SkillsVR Enterprise Cloud SDK

v1.1.0

About Enterprise Cloud (EC) SDK. The EC SDK will power you to develop with the EC Portal and update scenarios and analytics via simple methods inside Unity..

This is a link to the Git Repository for EC SDK:
git@github.com:joybusinessacademy/EC_SDK.git

This is a link to the Documentation for EC SDK:
https://docs.google.com/document/d/12gh5Lf4u-pFcDlzWSsSaO81dcr0sKlf8ps6BOVuKSuQ

Unity Package Manager Install Git Url
git@github.com:joybusinessacademy/EC_SDK.git#1.1.0

This is links to the EC Portal:

- Development Portal: https://develop-ec.skillsvr.com
- Internal Portal: https://internal-ec.skillsvr.com.

If you don't already have a Unity project and just want to see what the EC SDK is about, check out the video below. (Show video or Sample of EC)

Prerequisites
Unity Version: 2019.4.x or later. Earlier versions may also be compatible but will not be actively supported.

Table of Contents

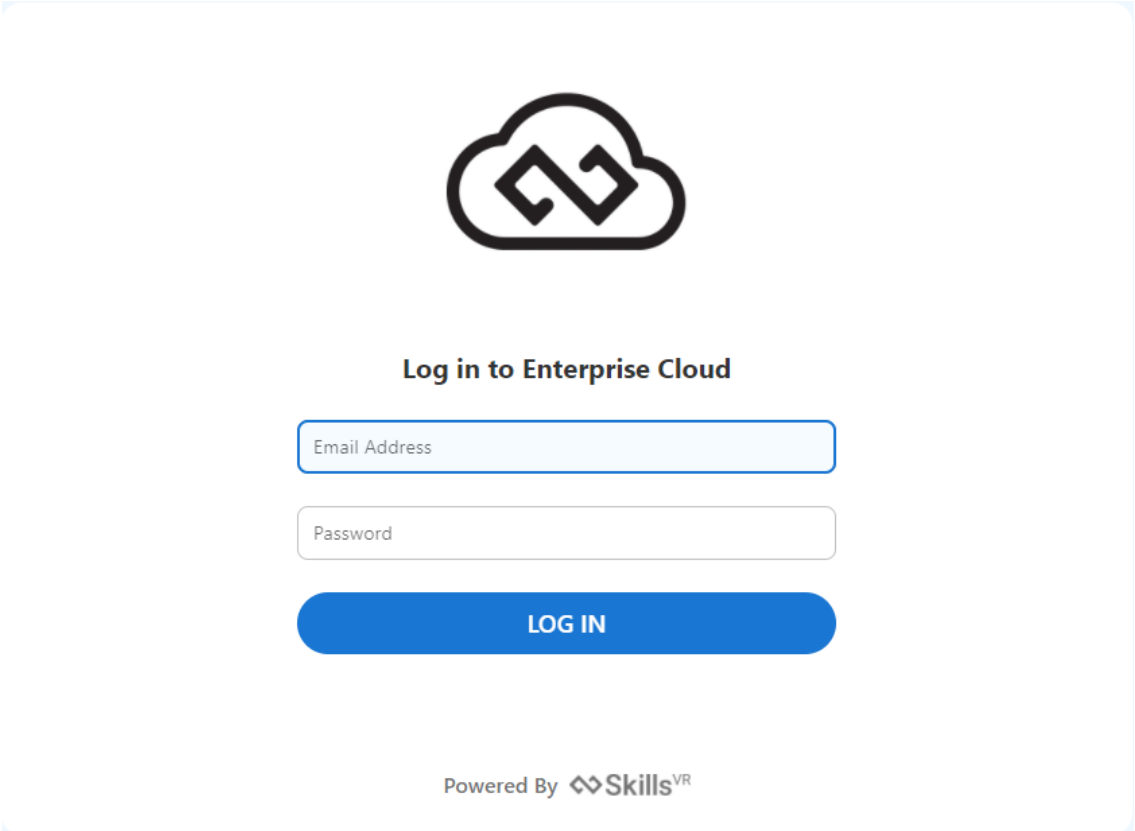
Step 1: Setup Scenario at EC Portal

1. Create/Login to Enterprise Cloud.

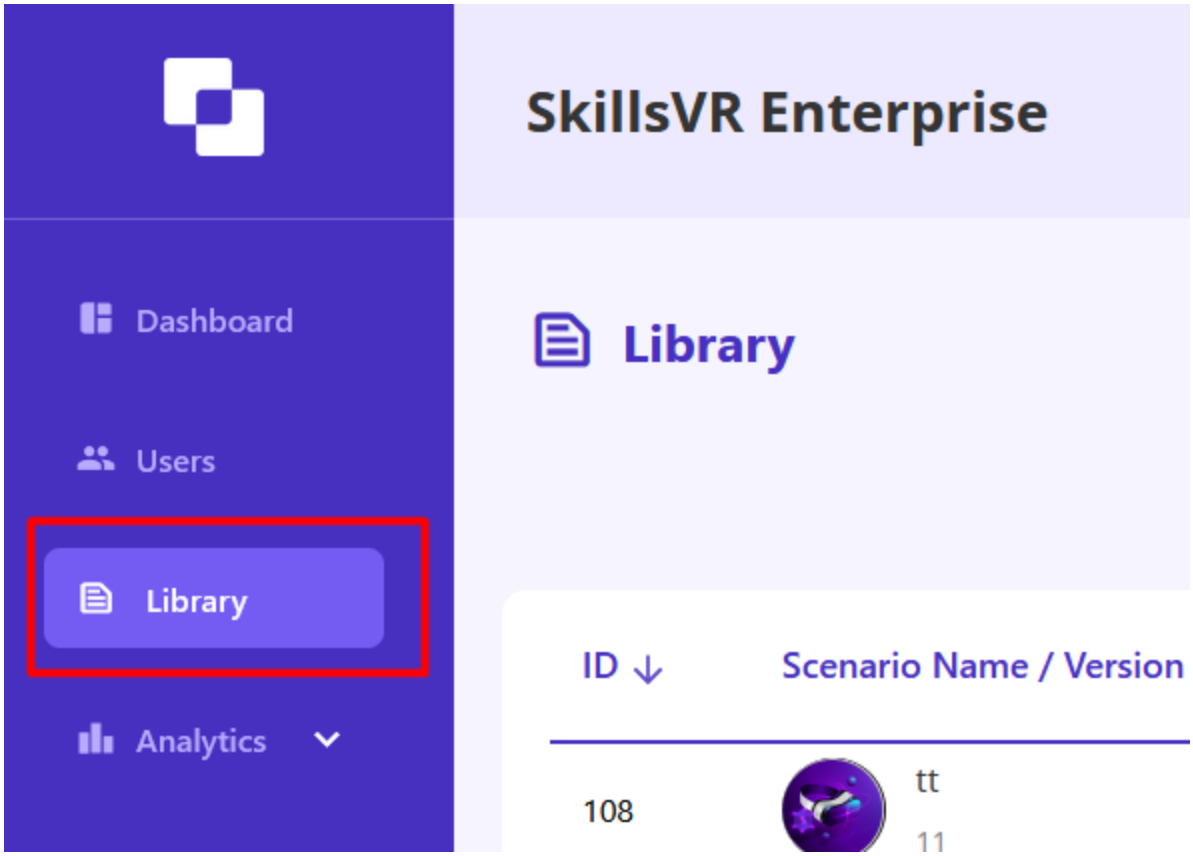
For development portal, go to https://develop-ec.skillsvr.com.
For internal portal, go to https://internal-ec.skillsvr.com.

IMPORTANT: Account, Settings and Scenarios are not shared between different portal environments.

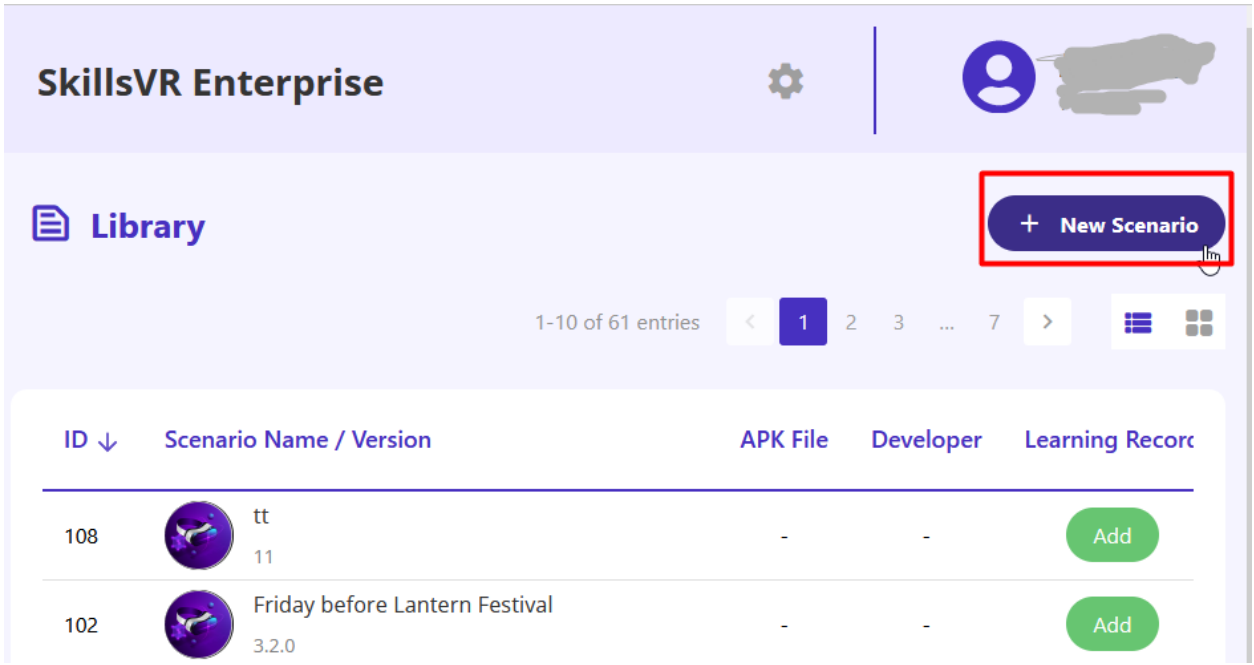
v1.1.0 : staging and product website not online yet.



2. Select “Library” button at left side menu;

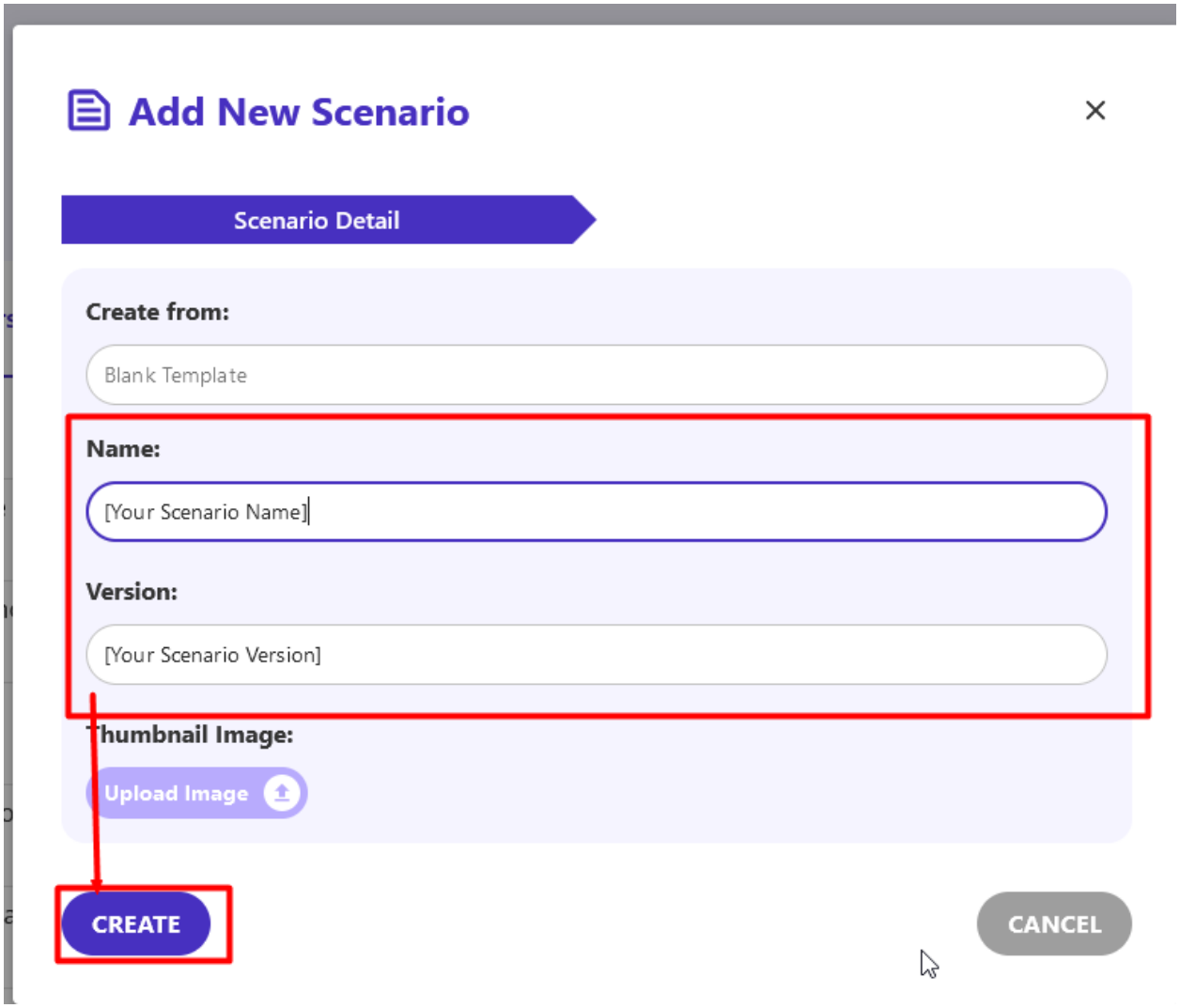


3. Click “+ New Scenario” button at right side;



4. In the popup window, type your scenario name and version, then click the “Create” button. In this tutorial, a test scenario named “EC Test Scenario” with version “1” will be

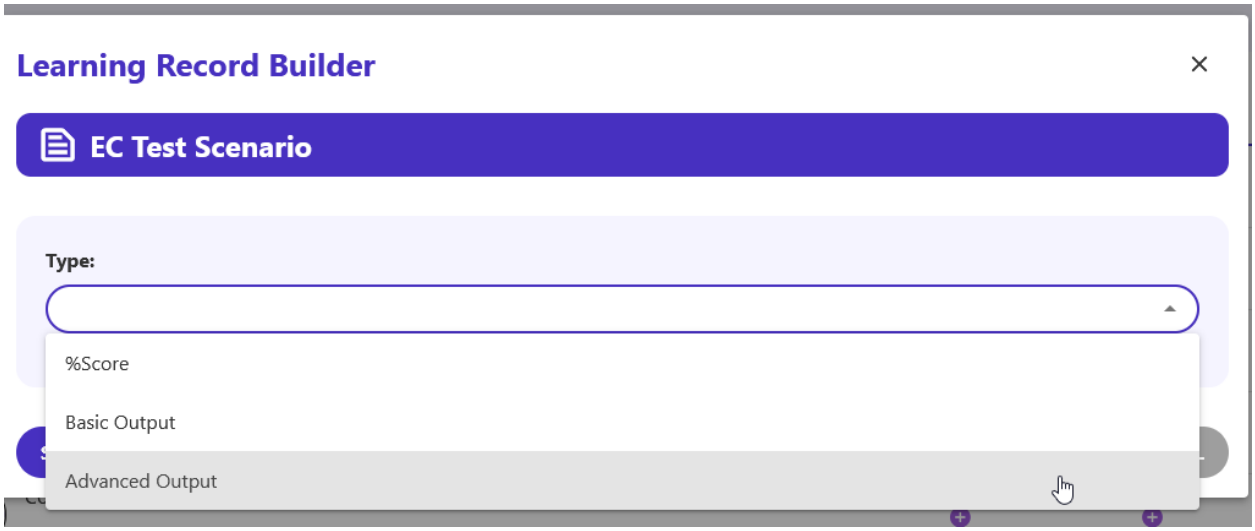
created as example.



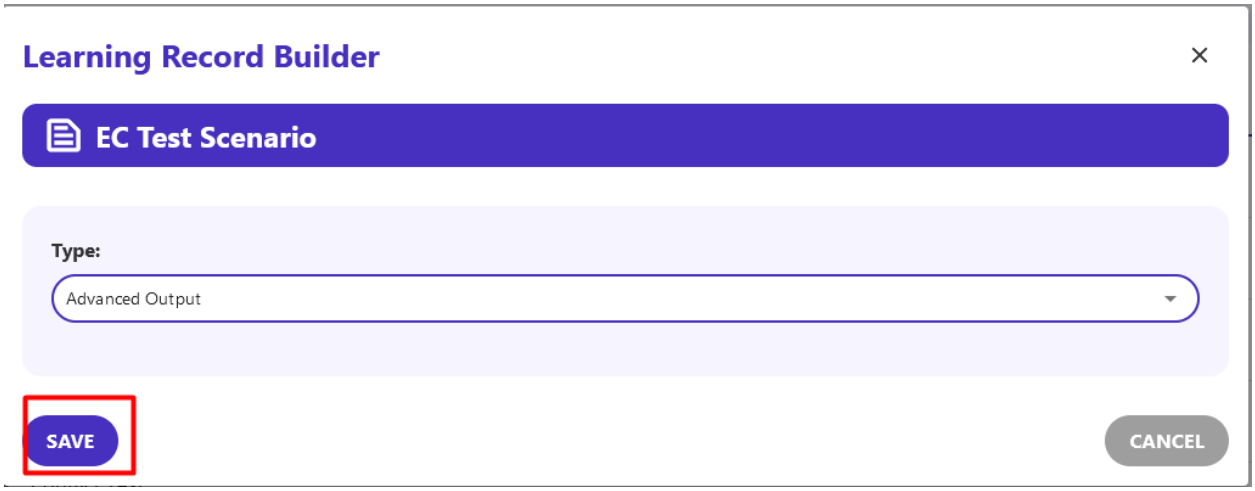
5. After the new scenario listed in Library, click the “Add” button;

6. In the popup window, click the “Type” dropdown and select output type:
   - Basic Output - 2 levels record structure;
   - Advanced Output - 3 levels record structure.

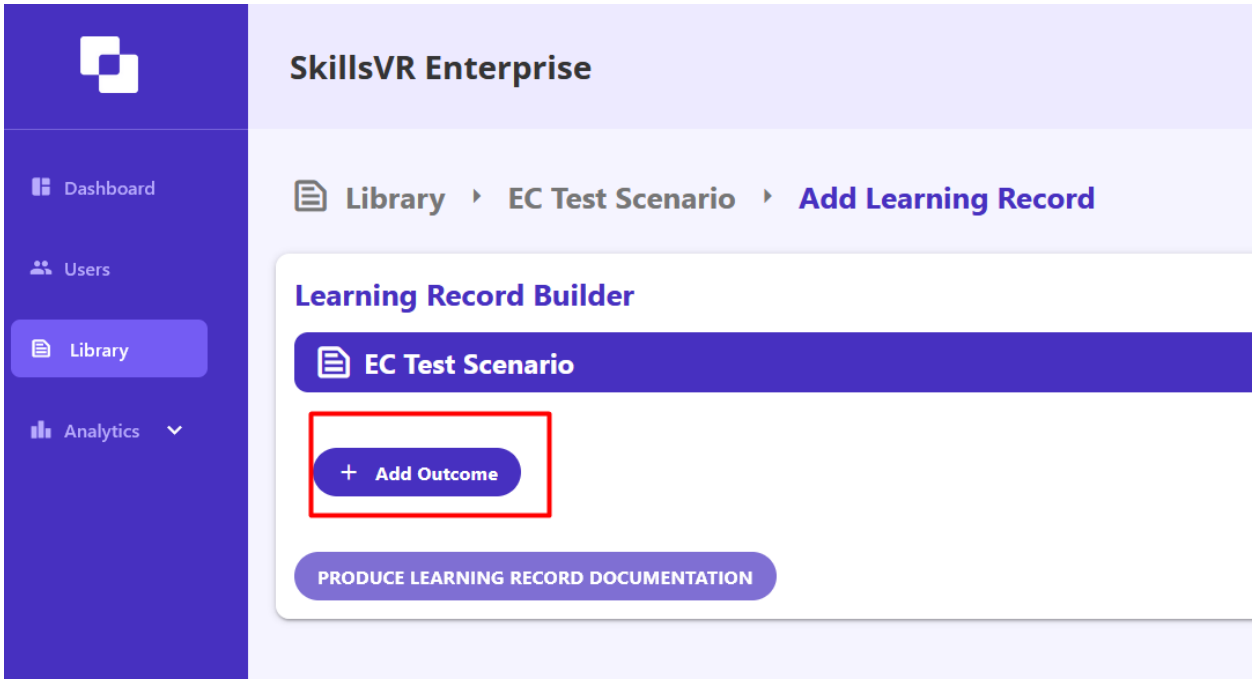
In this tutorial, an “Advanced Output” will be created;



7. Click “Save” button;

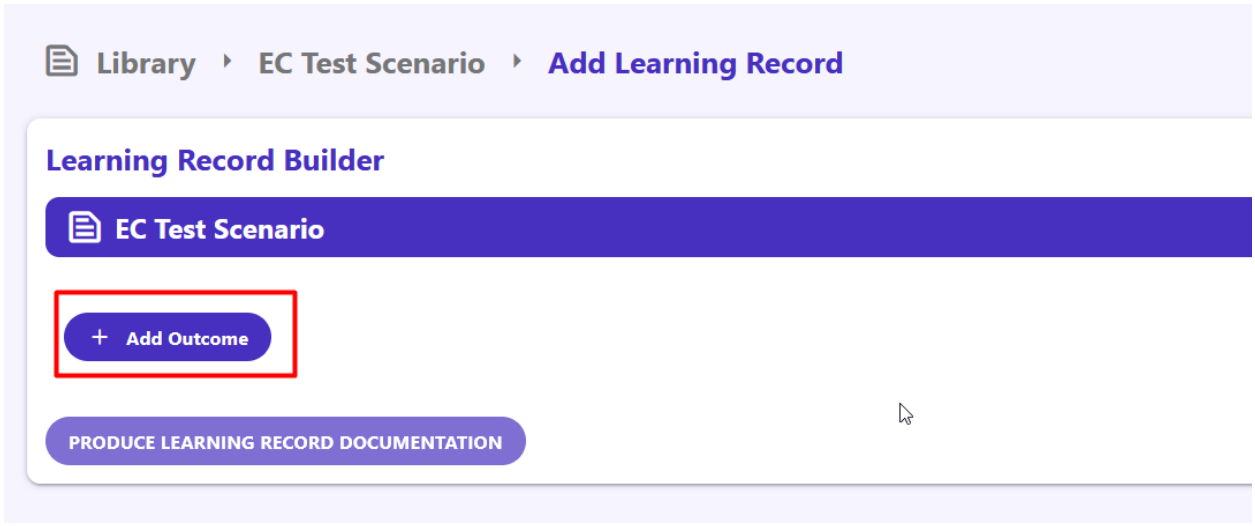


8. After save scenario type, the web page will auto jump to Add Learning Record;

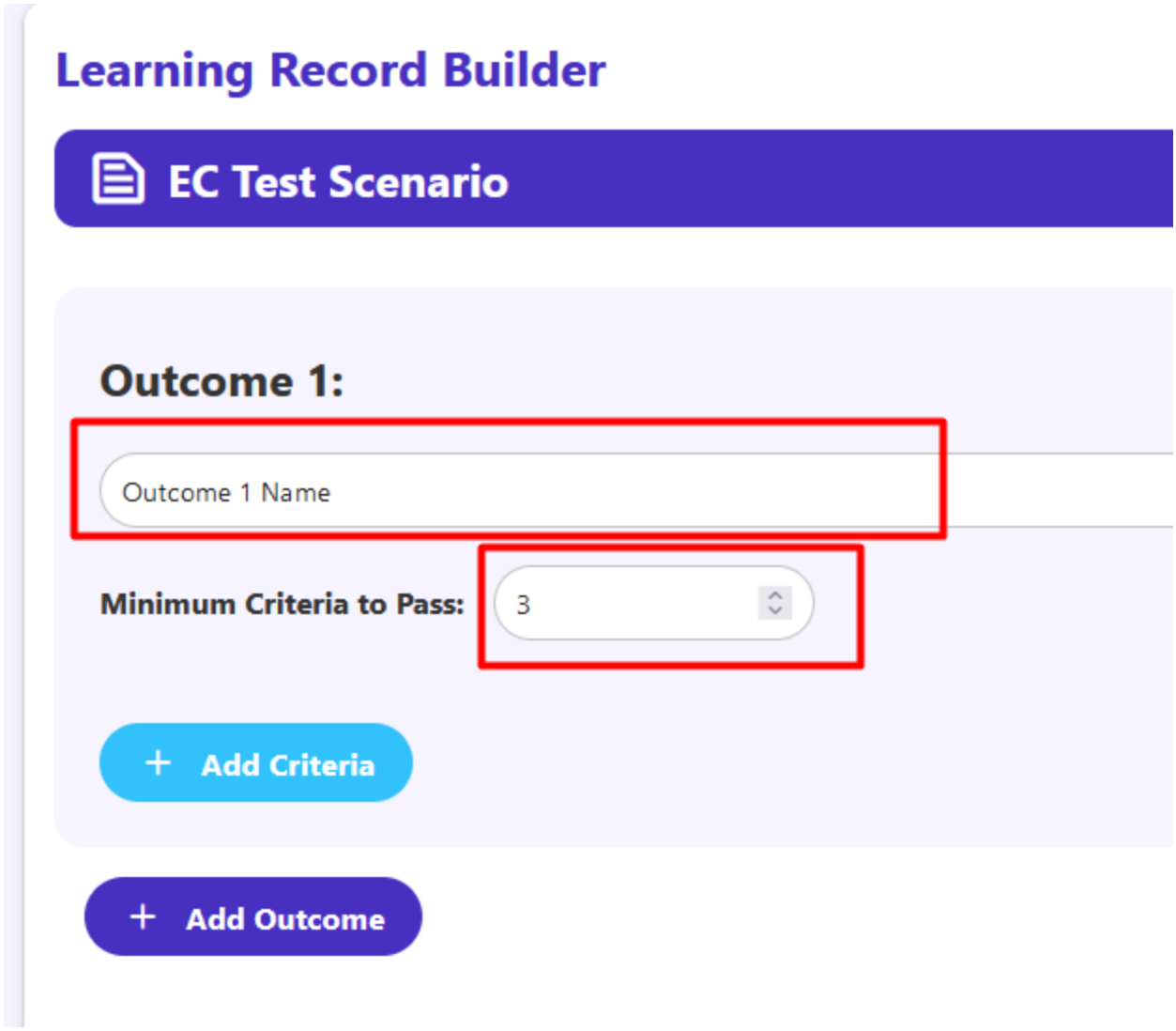


To manually open the Add Learning Record page for next time, click the “View” button in “Learning Record” column.

