process has failed

```
showLogs
```

Public flag that determines whether to show logs on screen or not

```
useMultithreading
```

Public flag that defines if a second thread is used to authenticate the user