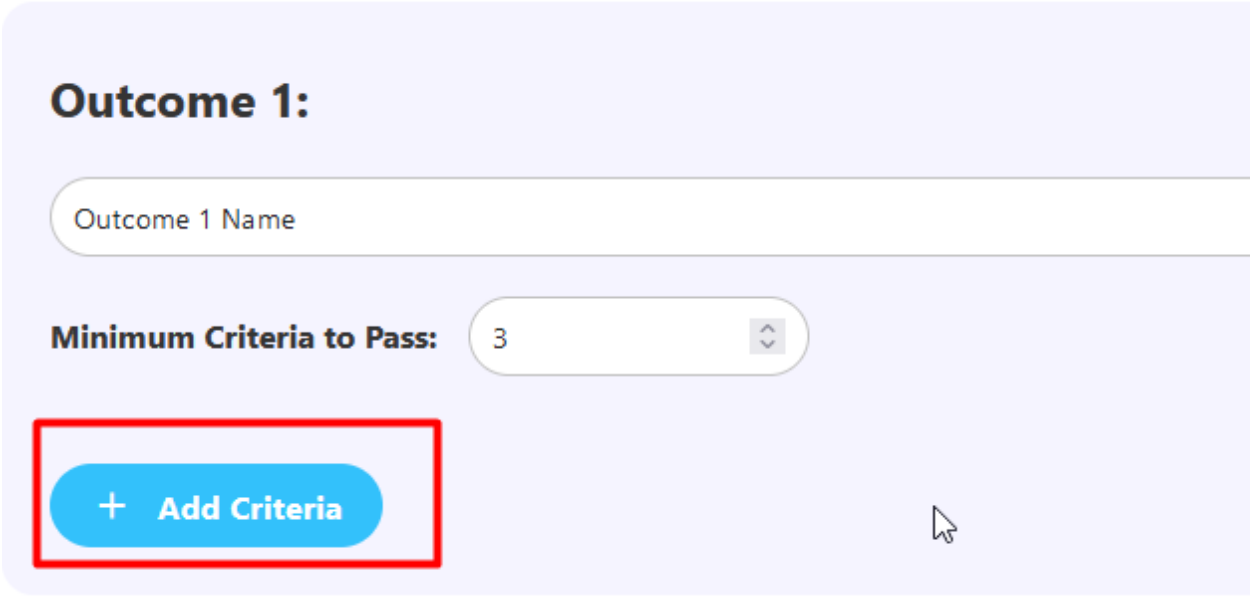
9. To add a new outcome, click the “Add Outcome”



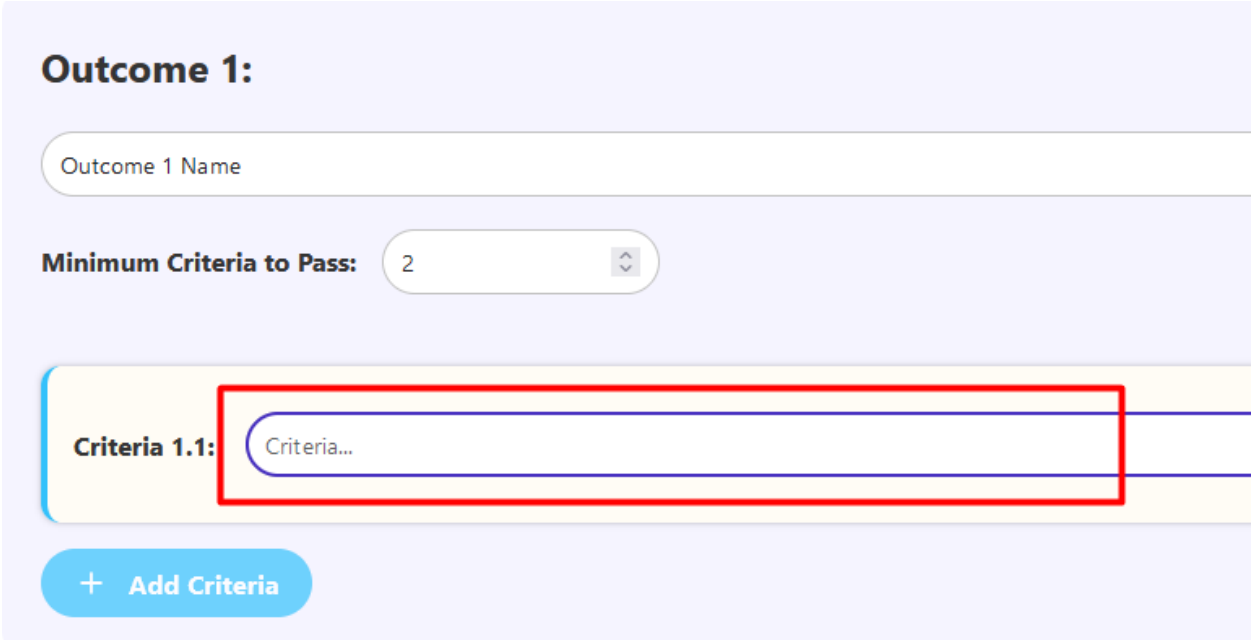
10. Then fill the outcome name filed and “Minimum Criteria to Pass” field;
    - Name - The name of outcome;
    - Minimum Criteria to Pass - the minimum number of criteria must to pass to before current outcome passed


Learning Record Builder
EC Test Scenario
Outcome 1:
Outcome 1 Name
Minimum Criteria to Pass:
3
+ Add Criteria
+ Add Outcome

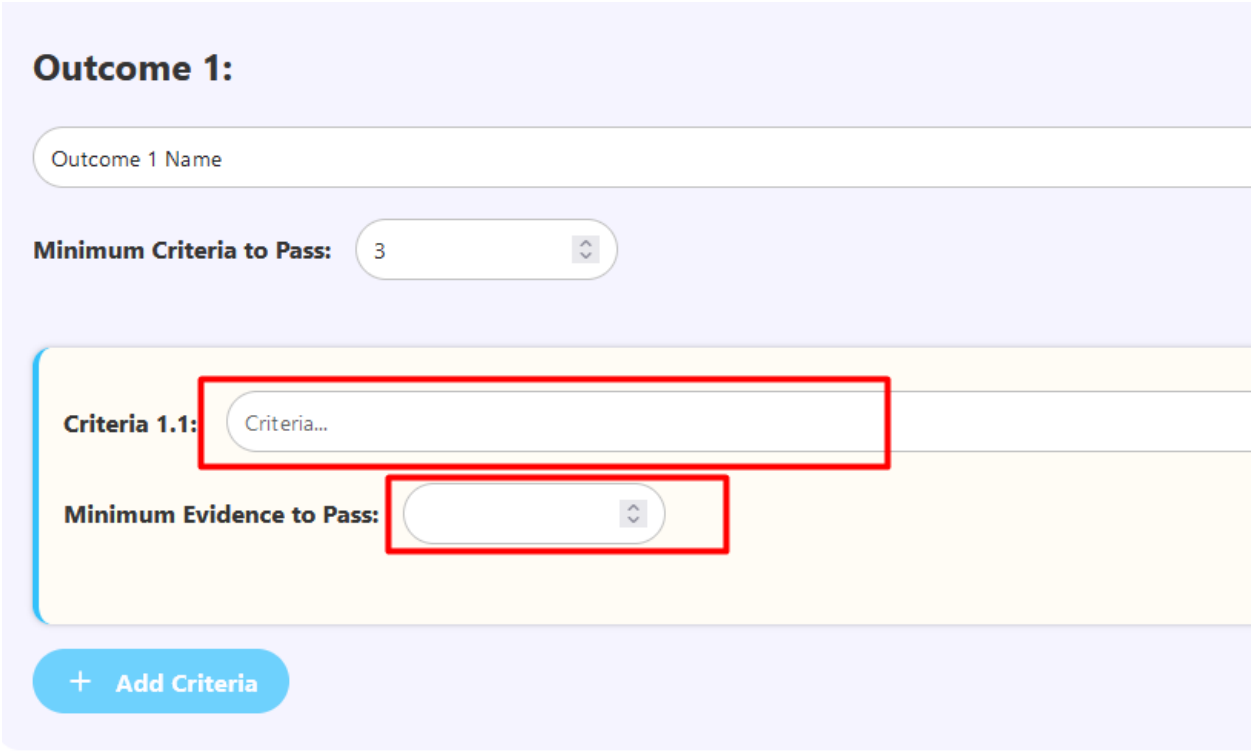
11. To add a criteria, click the “Add Criteria” button;



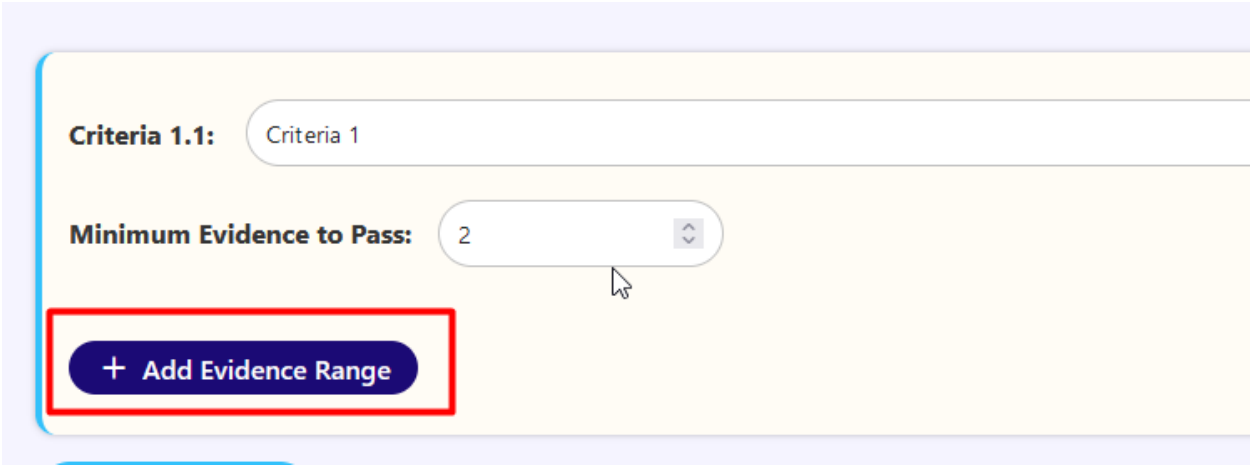
12. For basic output type, fill the criteria name field;

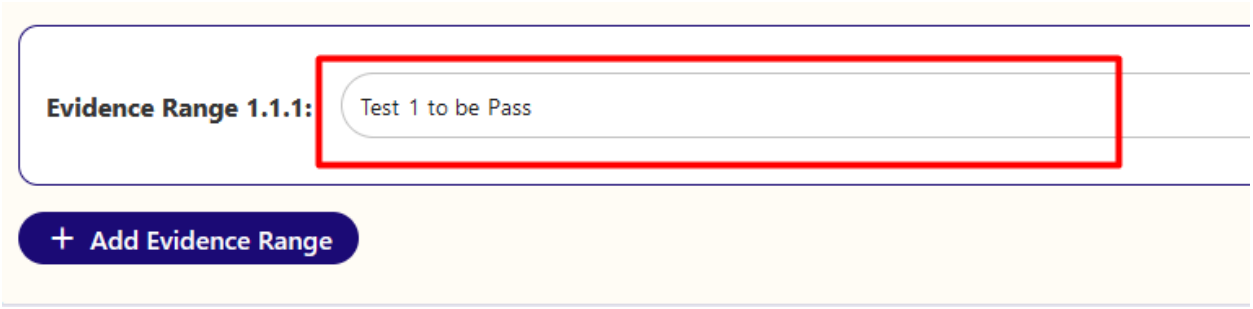


For advanced output type, fill the criteria name and “Minimum Evidence to Pass” field;



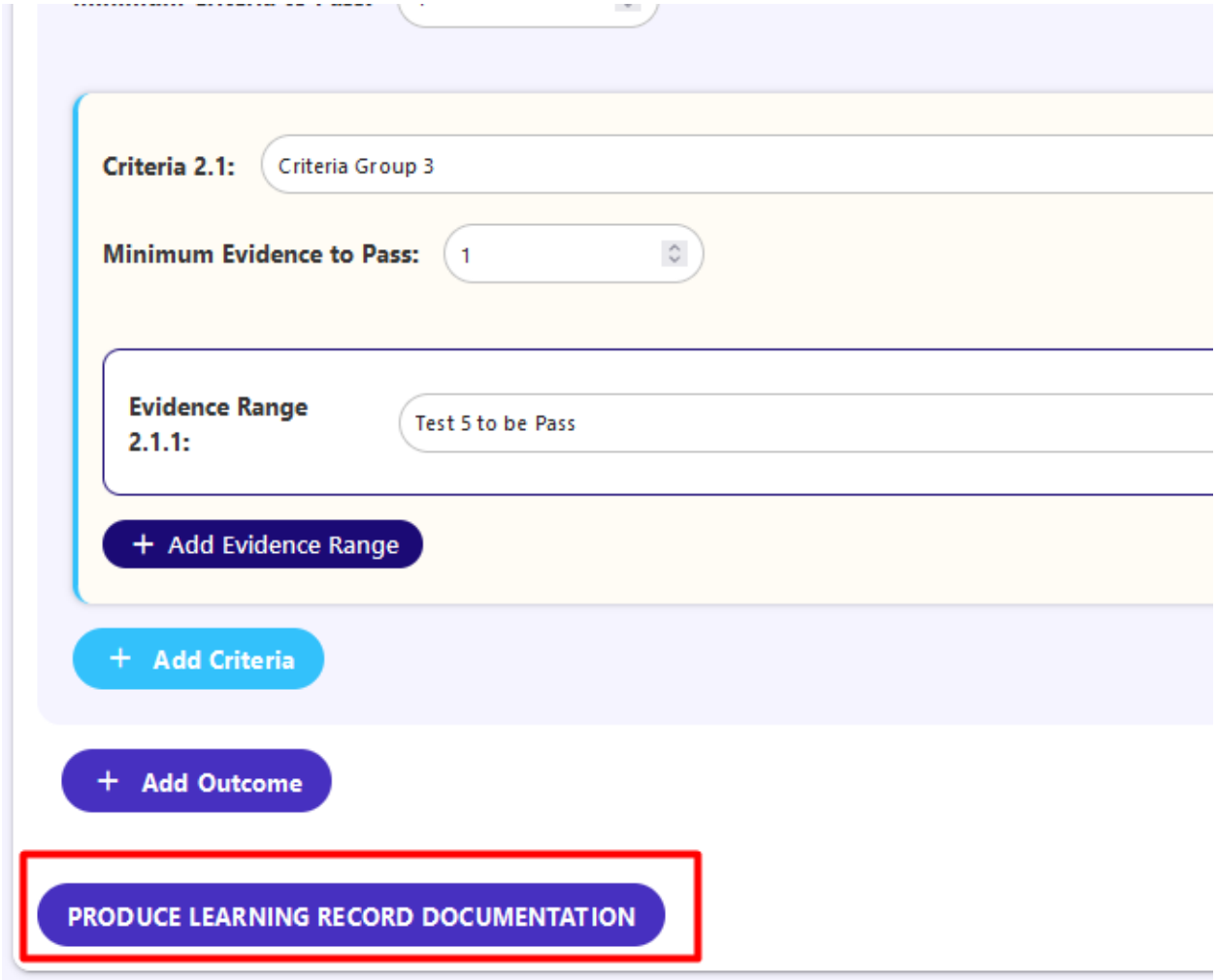
13. [Advanced Output Type Only] To add an Evidence Range, Click the “+ Add Evidence Range” button, then fill the name field.



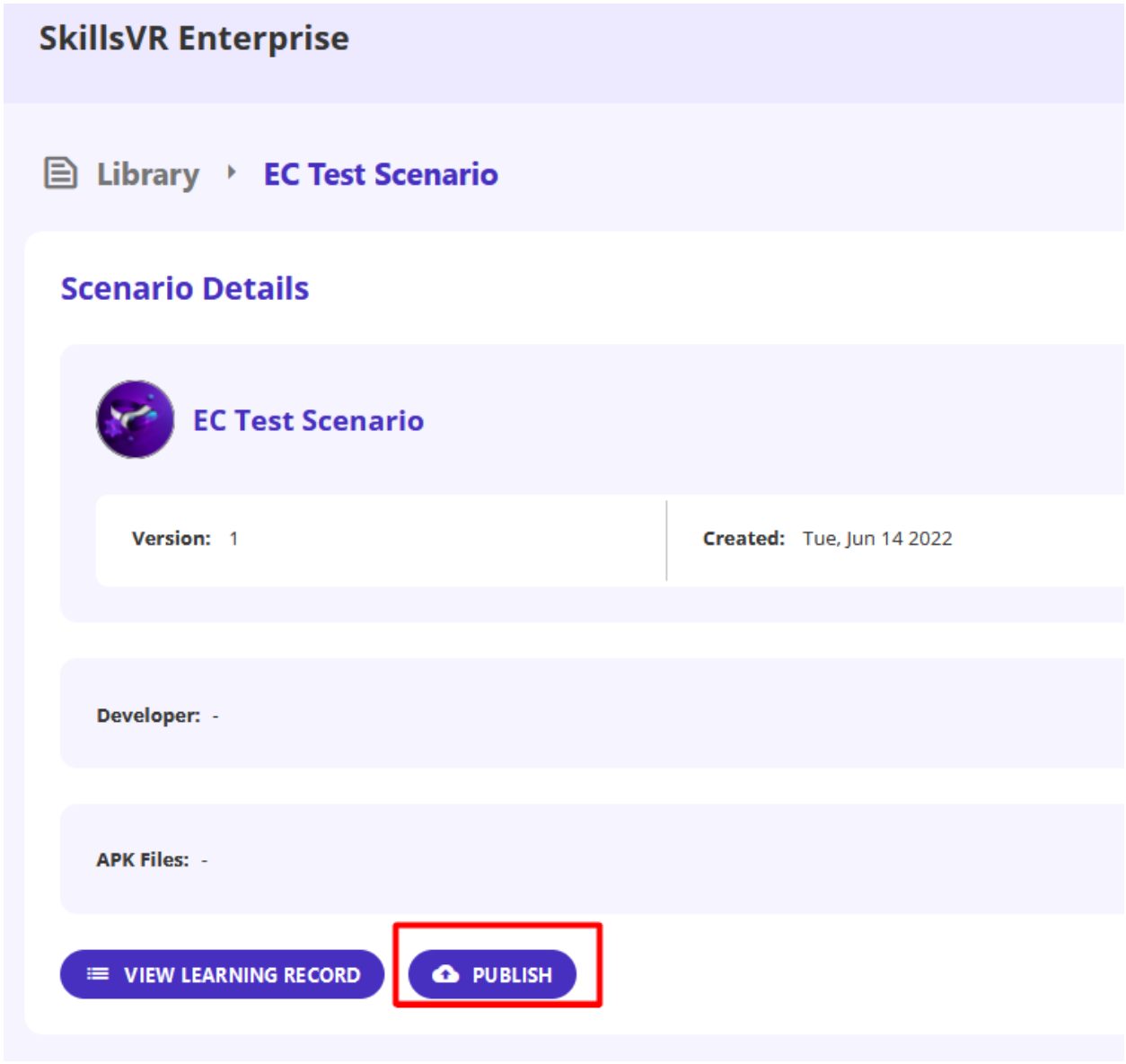


14. To remove an item created, click the “rubbish bin” or “x” button at right side;

15. Repeat step 9 to 14 to create all other outcomes and criterias.
16. To make the created scenario data can be accessed by API, click the “Produce Learning Record Documentation”, then confirm with clicking the “Produce Documentation” button. **IMPORTANT: You CAN NOT CHANGE outcomes and criterias any more after producing the record documentation.**

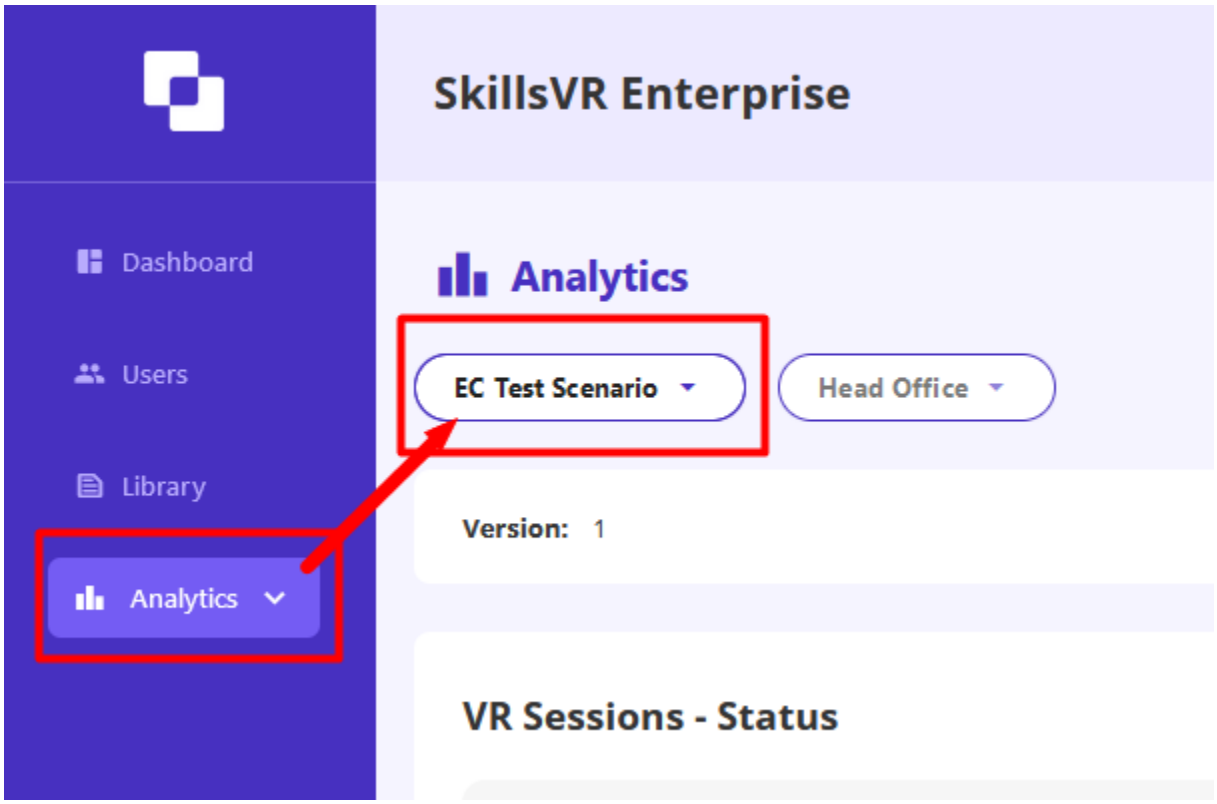


17. To publish the scenario, click the “Publish” button, then confirm.
    This step is not necessary for development but necessary for production.


SkillsVR Enterprise
Library
EC Test Scenario
Scenario Details
EC Test Scenario
Version: 1
Created: Tue, Jun 14 2022
Developer: -
APK Files: -
VIEW LEARNING RECORD
PUBLISH

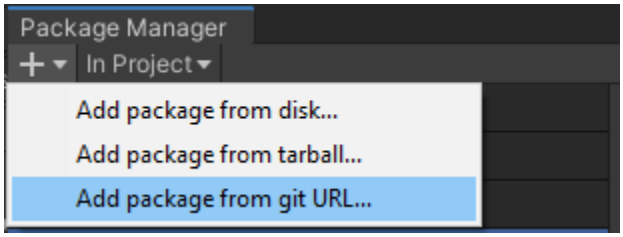

n 14 2022
Modified: Tue, Ju
Are you sure you want to publish the scenario?
PUBLISH
CANCEL

18. [Optional] After producing record documentation, Analytics is ready for this scenario. View the scenario state from “Analytics” in the left menu, then select the scenario name from the dropdown at top.
    Notice that the tutorial doesn’t have any score here until Step 3: Set Up SDK Configuration inside Unity is completed.

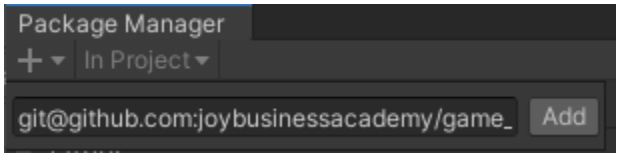


Step 2: Add SDK Package to Unity Project

1. Package Manager - Via Git URL. You will need to add the package via the Unity Package Manager System

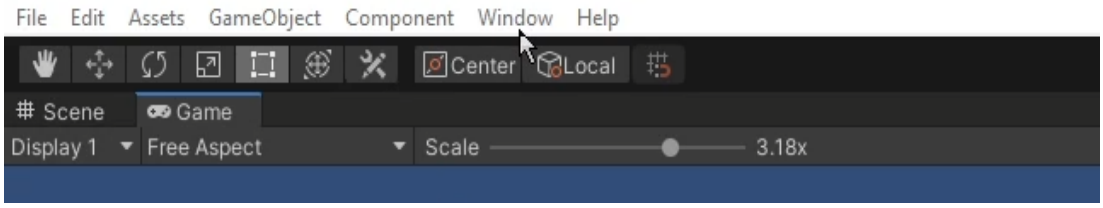


2. Git Repository URL (**git@github.com:joybusinessacademy/EC_SDK.git#1.1.0**) , after adding it will show up under packages in project

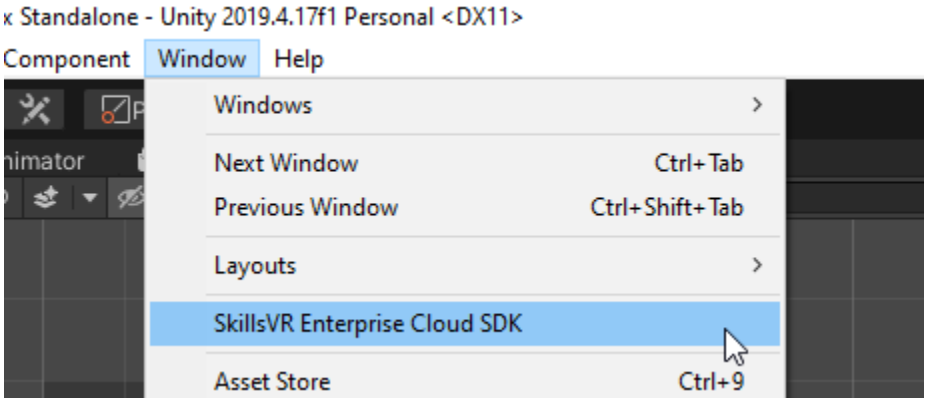


Step 3: Set Up SDK Configuration inside Unity

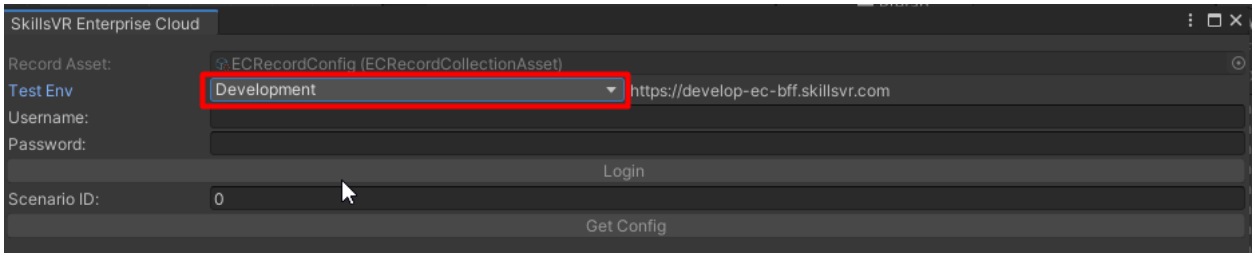
1. Once added, you will be able to see the Libraries under the packages folder in Unity.
2. A new dropdown has been added under the windows tab located at the top of Unity.



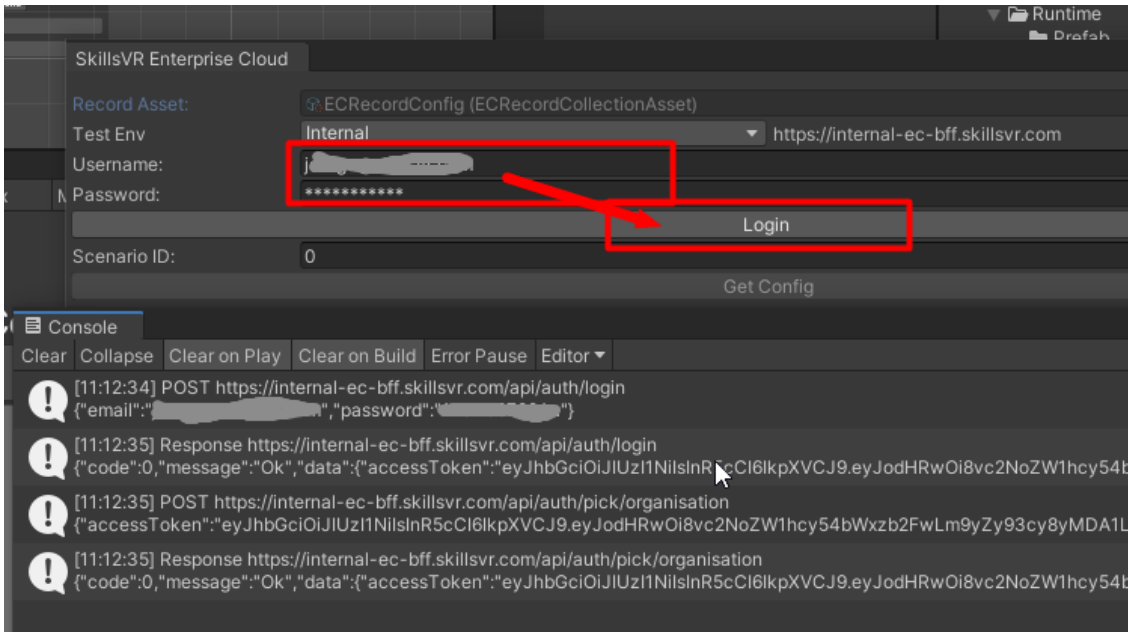
3. Drop open that tab, and a new selection called ‘SkillsVR Enterprise Cloud SDK’ will appear. Select the ‘SkillsVR Enterprise Cloud SDK’ option



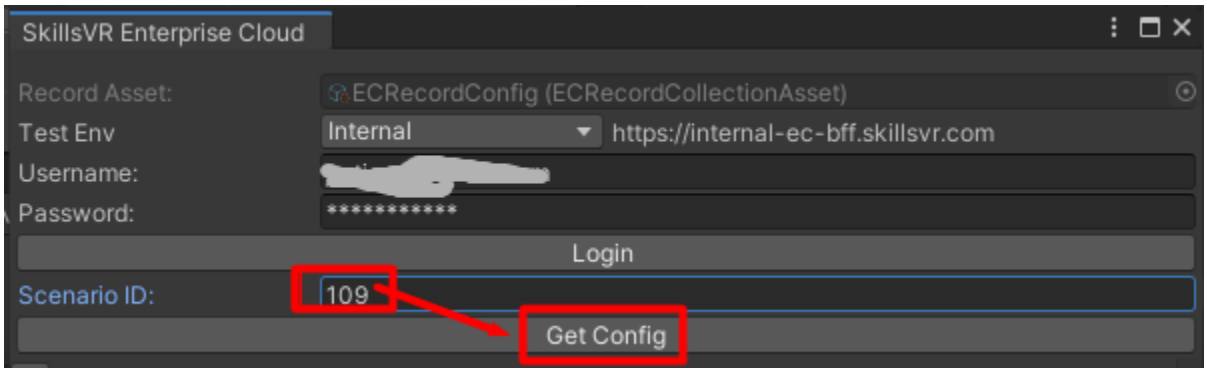
4. This will open the EC Editor Window.
5. Select the test environment from the “Test Env” dropdown.
   Environment affects the api domain and saved config assets.
   Note that all accounts/settings/configs are not shared between different environments.
   If Unity Editor is in play mode, the Test Env will be disabled and auto select the runtime environment.

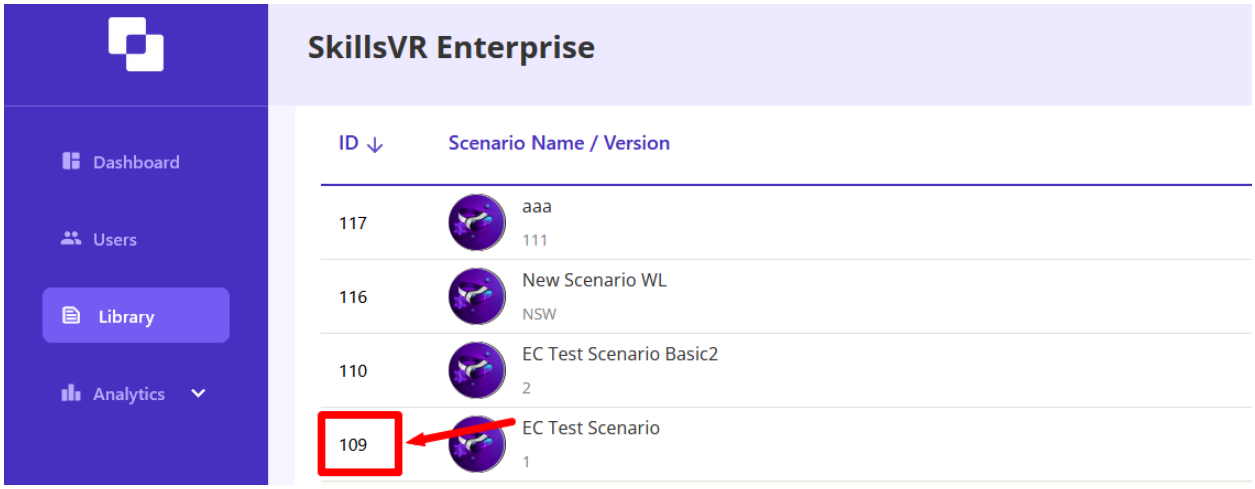


6. After setting up the test environment you will need to enter your Username and Password, then click the Login button. Login results will be posted to the log console.

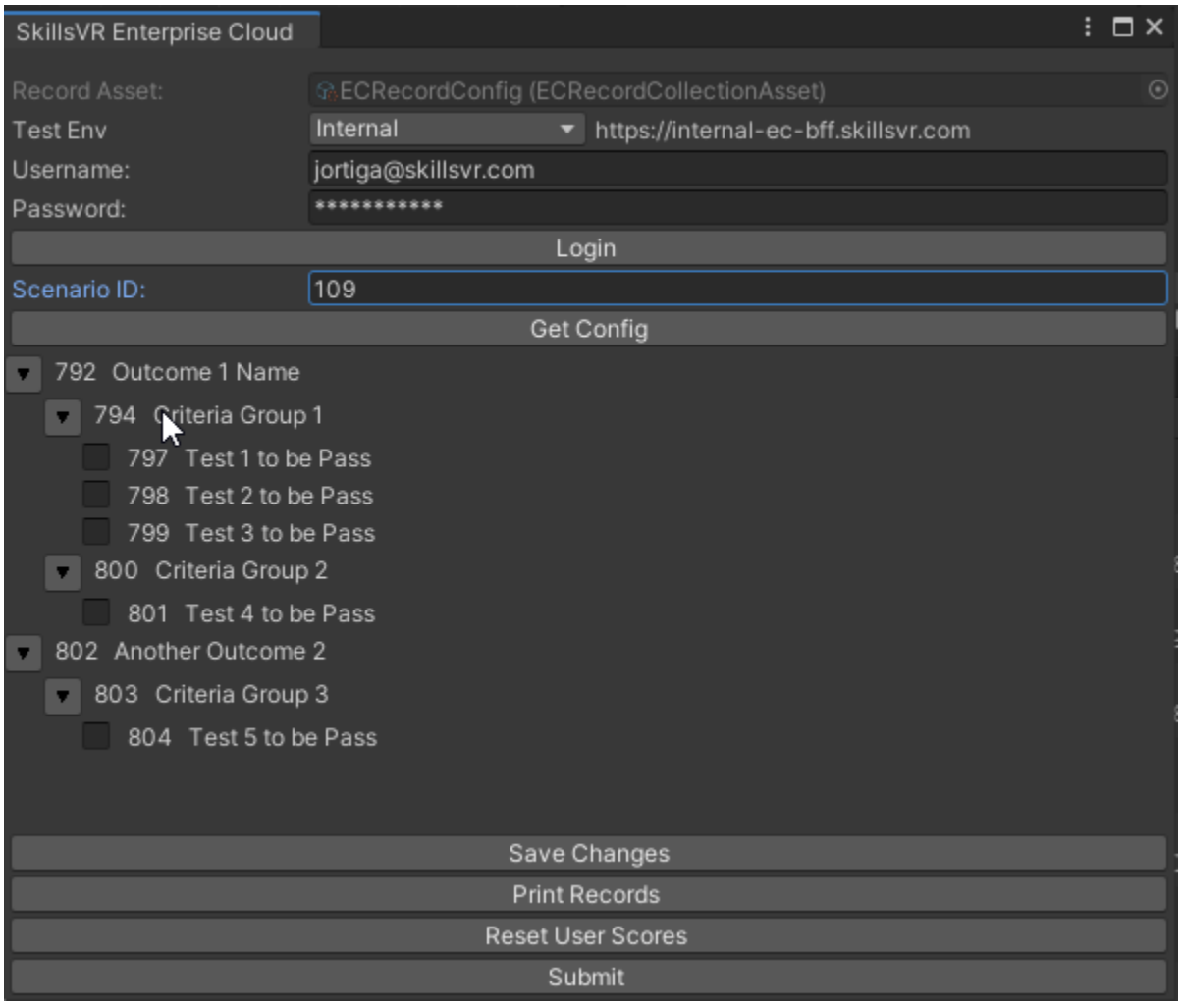


7. Enter Scenario ID you wish to see data for, then click the “Get Config” button. Scenario ID Generated from the EC Portal and can be found in the “Library” page.
Note: Config data will save to an ECRecordCollectionAsset and overwrite the old version automatically. Click the “Get Config” button will cause the current config to be lost and replaced by the new downloaded version.





8. This will pull the data to be shown, from here you can manipulate the outcomes responses for testing. And submit the data back to the enterprise cloud portal.



Next Step: Test and view scores with [Step 6: Test SDK / View Reports]

Step 4: Setup EC Environment

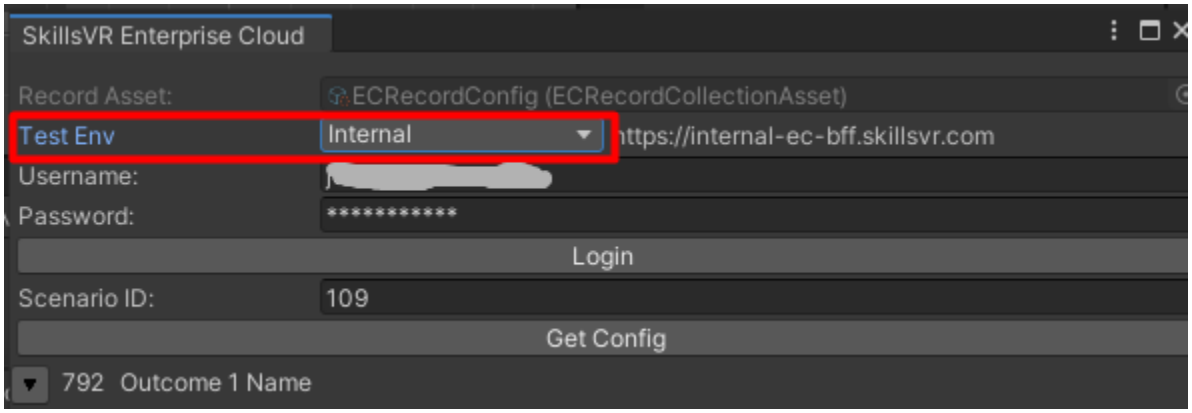
There are 4 environments predefined for EC:

- Development,
- Internal,
- Staging,
- And Production.

Set Editor Environment

In Unity Editor mode, EC Environment could be easily changed by the “Test Env” dropdown in SDK window.

Note: This change is for editor test use only and not saved for runtime.



Set Runtime Environment

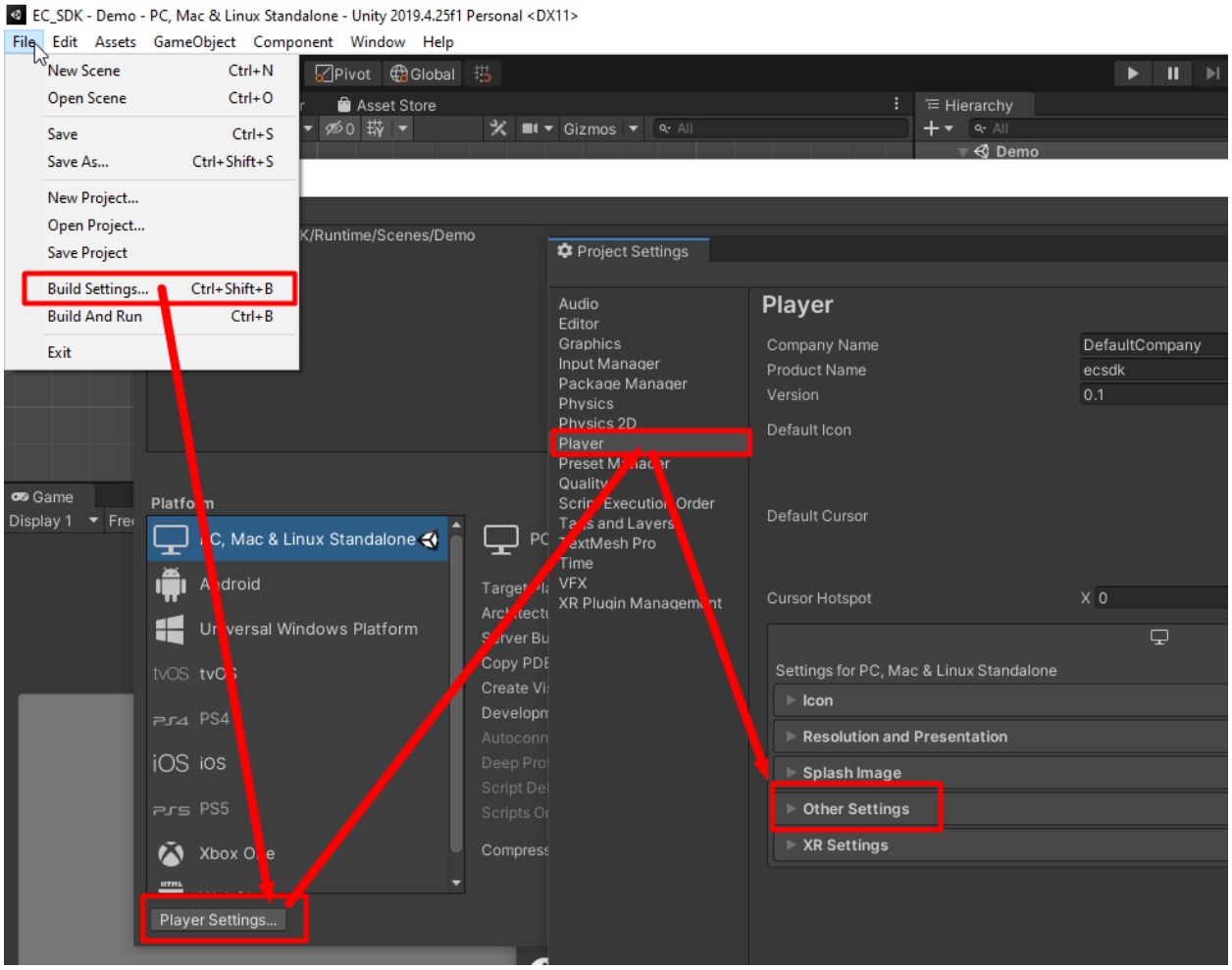
Runtime environment is set up by project predefined symbols and automatically applied when the game starts (or play mode start).

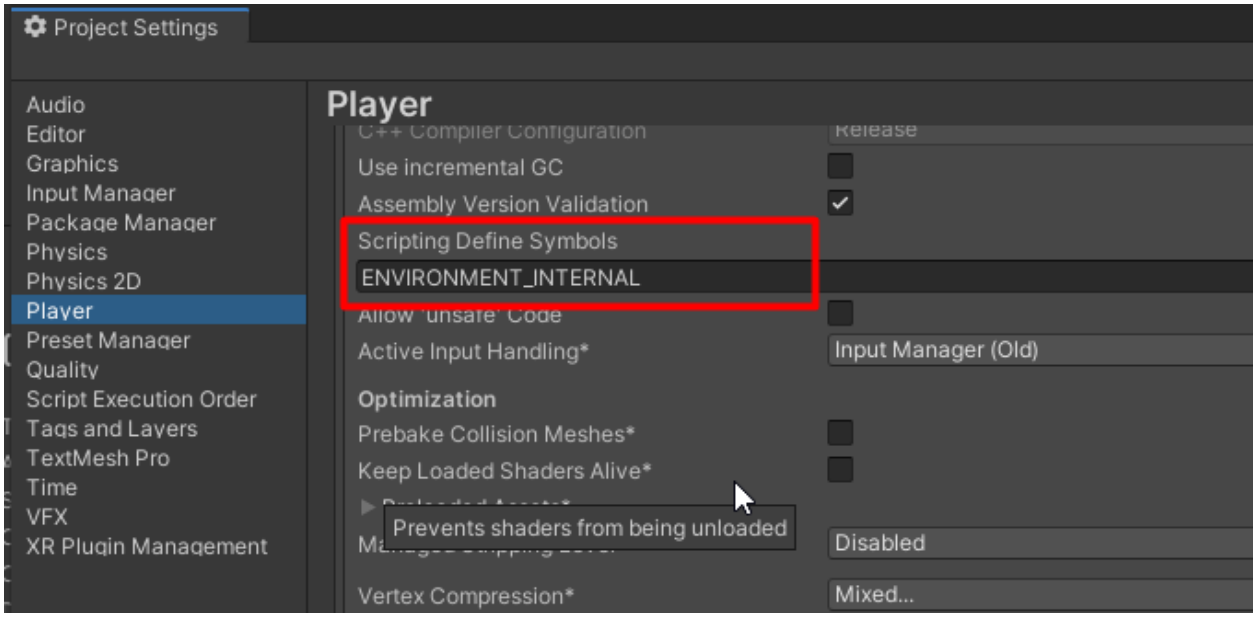
To change the runtime environment, one of the four scripting defined symbols must be added into project settings.

- Development: ENVIRONMENT_DEVELOPMENT
- Internal: ENVIRONMENT_INTERNAL
- Staging: ENVIRONMENT_STAGING
- Production: ENVIRONMENT_PRODUCTION

If no symbol is defined, Internal will be auto set as default environment.

The follow steps show how to add “ENVIRONMENT_INTERNAL” into project settings:





Repeat those steps for all platforms supported by the project.

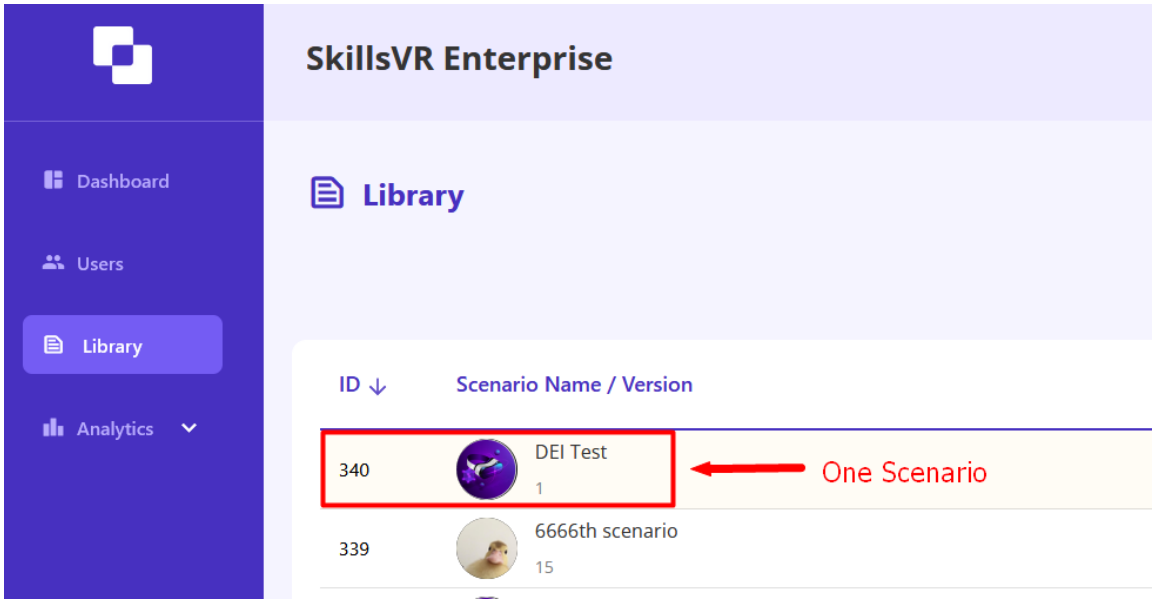
Concept Introduction

Scenario

Scenario is a game project level concept reference to a type of vr game flow, like Site Safe or Kiwi Fruit. Most skills vr games only have one scenario for one unity project, and some of them may have multiple scenarios in one project for different variants for different clients, like Conflict: AirNZ and Conflict: ShopCare.

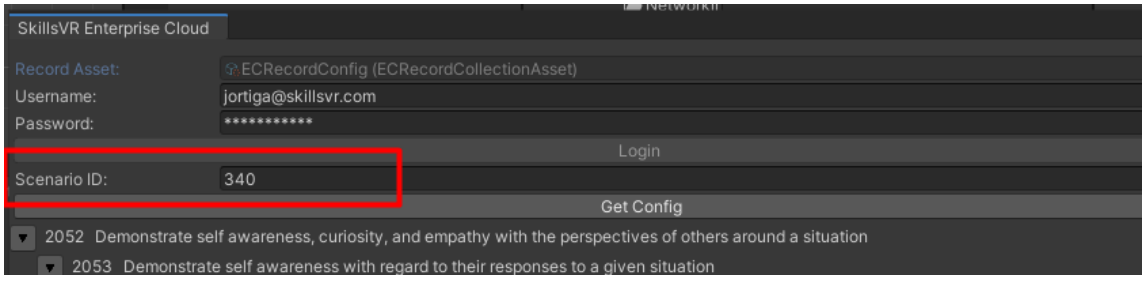
Scenarios are identified by scenario ids.

Scenario in EC Portal



In the image above, “DEI Test” is a scenario and “340” is scenario Id.

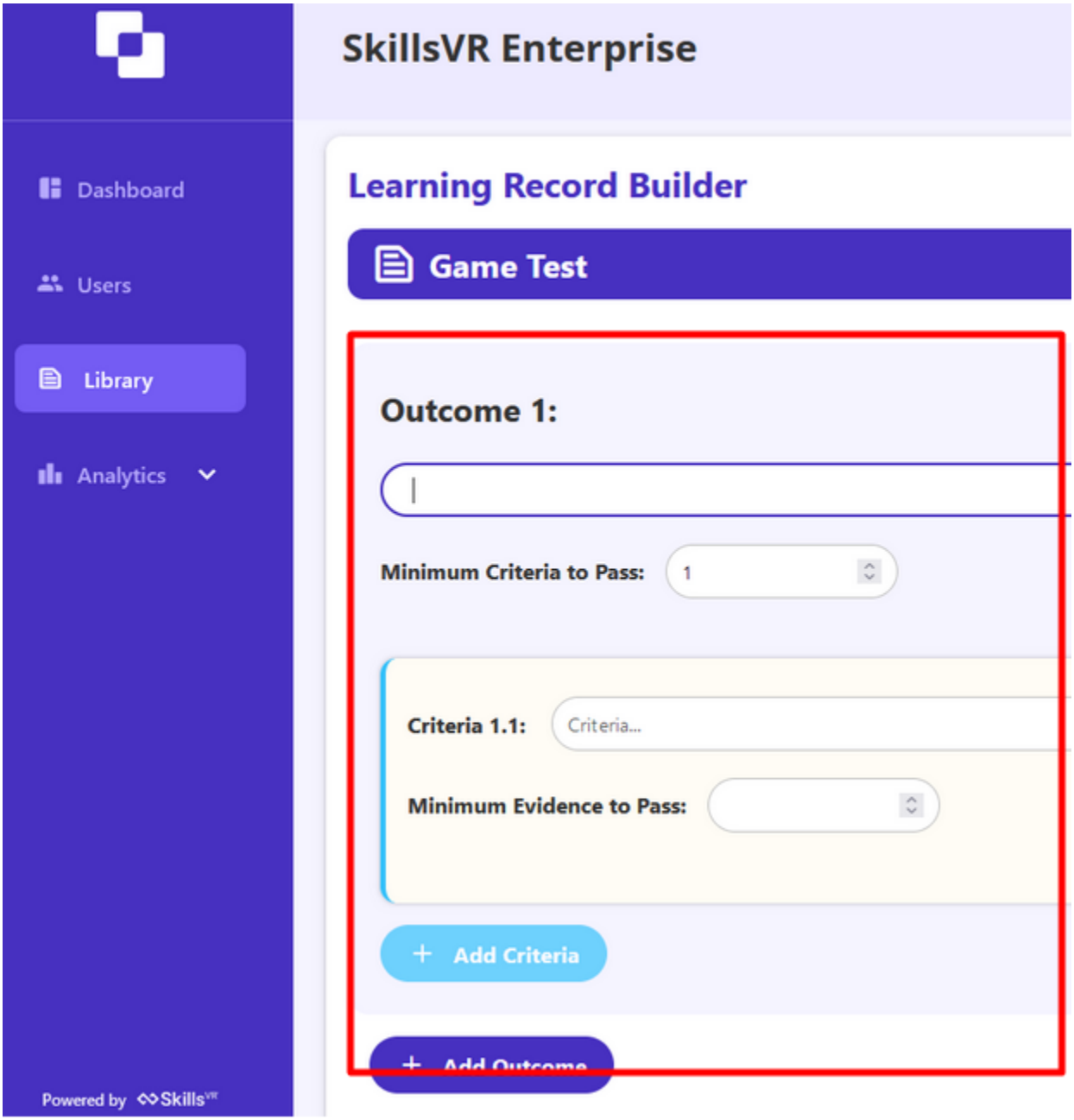
Scenario in Unity Editor



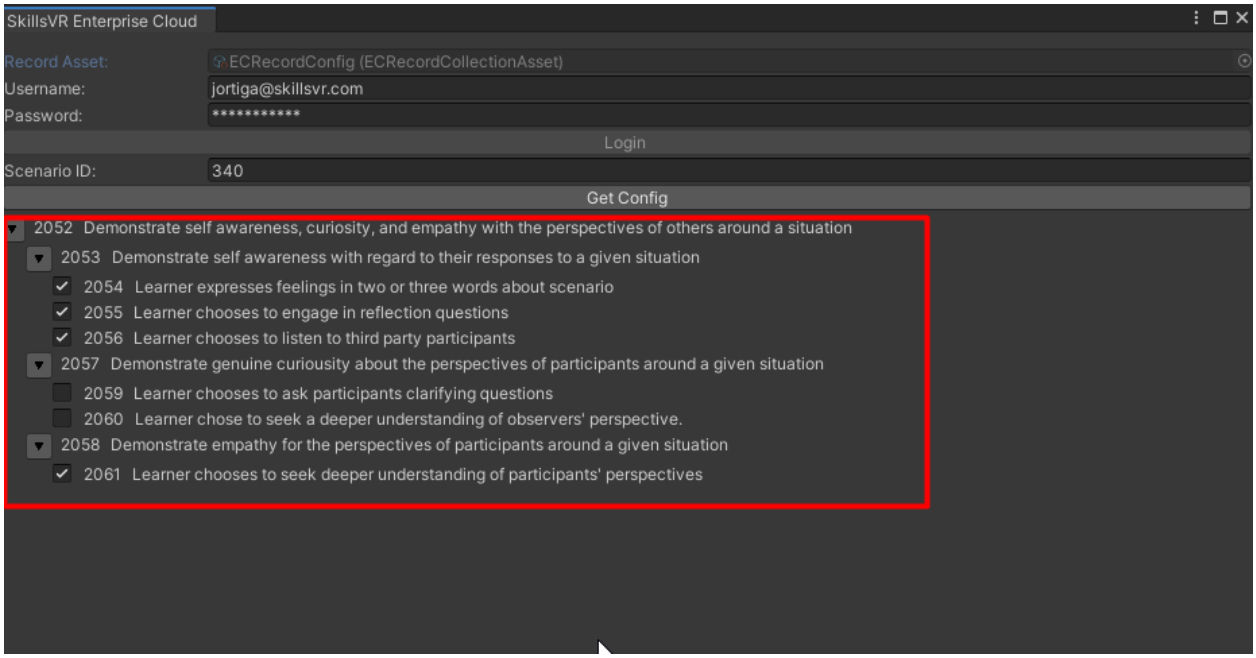
Scenario Config

Scenario config is a collection of EC records. One scenario has one config.

Scenario Config in EC Portal



Scenario Config in Unity Editor

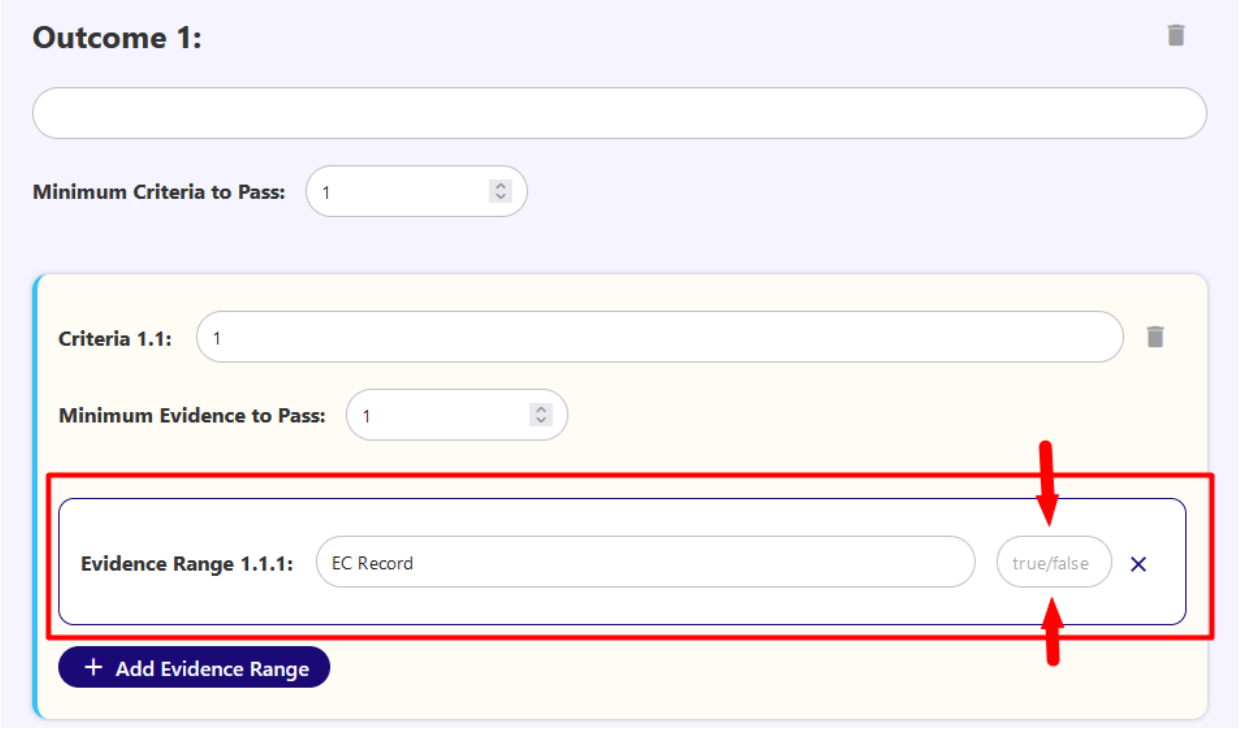


EC Record

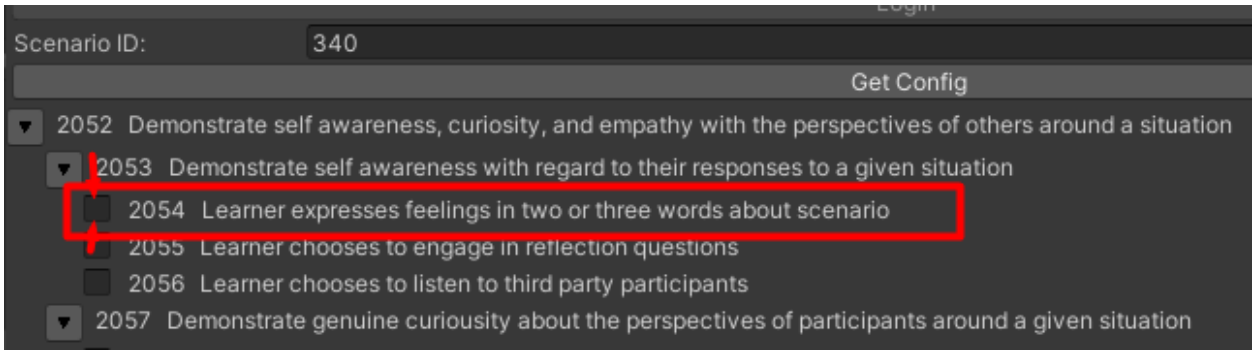
EC Record is one criteria or evidence range, has id, name, game score type and other properties.

EC Record in EC Portal

EC Record in the portal is the criteria (Basic Output) or the evidence range (Advanced Output) which has a **true/false** mark. Note that the record Id (i.e. 2064) is not shown in the portal.



EC Record in Unity Editor

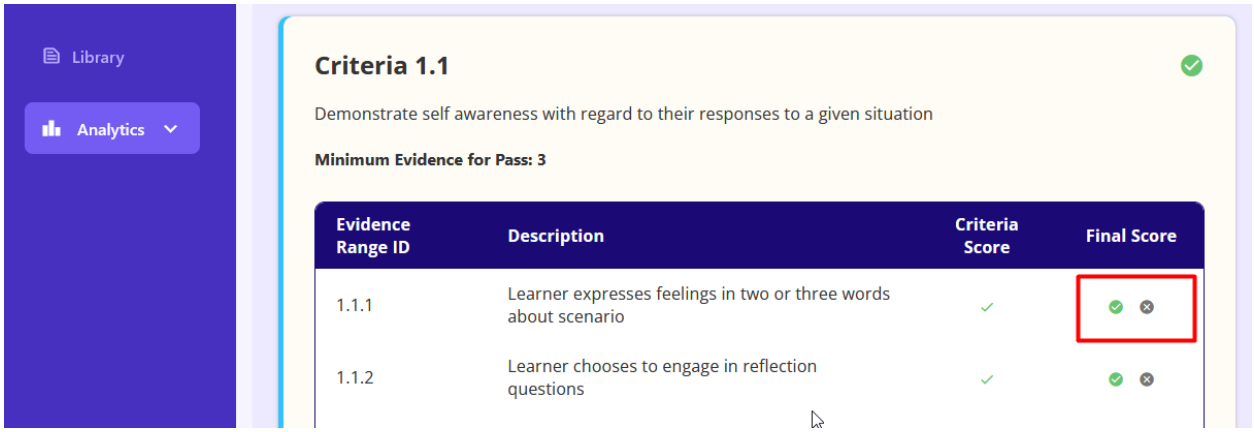


In editor an EC record is displayed in format of “Toggle, Id, Name”, as in the image above.

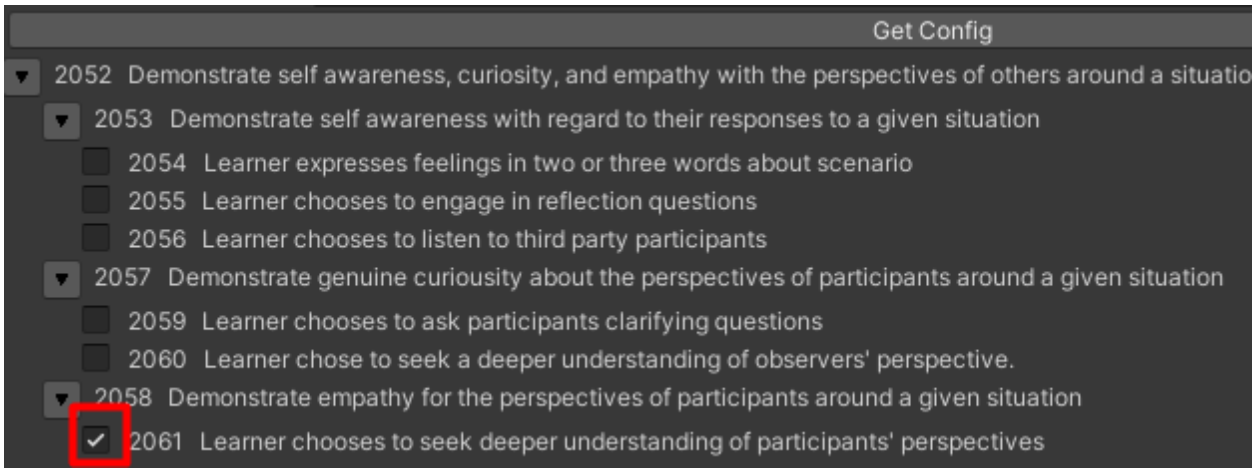
Game Score

Game score is a boolean value within an EC record, which is used to check if a player passes the criteria of this record.
In v1.0.0, game scores only have one type of data which is boolean. May have more types in feature.

Game Score in EC Portal



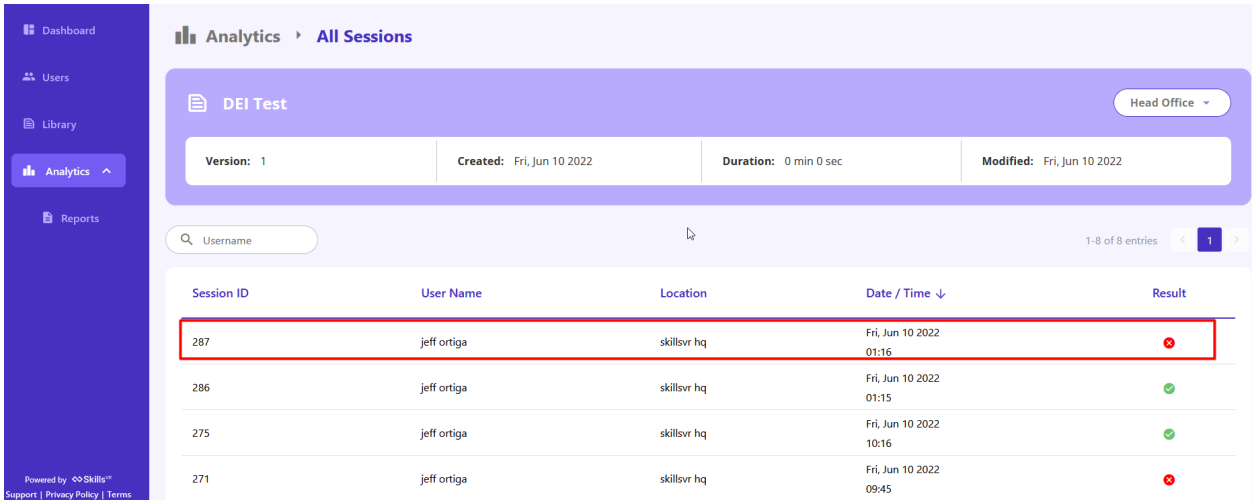
Game Score in Unity Editor



Session

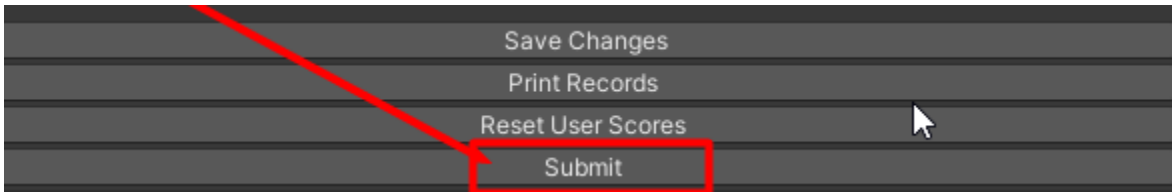
A session is a general name of an instance (or gameplay) of any kind of assessment or training. Every time a player starts a new assessment or training, it starts a new session for the backend. For EC, sessions are automatically created by submitting game scores.

Session in EC Portal

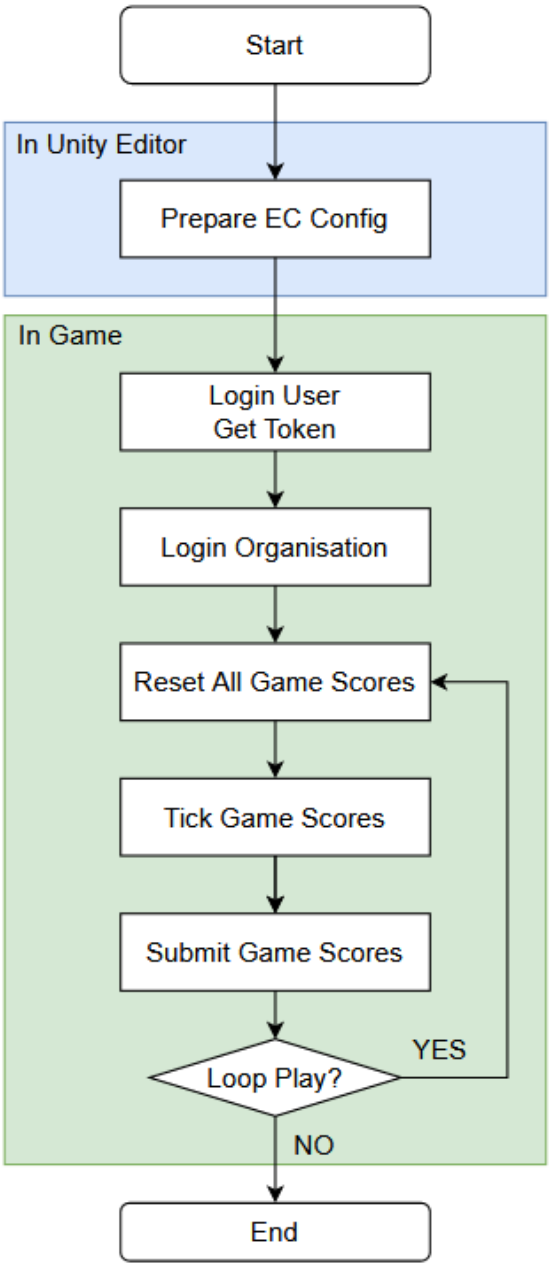


Session in Unity Editor

There is no session view in editor or in game. Sessions are created when clicking the “Submit” button (in editor), or submit score by ECAPIs.



EC Record Workflow



EC Record Workflow

Step 5: How to use the EC SDK in Unity (Code and No Code)

Based on EC Record workflow, there are few main actions to use EC records:

- Login;
- Login Organisation;

- Get Config;
- Reset All Game Scores;
- Set Game Scores;
- Submit Game Scores;

In Unity, there are two ways to run each action:

- Call APIs from Code,
- And Build Unity Events from Inspector.

Call APIs from Code

In EC SDK there is a class called ECAPI, all kinds of action methods can be found here and directly called with static methods.

Class: ECAPI
Name Space: SkillsVR.EnterpriseCloudSDK

Static Methods

Login	Login user to EC backend and grab access token.
void Login(string @email, string @password, System.Action<Login.Response> success = null, System.Action<string> failed = null);	
Parameters	
email	user account
password	user password
success	Action runs when login success. Params: Login.Response - response data for login request.
failed	Action runs when login fails, including http and network errors. Params: string - the error message.

Login Organisation	Login organisation with access token.

void LoginOrganisation(int organisationId, string userRoleName, string userProjectName, System.Action<Login.Response> success = null, System.Action<string> failed = null);	
Parameters	
organisationId	User organisation id.Can get from LoginResponse.data.organisations.id.
userRoleName	User role name. Can get from LoginResponse.data.organisations.roles.key.
userProjectName	User project name. Can get from LoginResponse.data.organisations.name.
success	Action runs when login success. Params: Login.Response - response data for login request.
failed	Action runs when login fails, including http and network errors. Params: string - the error message.

SetUserGameScoreBool	Set bool type game score to a record by record id.
bool SetUserGameScoreBool(int recordId, bool isOn, System.Action<string> failed = null);	
Parameters	
recordId	Id that matches EC record id.
isOn	The boolean game score value.
failed	Action runs when fails. Params: string - the error message.
Return	Success of setting a user game score.

GetUserGameScoreBool	Get bool type game score from a record by record id.
bool GetUserGameScoreBool(int recordId);	

Parameters	
recordId	Id that matches EC record id.
Return	Boolean type value of game score.

ResetAllUserScores	Reset all user scores to init stats. Any user changes will be lost.
void ResetAllUserScores();	

SubmitUserLearningRecord	Submit all user scores to the EC backend. Note: for v1.0.0 only send records that type is 0 (bool type game score).
void SubmitUserLearningRecord(System.Action<AbstractAPI.EmptyResponse> success = null, System.Action<string> failed = null);	
Parameters	
success	Action runs when submit success. Params: AbstractAPI.EmptyResponse - not in use, empty data.
failed	Action runs when submit fails, including http and network errors. Params: string - the error message.
void SubmitUserLearningRecord(int xScenarioId, IEnumerable<ECRecordContent> recordCollection, System.Action<AbstractAPI.EmptyResponse> success = null, System.Action<string> failed = null);	
xScenarioId	The id of the scenario to be sent.
recordCollection	List of records to be sent. Note: for v1.0.0 only send records that type is 0 (bool type game score).
success	Action runs when submit success.

	Params: AbstractAPI.EmptyResponse - not in use, empty data.
failed	Action runs when submit fails, including http and network errors. Params: string - the error message.

GetConfig	Download scenario record config by id.
void GetConfig(int scenarioId, System.Action<GetConfig.Response> success = null, System.Action<string> failed = null)	
Parameters	
scenarioId	Scenario config id
success	Action runs when submit success. Params: GetConfig.Response - config data including a list of records.
failed	Action runs when login fails, including http and network errors. Params: string - the error message.

See more detailed definitions in [PackageRootDir]\Runtime\Scripts\ECAPI.cs
Note: Some APIs may fail without login, make sure to do Login() first.

Example Code

```
// == Login User == //
ECAPI.Login(user, passwork,  (loginResp) => {

        var organisation = loginResp.data.organisations[0];
        int organisationId = int.Parse(organisation.id);
        string projectName = organisation.name;
        var userRole = organisation.roles[0].key;

        // == Login Organisation == //
        ECAPI.LoginOrganisation(organisationId, userRole, projectName, (loginOrgResp) =>{

                // == Download Config == //
                ECAPI.GetConfig(99, (configResp) => {

                        // == Set Config to Reference Asset == //
                        ECRecordCollectionAsset.GetECRecordAsset()
                                .AddRange(configResp.data);

                        // == Reset All Game Scores == //
                        ECAPI.ResetAllUserScores();

                        // == Set Game Score by Record Id == //
                        bool success = ECAPI.SetUserGameScoreBool
                                        (recordId, true, Debug.LogError);

                        // == Get Game Score by Record Id == //
                        bool gameScore = ECAPI.GetUserGameScoreBool(recordId);

                        // == Submit Records to Backend == //
                        ECAPI.SubmitUserLearningRecord(
                                (submitResp) => {}, Debug.LogError);
                }, Debug.LogError);
        }, Debug.LogError);
}, Debug.LogError);
```

Build Unity Events from Inspector

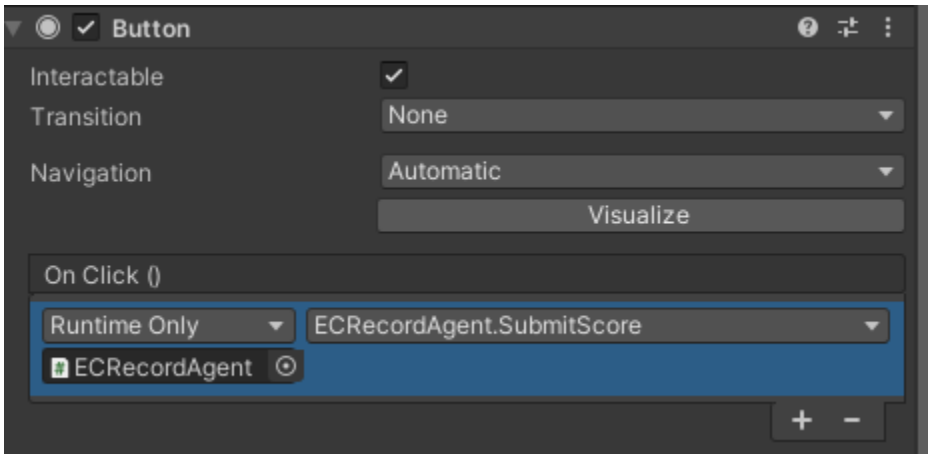
EC SDK also provides a non-code way to use APIs.

There is a built-in unity component named ECRecordAgent which is a ready to use EC API event builder.

EC Record Agent Introduction

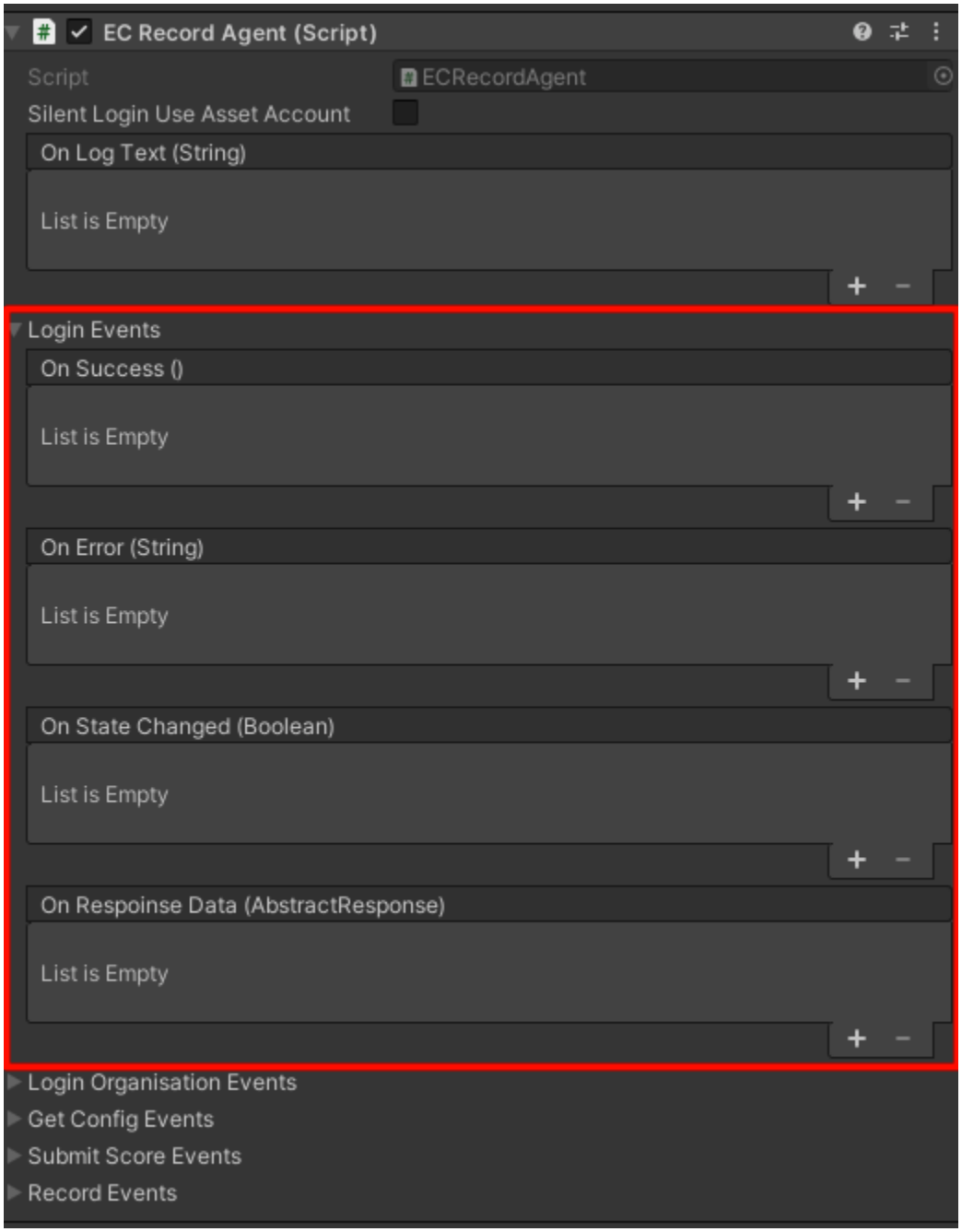
ECRecordAgent contains two parts:

- API Inspector Callers: Invoke ECAPIs from Unity Events in inspector.
  The following image shows an example of submitting scores from a button click.



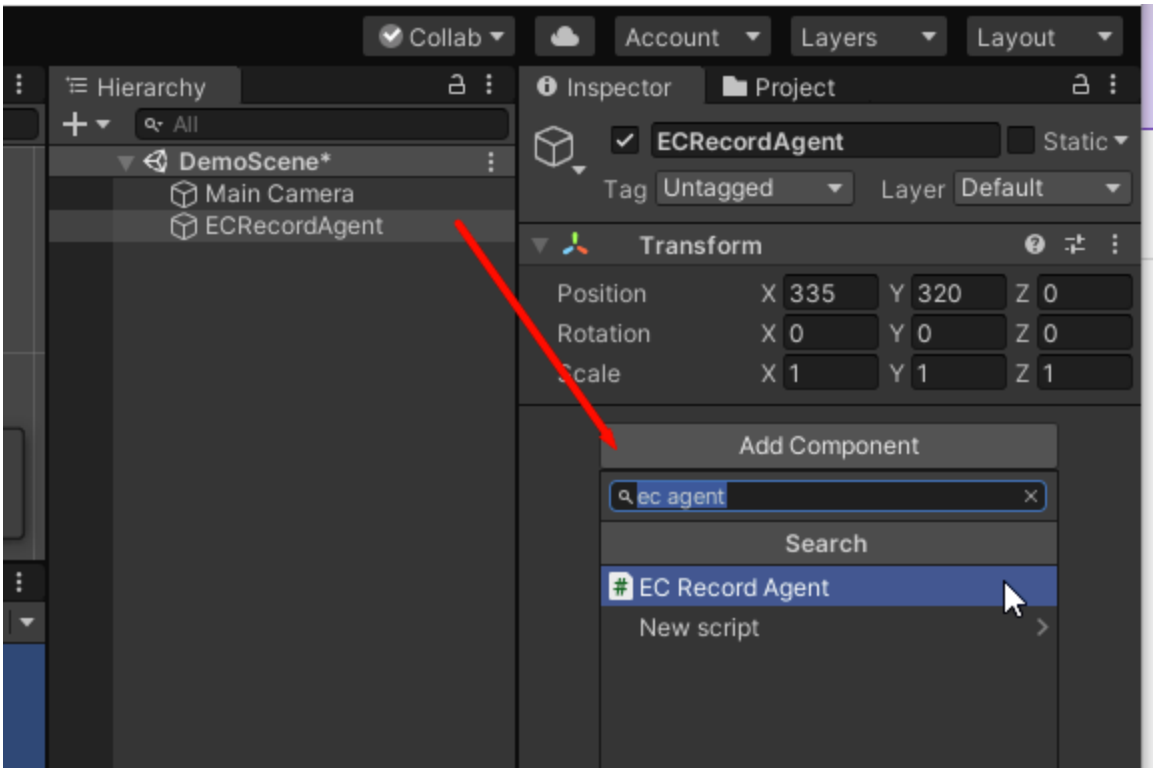
- EC Event Receivers: Receive ECAPI action (i.e. login or submit) callbacks and provide Unity Event interface to inspector.

The red area in the following image shows an example of EC login events.

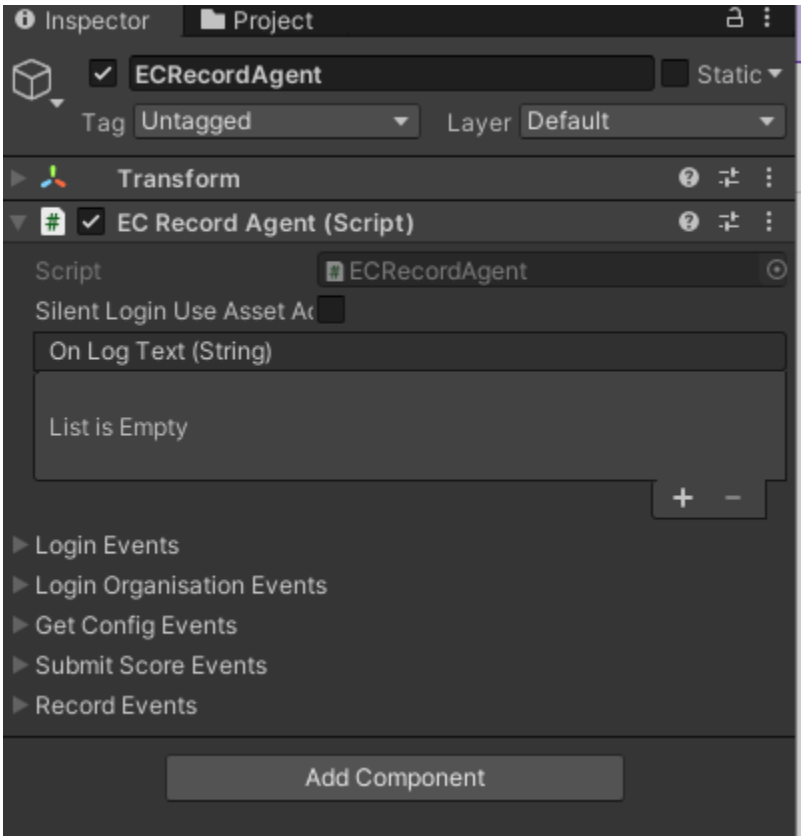


1. Setup Agent Component

1. Select or create a game object in Scene Hierarchy view;
2. In inspector, click the “Add Component” button;
3. Search “EC Agent” then click the “EC Record Agent”;



Now the attached EC record agent component is ready to use, as the following image shows.

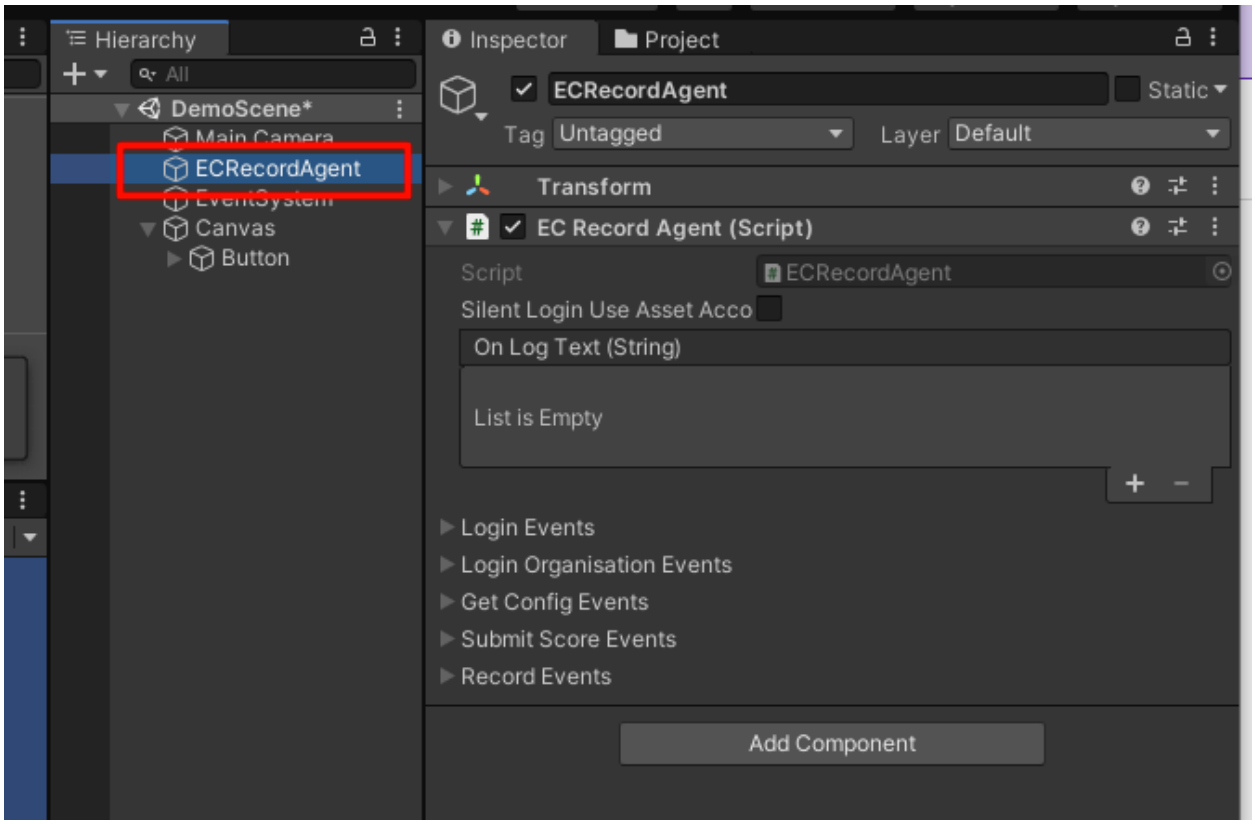


Note: EC Record Agent could have more than one instance in a scene, however, all agents share and operate one record asset source. This means if a record is set true by agent A then reset by agent B, all agents will get a false value after reset record.

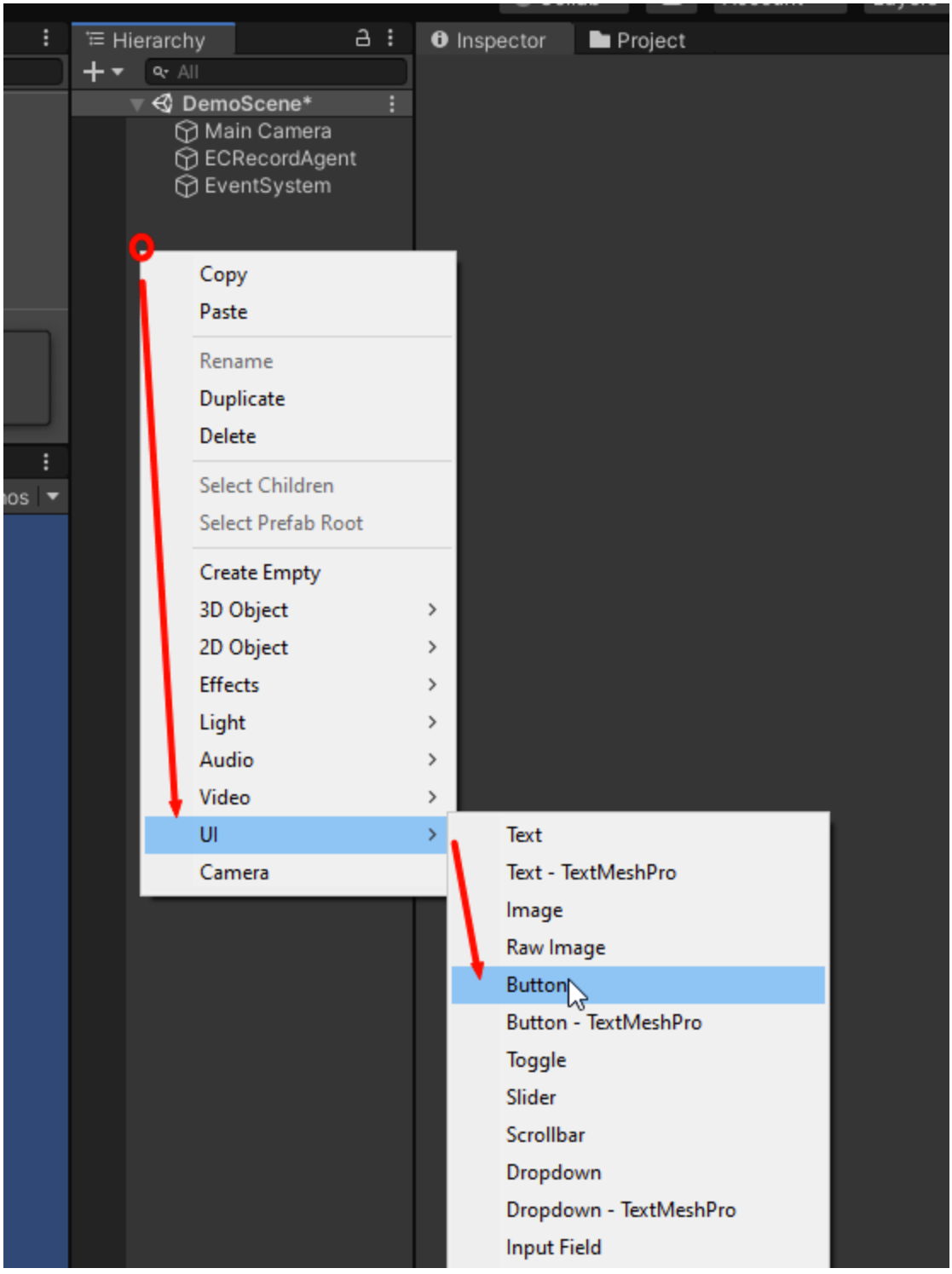
2.Invoke ECAPIs from Inspector

The following example steps show how to invoke the EC Login method from a default GUI button OnClick event. With the same setup steps, ECAPIs can be invoked from any type of Unity Event.

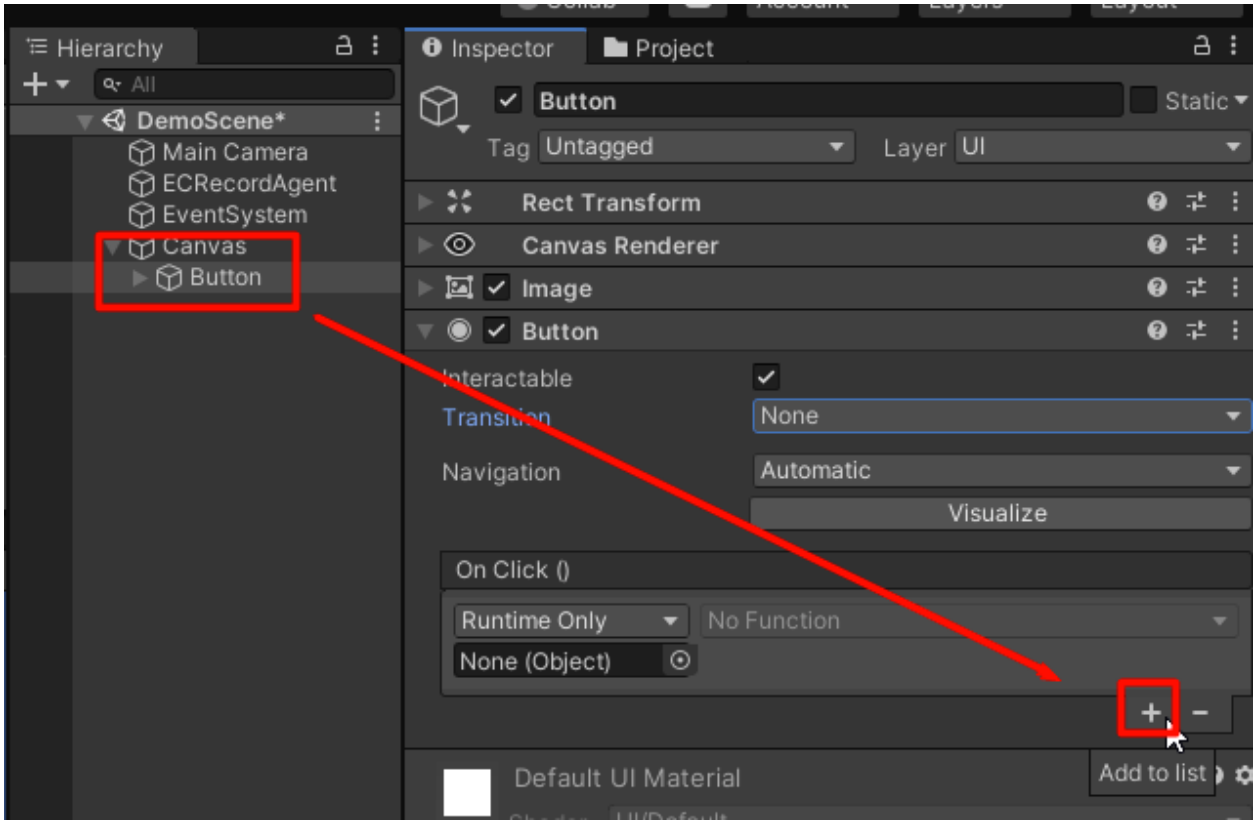
1. Follow Setup Agent Component to create at least 1 ECRecordAgent in scene;



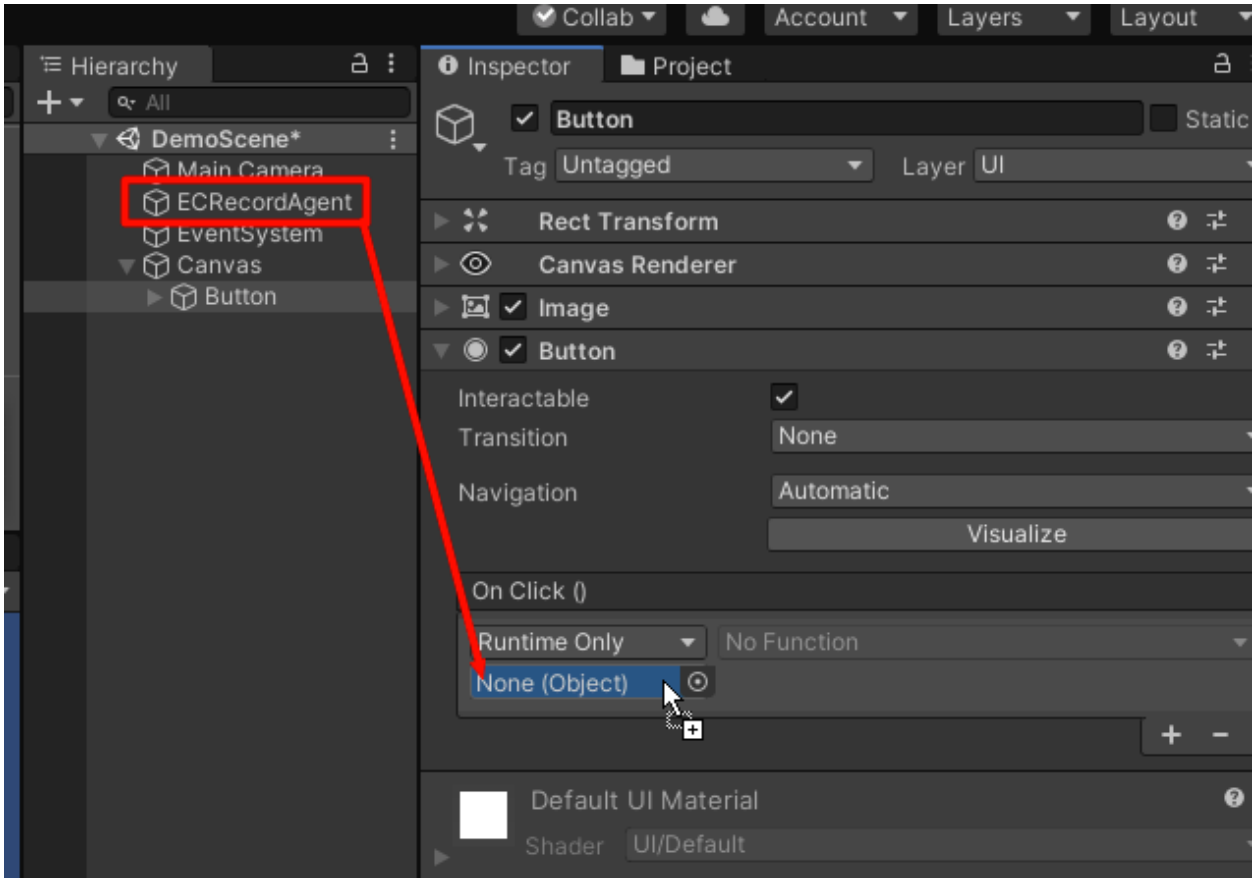
2. Create or select a game object that has unity events in the inspector, in this example, right click in Hierarchy View, then click the “UI -> Button”;



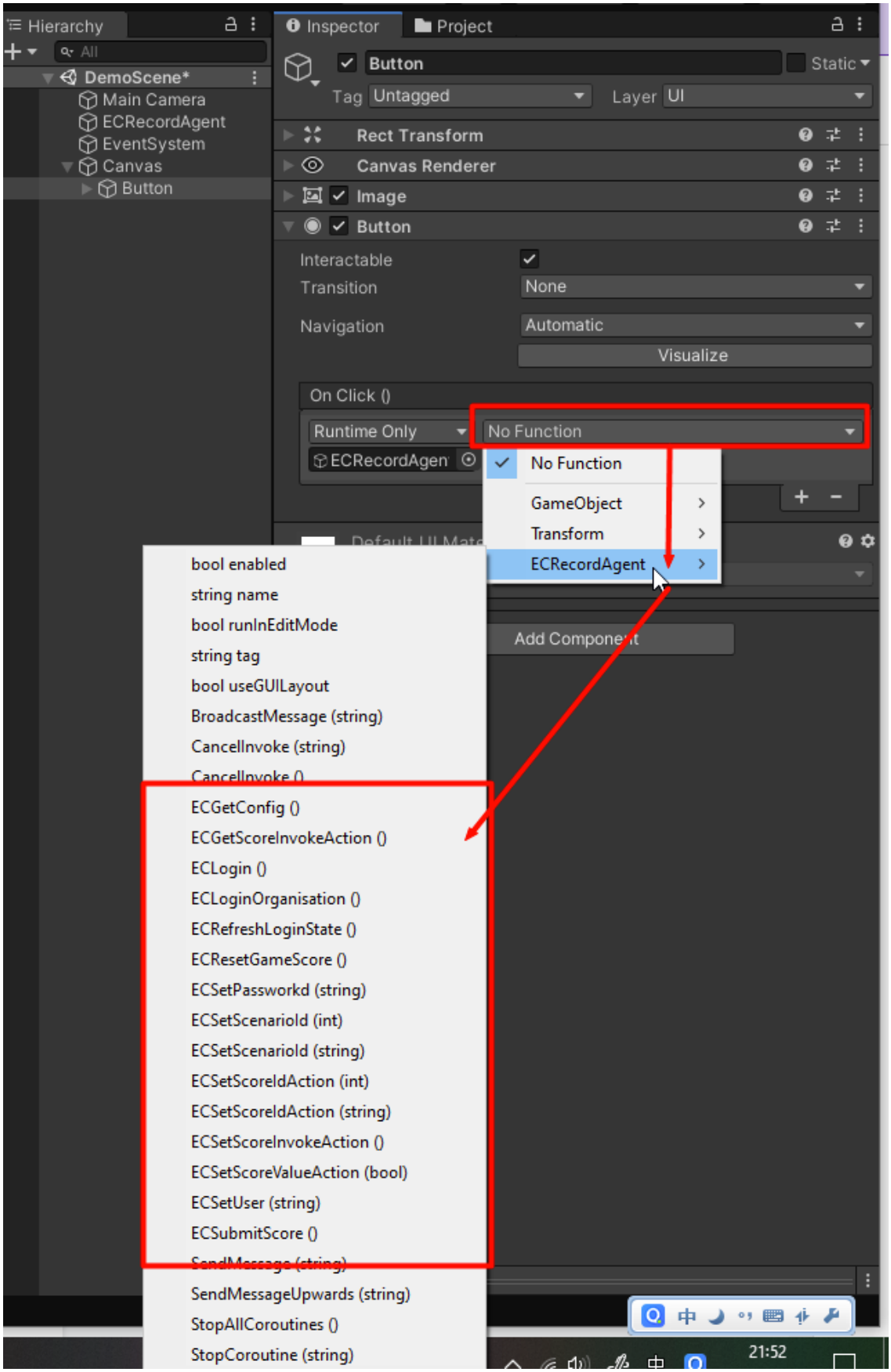
3. Select the Button game object, then click the “+” button under On Click () event;



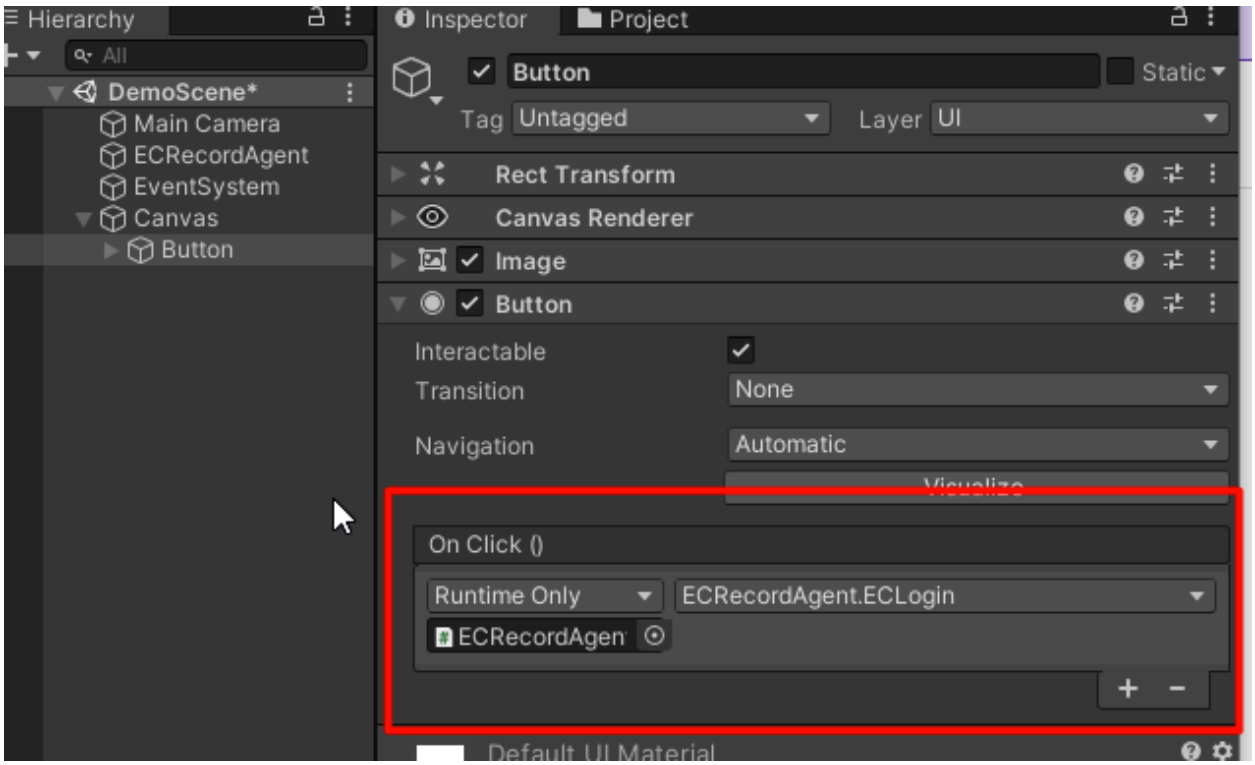
4. Drag the ECRecordAgent game object created by step 1, then drop off to the “None (Object)” field;



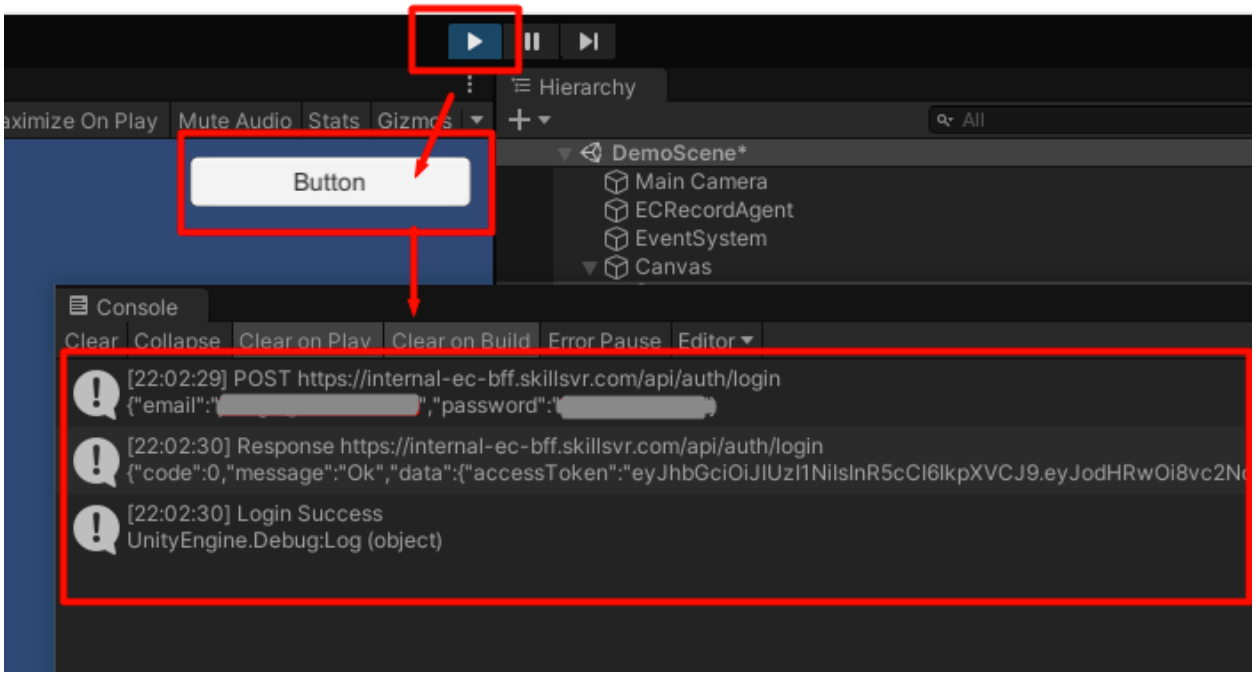
5. Click the “No Function” dropdown, then select “ECRecordAgent”;


Hierarchy
Inspector
Project
DemoScene*
Main Camera
ECRecordAgent
EventSystem
Canvas
Button
Button
Static
Tag Untagged
Layer UI
Rect Transform
Canvas Renderer
Image
Button
Interactable
Transition None
Navigation Automatic
Visualize
On Click ()
Runtime Only
No Function
ECRecordAgen
No Function
GameObject
Transform
ECRecordAgent
Add Component
bool enabled
string name
bool runInEditMode
string tag
bool useGUILayout
BroadcastMessage (string)
CancelInvoke (string)
CancelInvoke ()
ECGetConfig ()
ECGetScoreInvokeAction ()
ECLogin ()
ECLoginOrganisation ()
ECRefreshLoginState ()
ECResetGameScore ()
ECSetPasswordd (string)
ECSetScenariold (int)
ECSetScenariold (string)
ECSetScoreldAction (int)
ECSetScoreldAction (string)
ECSetScoreInvokeAction ()
ECSetScoreValueAction (bool)
ECSetUser (string)
ECSubmitScore ()
SendMessage (string)
SendMessageUpwards (string)
StopAllCoroutines ()
StopCoroutine (string)
21:52

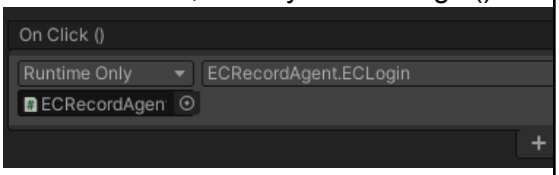
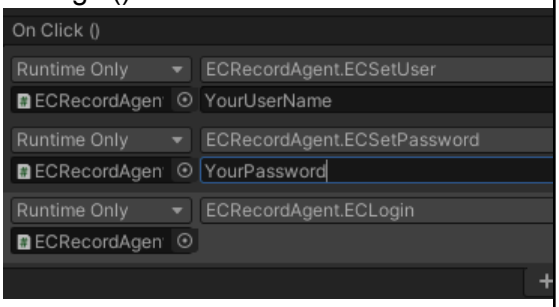
6. All EC related methods are named starting with “EC”, now select the “ECLogin()”. The final inspector is shown in the following image. For more information, please refer to [EC Record Agent Method Reference](#).

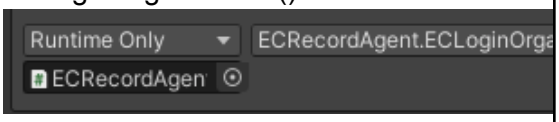
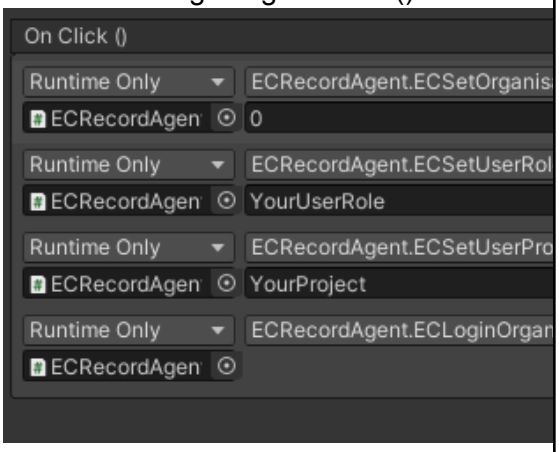


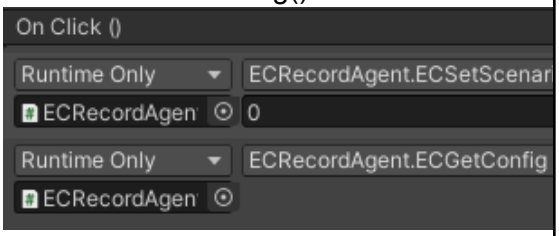
To test the login button, play game in editor then click the button. Login results will print to log console.

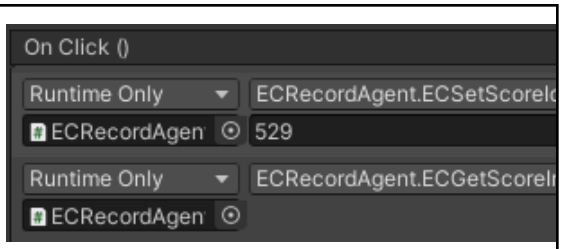
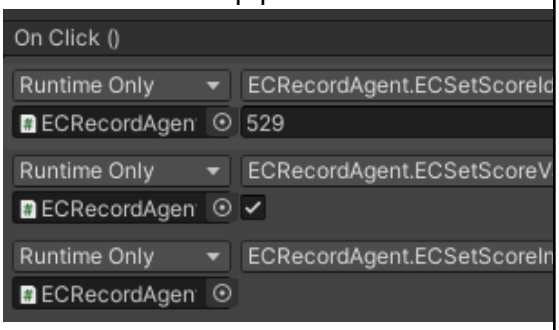


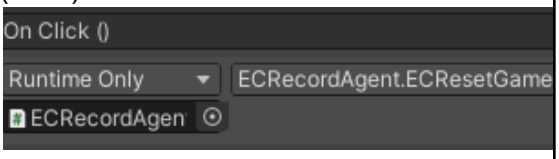
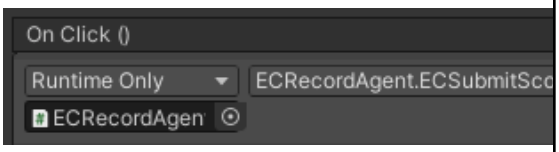
EC Record Agent Method Reference

Group	Method	Description
Login	void ECLogin()	Login user to EC backend. To login with default user account saved in record asset, directly call ECLogin().  To Login with custom user name and password, Call ECSetUser(string) and ECSetPassword(string) first, then call ECLogin().  Login result output by loginEvents.
	void ECSetUser(string userName)	Set a custom user name for ECLogin(). If nothing set, the user name saved in record asset will be used as default. Params: string - the custom user name.
	void ECSetPassword(string userPassword)	Set a custom password for ECLogin(). If nothing set, the password saved in record asset will be used as default. Params: string - the custom password.
	void ECRefreshLoginState()	Detect has been logged in and trigger login events.
Login Organisation	void ECLoginOrganisation()	Login user organisation to EC backend.

		To login with default user organisation saved in record asset, directly call ECLoginOrganisation().  To Login with custom user organisation, Call SetOrganisationId(int) , SetUserRoleName(string), and SetUserProjectName(string) first, then call ECLoginOrganisation().  Login organisation result output by loginOrganisationEvents.
	void ECSetOrganisationId(int id)	Set a custom orgainsation id for ECLoginOrganisation(). If nothing set, the orgainsation id saved in record asset will be used as default. Params: int - the custom orgainsation id, should be larger than 0.
	void ECSetUserRoleName(string role)	Set a custom user role name for ECLoginOrganisation(). If nothing set, the user role name saved in record asset will be used as default. Params: string - the custom user role name.
	void ECSetUserProjectName(string project)	Set a custom project name for ECLoginOrganisation(). If nothing set, the project saved in record asset will be used as default.

		Params: string - the custom project name.
Get Config	void ECGetConfig()	Download config by scenario id. To get config with default scenario id saved in record asset, directly call ECGetConfig().  To get config with custom scenario id, Call ECSetScenarioId(int) first, then call ECGetConfig().  Get config result output by getConfigEvents.
	void ECSetScenarioId(int id)	Set a custom scenario id for GetConfig(). If nothing set, the scenario id saved in record asset will be used as default. Params: int - the custom scenario id, should be larger than 0.
	void ECSetScenarioId(string id)	Same as ECSetScenarioId(int id) but accepts a string type id. Params: string - the custom scenario id, must can be parse to int type and larger than 0.
Score Getter/Setter	void ECGetScoreInvokeAction()	Trigger a get game score action, result output by recordEvents.onGetRecordBoolScore. ECSetScoreIdAction(int) must be called first to set up parameters.

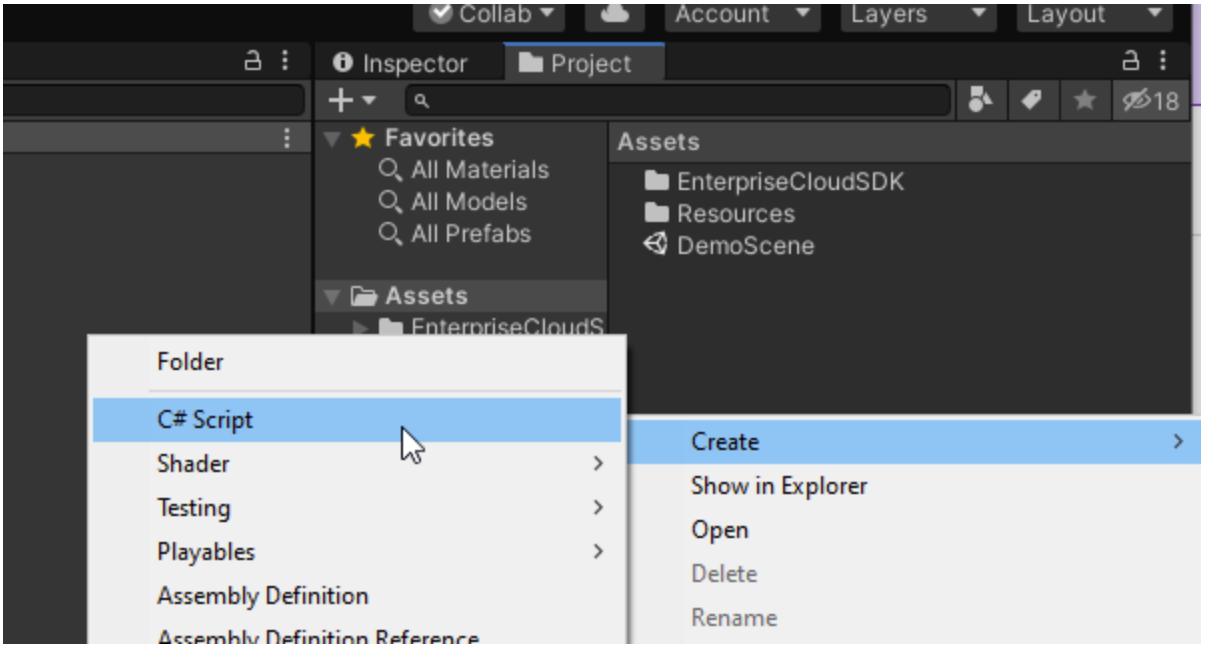
		 The image above shows an example of getting game score from record 529. Get score result output by recordEvents.onGetRecordBoolScore.
	void ECSetScoreInvokeAction()	Trigger a set game score action, result output by recordEvents.onSetRecordBoolScore. ECSetScoreIdAction(int) and ECSetScoreValueAction(bool) must be called first to set up parameters.  The image above shows an example of setting record 529 with a true value game score. Set score result output by recordEvents.onSetRecordBoolScore and recordEvents.setScoreResultEvents
	void ECSetScoreIdAction(int id)	Set a custom record id for ECSetScoreInvokeAction(). Params: int - the custom record id, should be larger than 0.
	void ECSetScoreIdAction(string id)	Same as ECSetScoreIdAction(int id) but accepts a string type id. Params: string - the custom record id, must can be

		parse to int type and larger than 0
	void ECSetScoreValueAction(bool value)	Set a custom game score for ECSetScoreInvokeAction(). Params: bool - the custom game score.
Reset Game Score	void ECResetGameScore()	Reset game scores to default value (false) for all records. 
Submit Score	void ECSubmitScore()	Submit current record game scores to backend.  Submit score result output by submitScoreEvents.

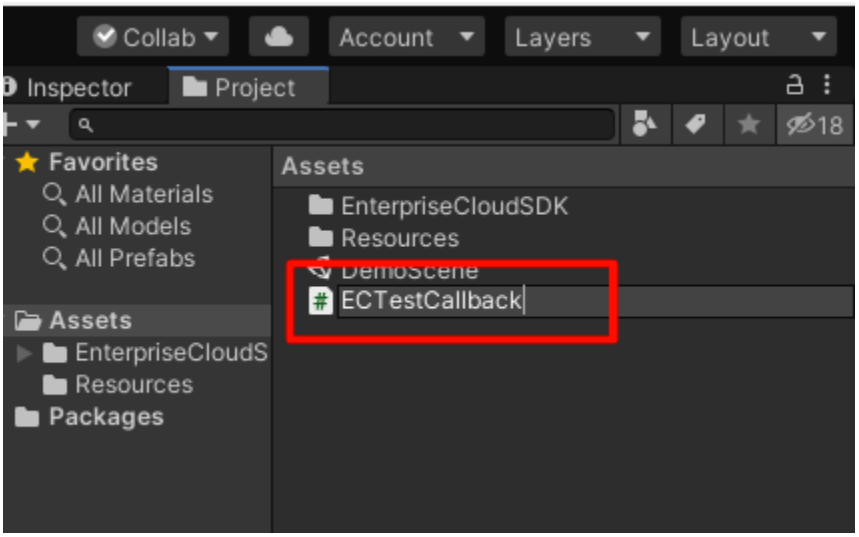
3.Receive ECAPI Callbacks from Inspector

The following example steps show how to receive EC Login event and trigger custom callback methods from ECRecordAgent. With the same setup steps, EC event callbacks can be set to any custom callbacks.

1. Prepare custom callbacks:
    a. Right click in Project window, Select “Create” then click “C# Script”,



b. Rename “NewBehaviourScript” to “ECTestCallback”,

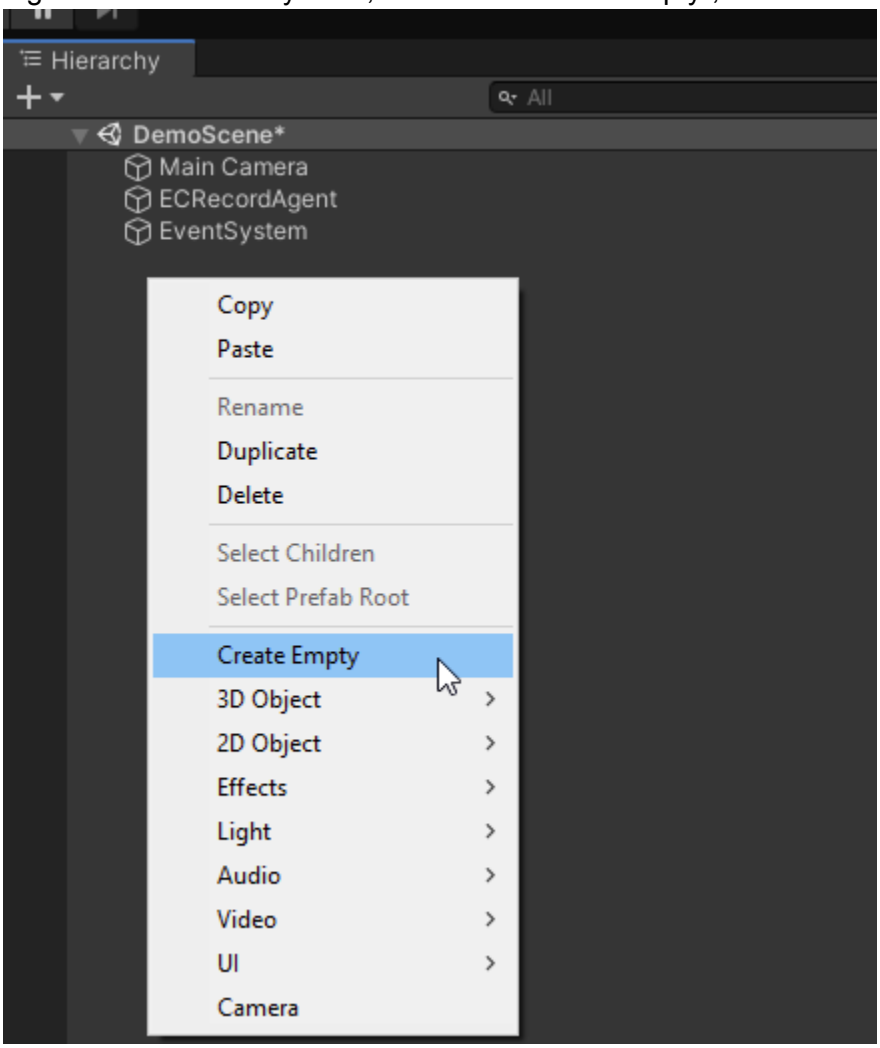


c. Open “ECTestCallback” script, and replace all text with following code:

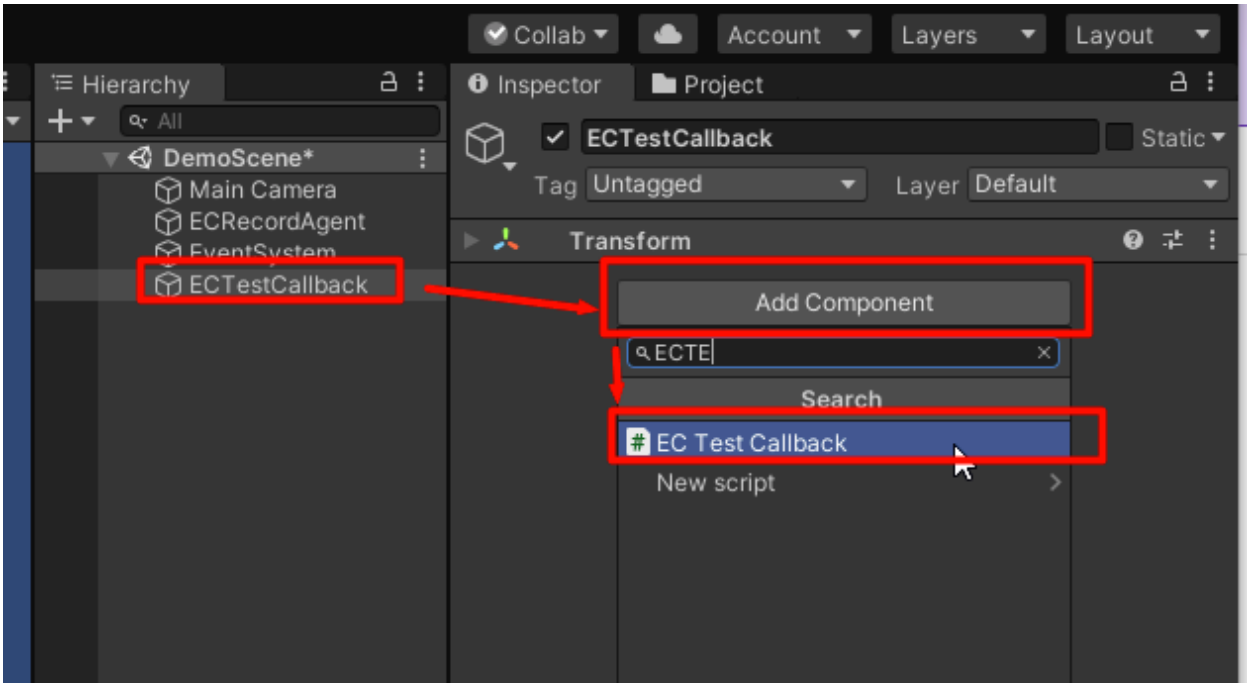
```
using UnityEngine;
using SkillsVR.EnterpriseCloudSDK.Networking.API;

public class ECTestCallback : MonoBehaviour
{
	public void OnReceiveLog(string msg)
	{
		Debug.Log(msg);
	}

	public void OnReceiveError(string msg)
	{
```

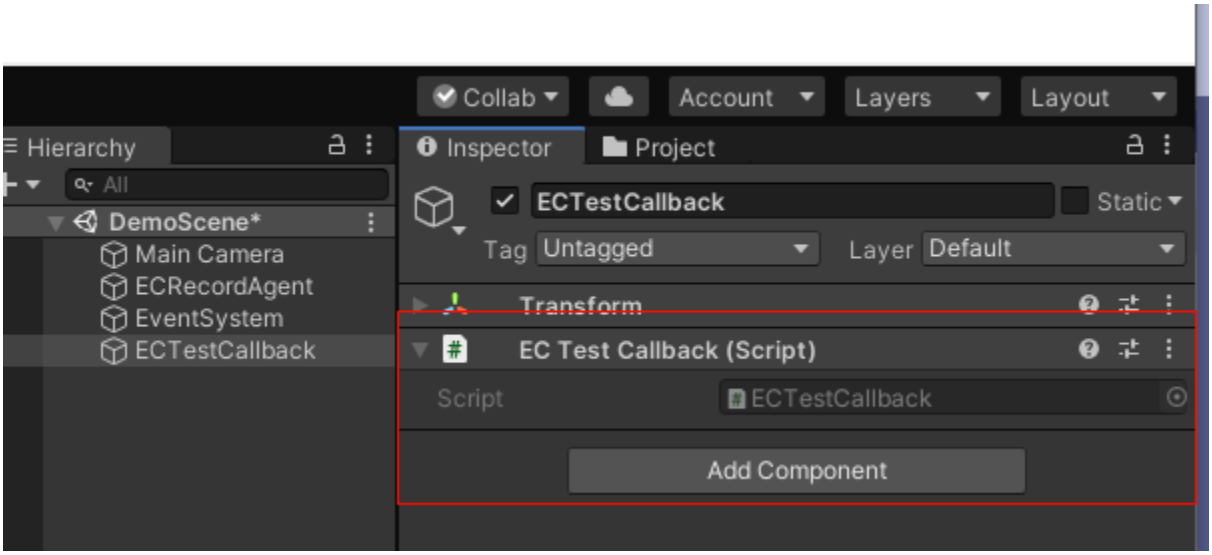
```
                Debug.LogError(msg);
        }

        public void OnReceiveState(bool state)
        {
                Debug.Log("Receive state change: " + state);
        }

        public void OnReceiveResponse(AbstractResponse response)
        {
                Debug.Log("Receive Response " +
response.GetType().FullName);
                Debug.Log(JsonUtility.ToJson(response, true));
        }
}
```

d. Right click in Hierarchy View, then click “Create Empty”;

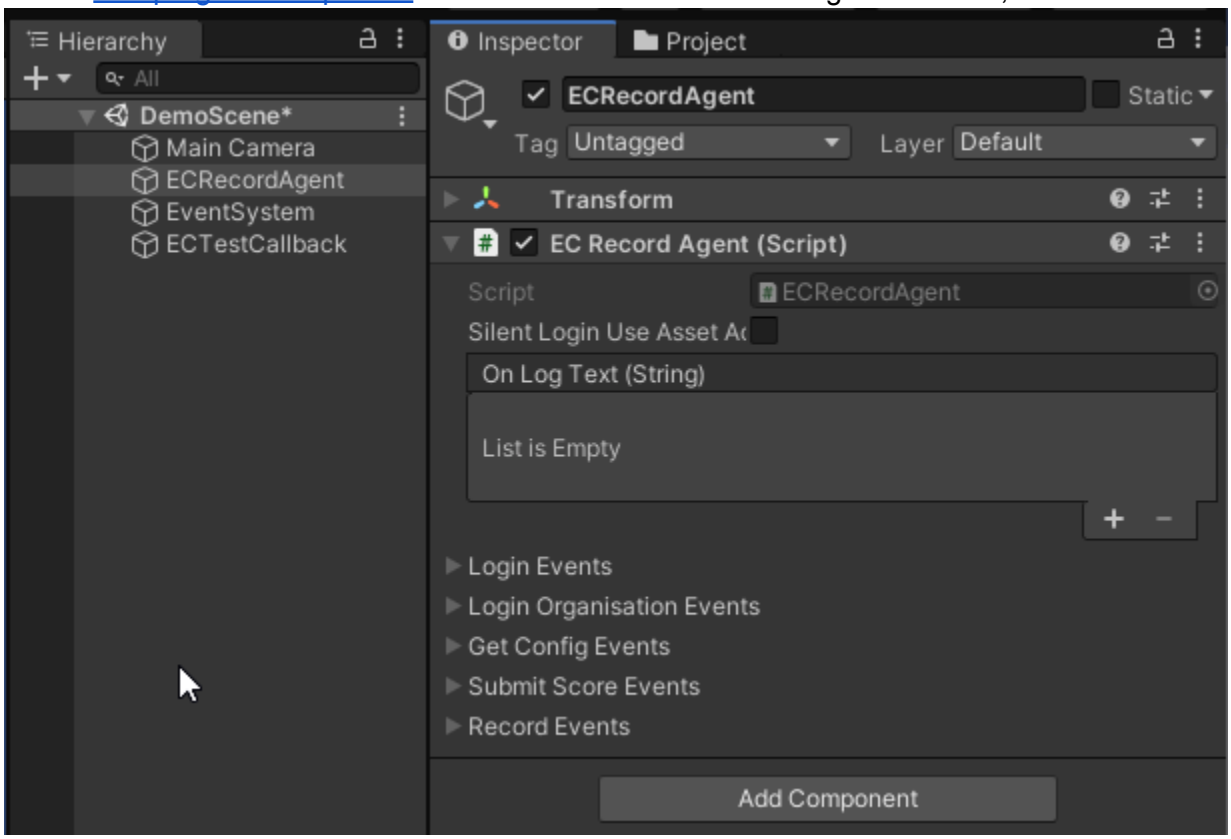


e. Rename new created game object to “ECTestCallback”, select it, click the “Add Component” button in inspector, then find and click “EC Test Callback”;

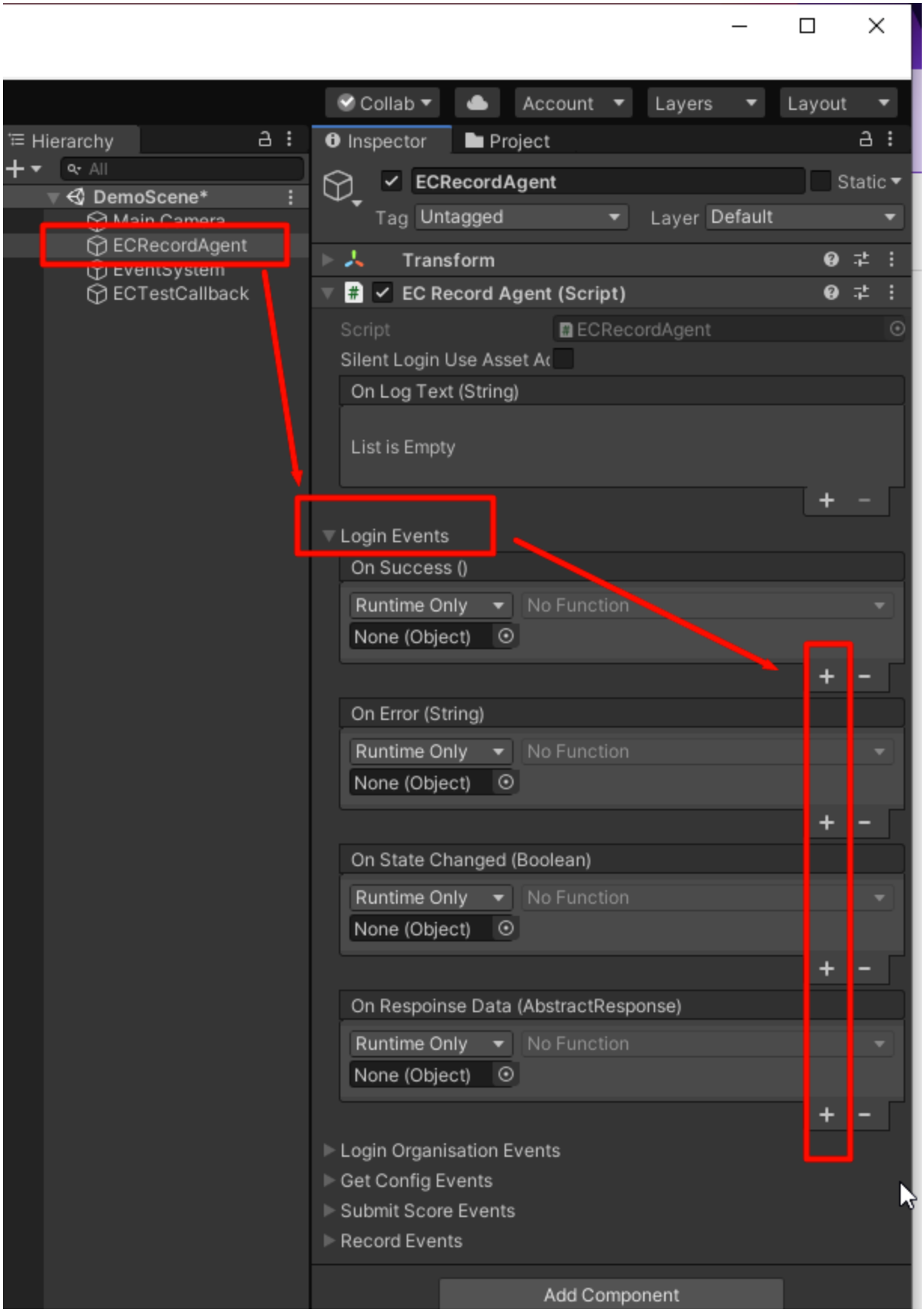




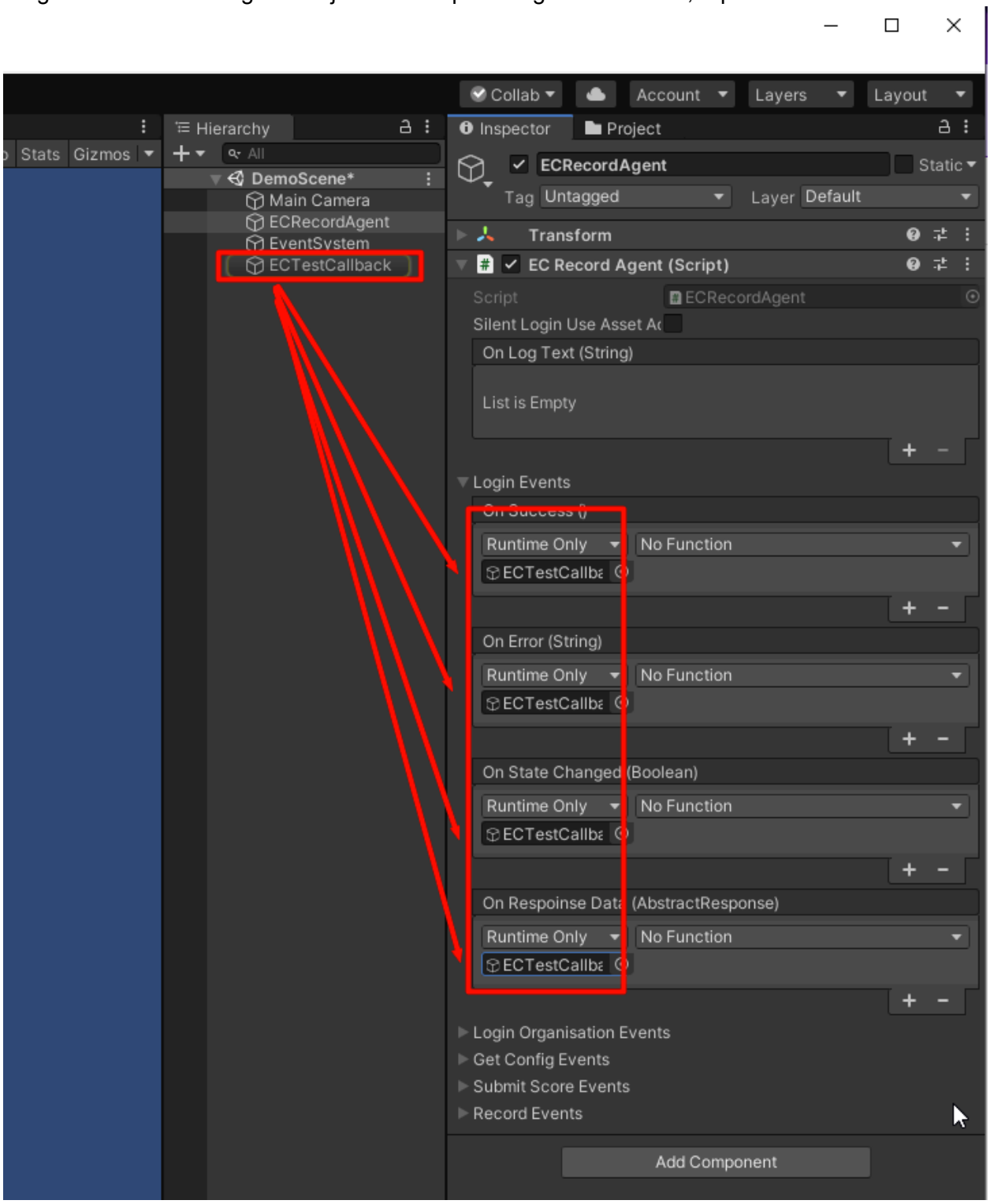
2. Follow Setup Agent Component to create at least 1 ECRecordAgent in scene;



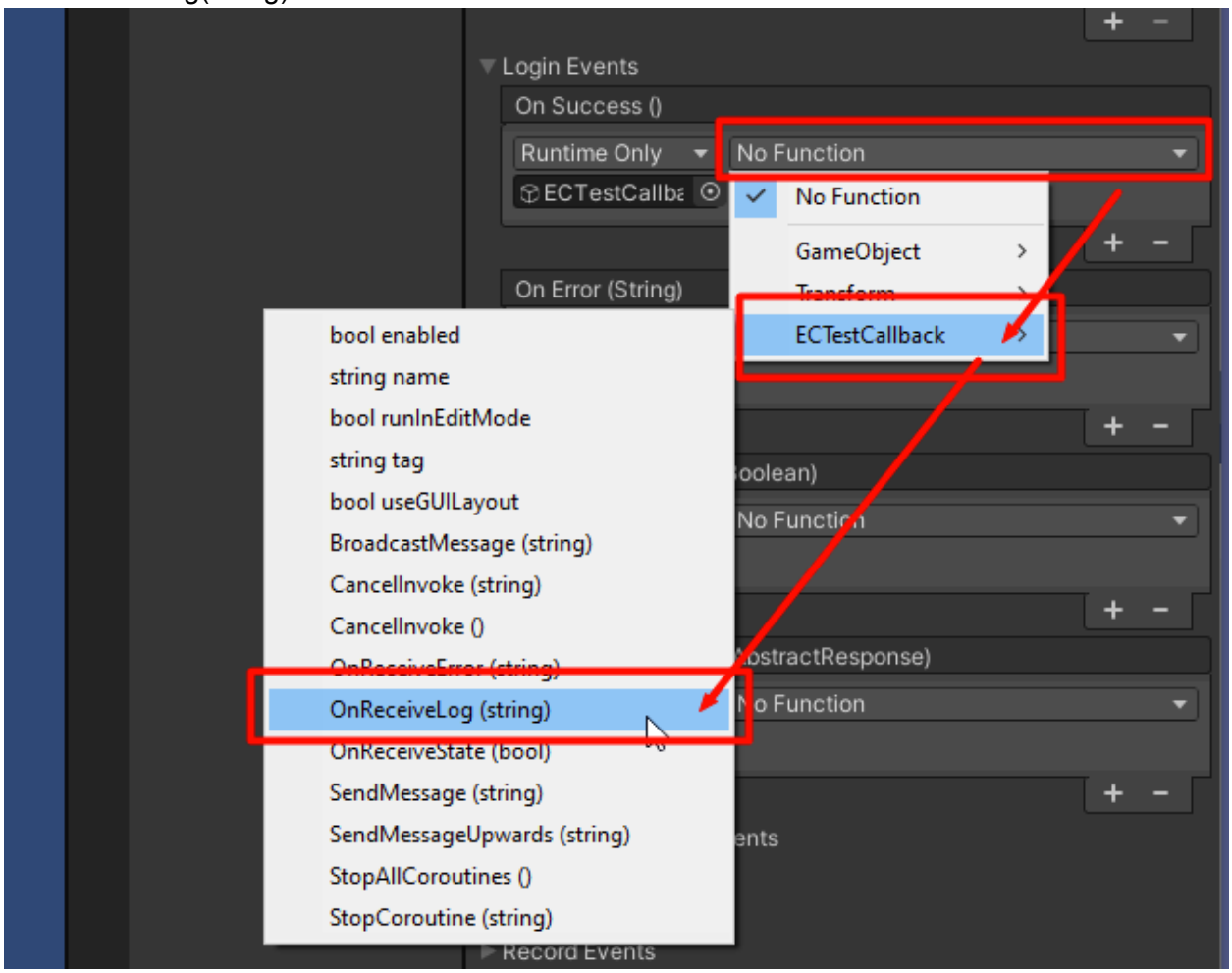
3. Select the “ECRecordAgent” game object, expend “Login Events”, then click all “+” buttons for those 4 events;


Collab
Account
Layers
Layout
Hierarchy
All
DemoScene*
ECRecordAgent
ECTestCallback
Inspector
Project
ECRecordAgent
Static
Tag
Untagged
Layer
Default
Transform
EC Record Agent (Script)
Script
ECRecordAgent
Silent Login Use Asset Ac
On Log Text (String)
List is Empty
Login Events
On Success ()
Runtime Only
No Function
None (Object)
On Error (String)
On State Changed (Boolean)
On Respoinse Data (AbstractResponse)
Login Organisation Events
Get Config Events
Submit Score Events
Record Events
Add Component

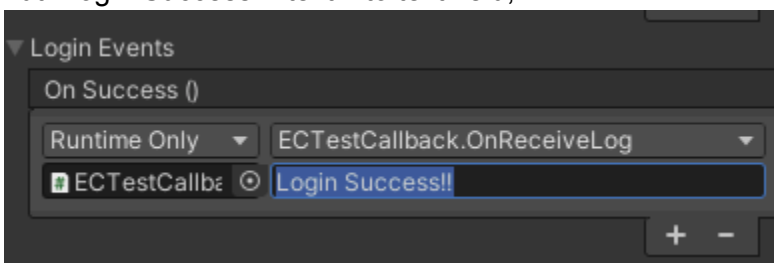
4. Drag “ECTestCallback” game object and drop into login event fields, repeat 4 times:



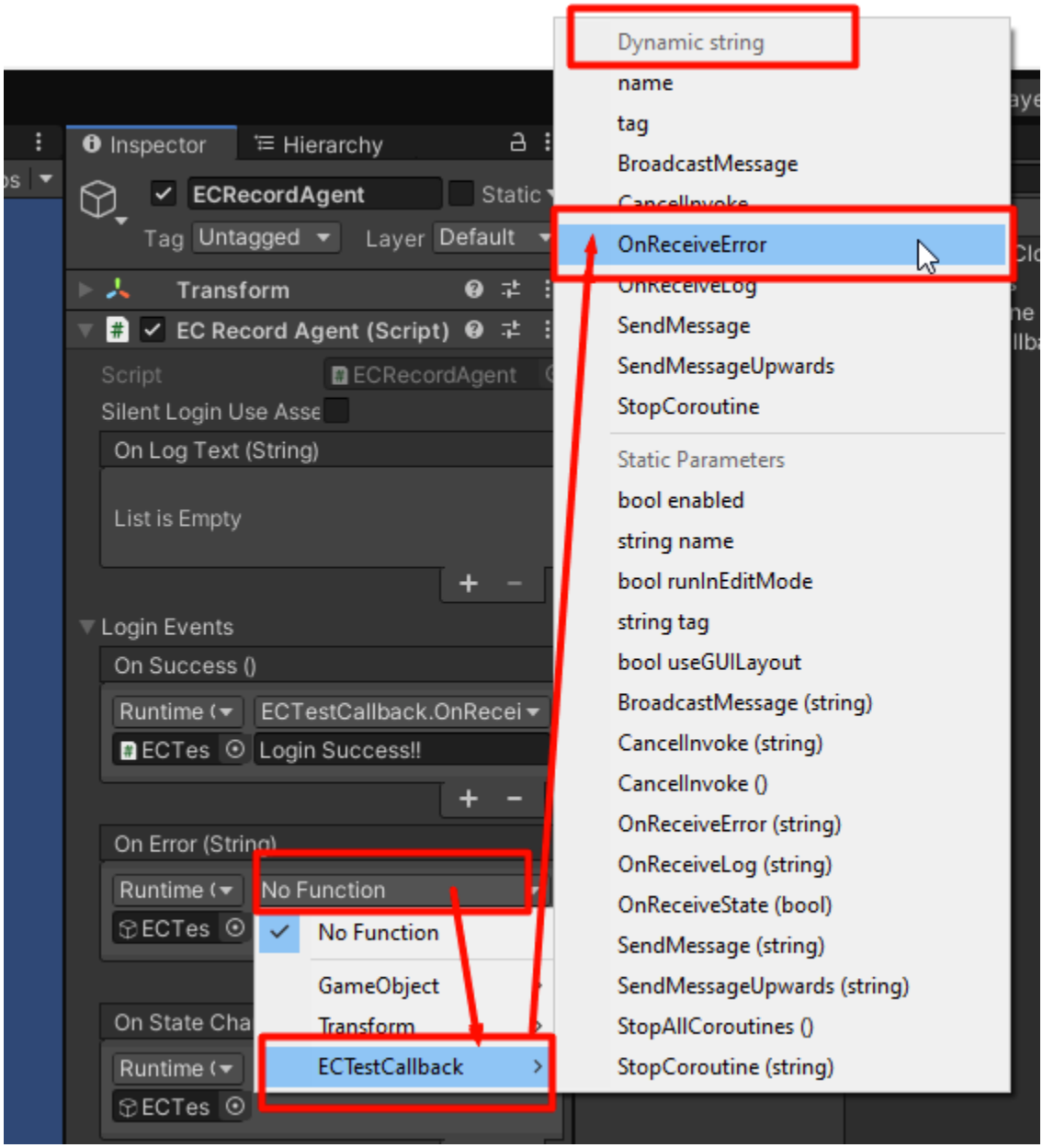
5. At “On Success” event, click “No Function”, select “ECTestCallback”, then click “OnReceiveLog(string)”



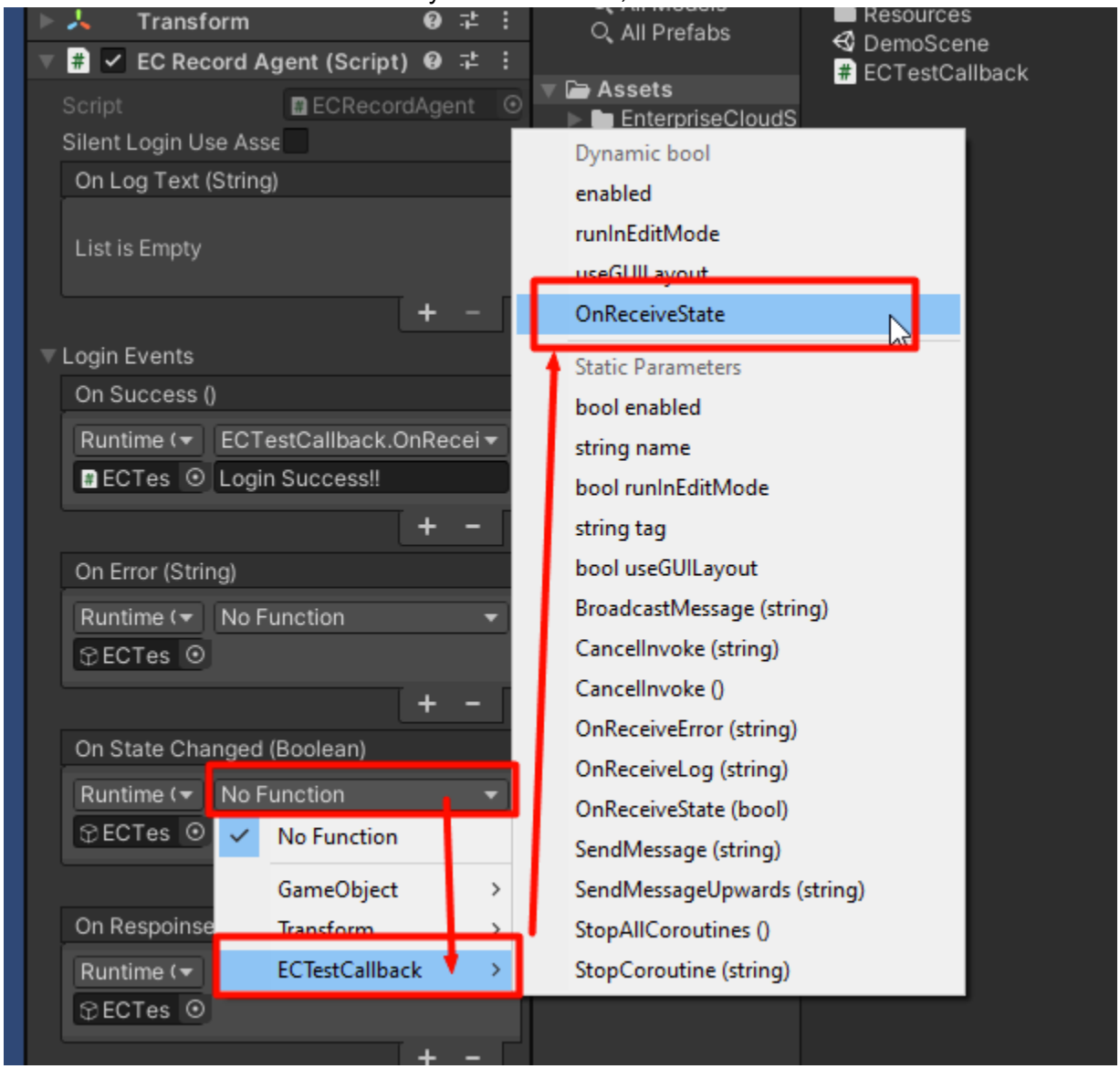
6. Put “Login Success!!” text into text field;



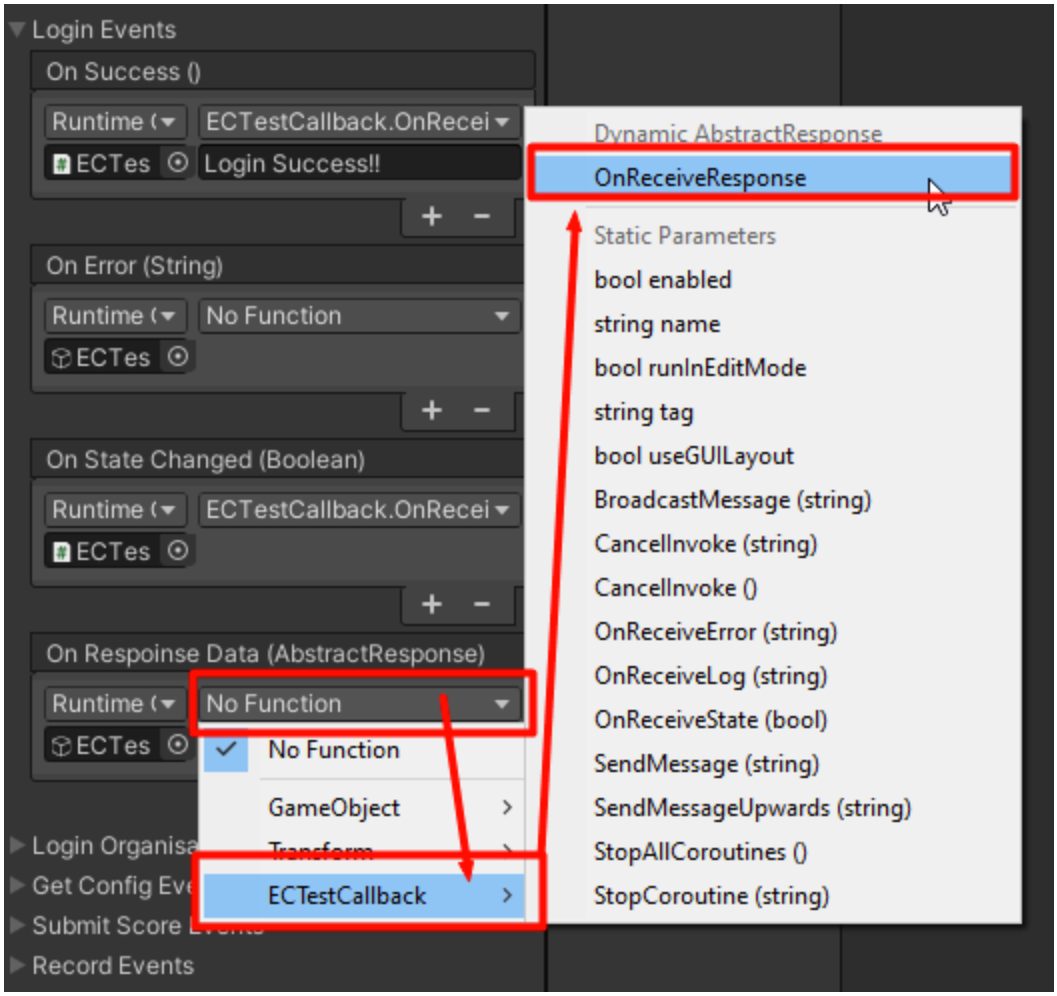
7. At “On Error(String)”, select “No Function”, select “ECTestCallback”, then click the “OnReceiveError” under “Dynamic string” list;



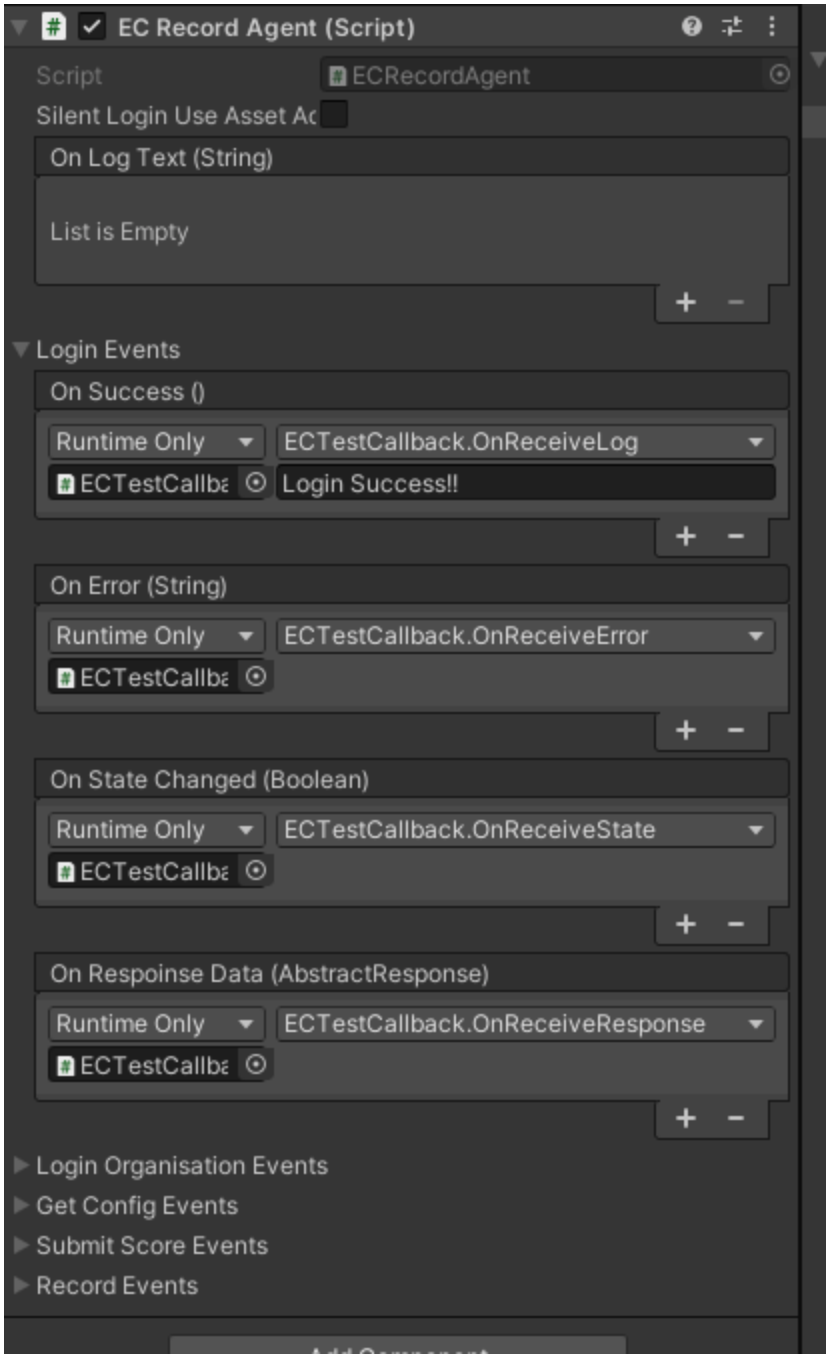
8. At “On State Changed (Boolean)”, select “No Function”, select “ECTestCallback”, then click the “OnReceiveState” under “Dynamic bool” list;



9. At “On State Changed (Boolean)”, select “No Function”, select “ECTestCallback”, then click the “OnReceiveResponse” under “Dynamic AbstractResponse” list;

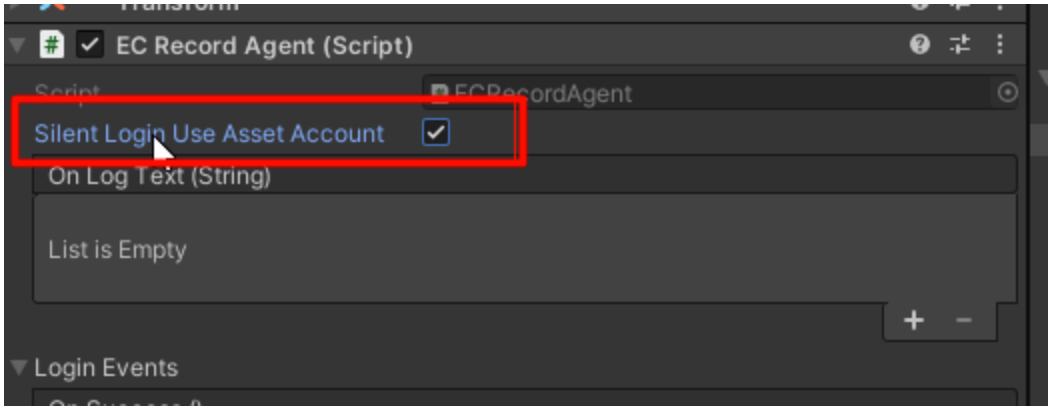


The final agent inspector looks like following image:

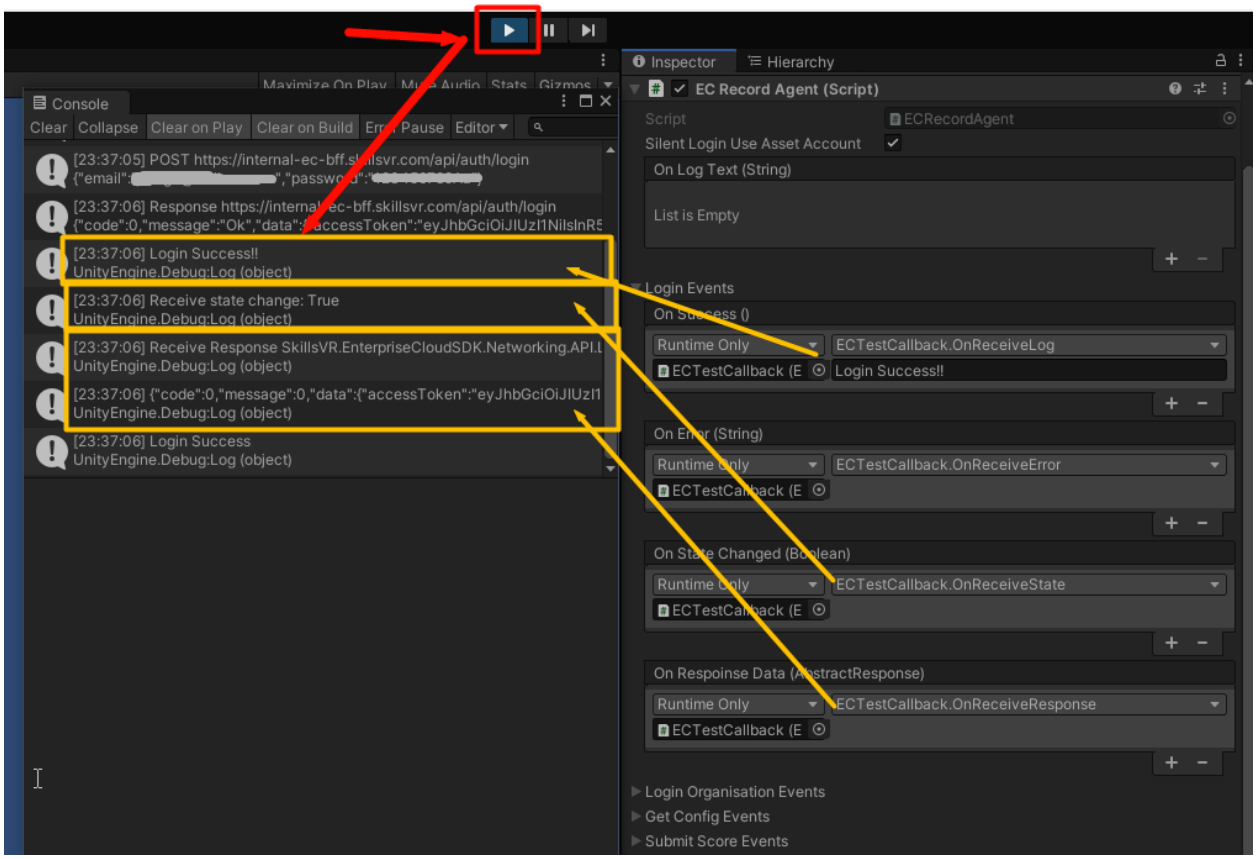


To test the login callbacks, do the following steps:

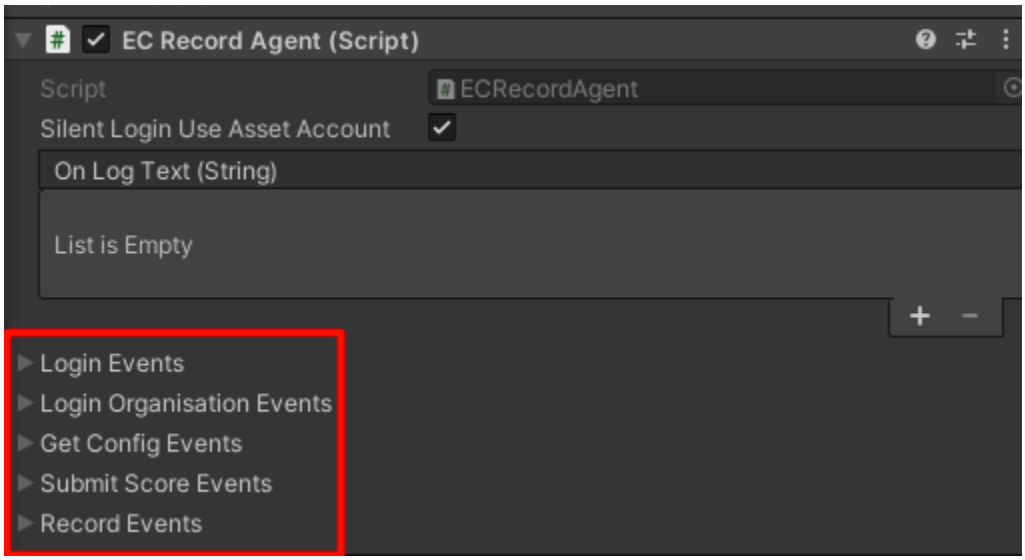
1. Select “ECRecordAgent” game object, tick “Silent Login Use Asset Account” on, this will trigger ECLogin() at agent start;



2. Play game in editor, then find the output log in console.



EC Record Agent Event Reference



Event Group	Event	Description
loginEvents	onSuccess	Triggered when ECLogin() succeeded.
	onError (string)	Triggered when ECLogin() failed. Params: string - Error message.
	onStateChanged (bool)	Triggered when ECLogin() has a result. Params: bool - Does ECLogin() success or not.
	onResponseData (AbstractResponse)	Triggered when ECLogin() has a response. Params: AbstractResponse - Response data object in Login.Response type.
loginOrganisationEvents	onSuccess	Triggered when ECLoginOrganisation() succeeded.
	onError (string)	Triggered when ECLoginOrganisation() failed. Params: string - Error message.
	onStateChanged (bool)	Triggered when ECLoginOrganisation() has a result.

		Params: bool - Does ECLoginOrganisation() success or not.
	onResponseData (AbstractResponse)	Triggered when ECLoginOrganisation() has a response. Params: AbstractResponse - Response data object in Login.Response type.
getConfigEvents	onSuccess	Triggered when ECGetConfig() succeeded.
	onError (string)	Triggered when ECGetConfig() failed. Params: string - Error message.
	onStateChanged (bool)	Triggered when ECGetConfig() has a result. Params: bool - Does ECGetConfig() success or not.
	onResponseData (AbstractResponse)	Triggered when ECGetConfig() has a response. Params: AbstractResponse - Response data object in GetConfig.Response type.
submitScoreEvents	onSuccess	Triggered when ECSubmitScore() succeeded.
	onError (string)	Triggered when ECSubmitScore() failed. Params: string - Error message.
	onStateChanged (bool)	Triggered when ECSubmitScore() has a result. Params: bool - Does ECSubmitScore() success or not.
	onResponseData (AbstractResponse)	Triggered when ECSubmitScore() has a response.

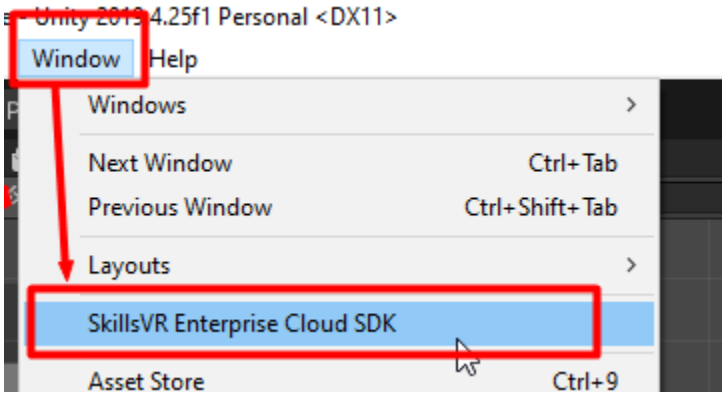
		Params: AbstractResponse - Response data object in AbstractAPI.EmptyResponse type.
recordEvents	onResetAllGameScores	Triggered when ECResetGameScore() or ECAPI.ResetAllUserScores() invoked. This is a record asset based event, all agents will receive this when reset.
	onRecordStateChanged (int)	Triggered when a record has game score changes. This is a record asset based event, all agents will receive this when the record changes. Params: int - the id of changed record
	onRecordBoolScoreChanged (int, bool)	Triggered when a record has game score changes. This is a record asset based event, all agents will receive this when the record changes. Params: int - the id of changed record. bool - the new value of game score in boolean type.
	onGetRecordBoolScore (int, bool)	Triggered when ECGetScoreInvokeAction() invoked. Params: int - the id of checking record. bool - the value of game score in boolean type
	onSetRecordBoolScore (int, bool)	Triggered when ECSetScoreInvokeAction() invoked. Params: int - the id of checking record. bool - the value of game score in boolean type
recordEvents.setScoreResultEvents	onSuccess	Triggered when ECSetScoreInvokeAction() succeeded.

	onError (string)	Triggered when ECSetScoreInvokeAction() failed. Params: string - Error message.
	onStateChanged (bool)	Triggered when ECSetScoreInvokeAction() has a result. Params: bool - Does ECSetScoreInvokeAction() success or not.

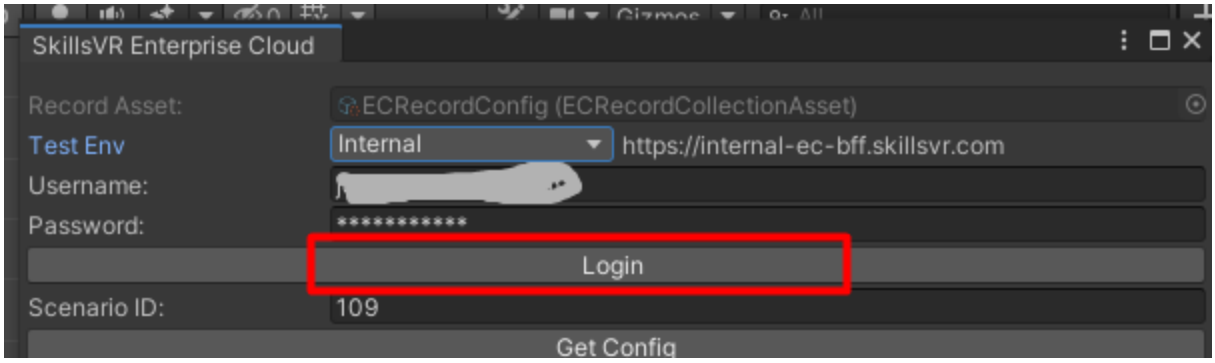
Step 6: Test SDK / View Reports

To test submit scores,

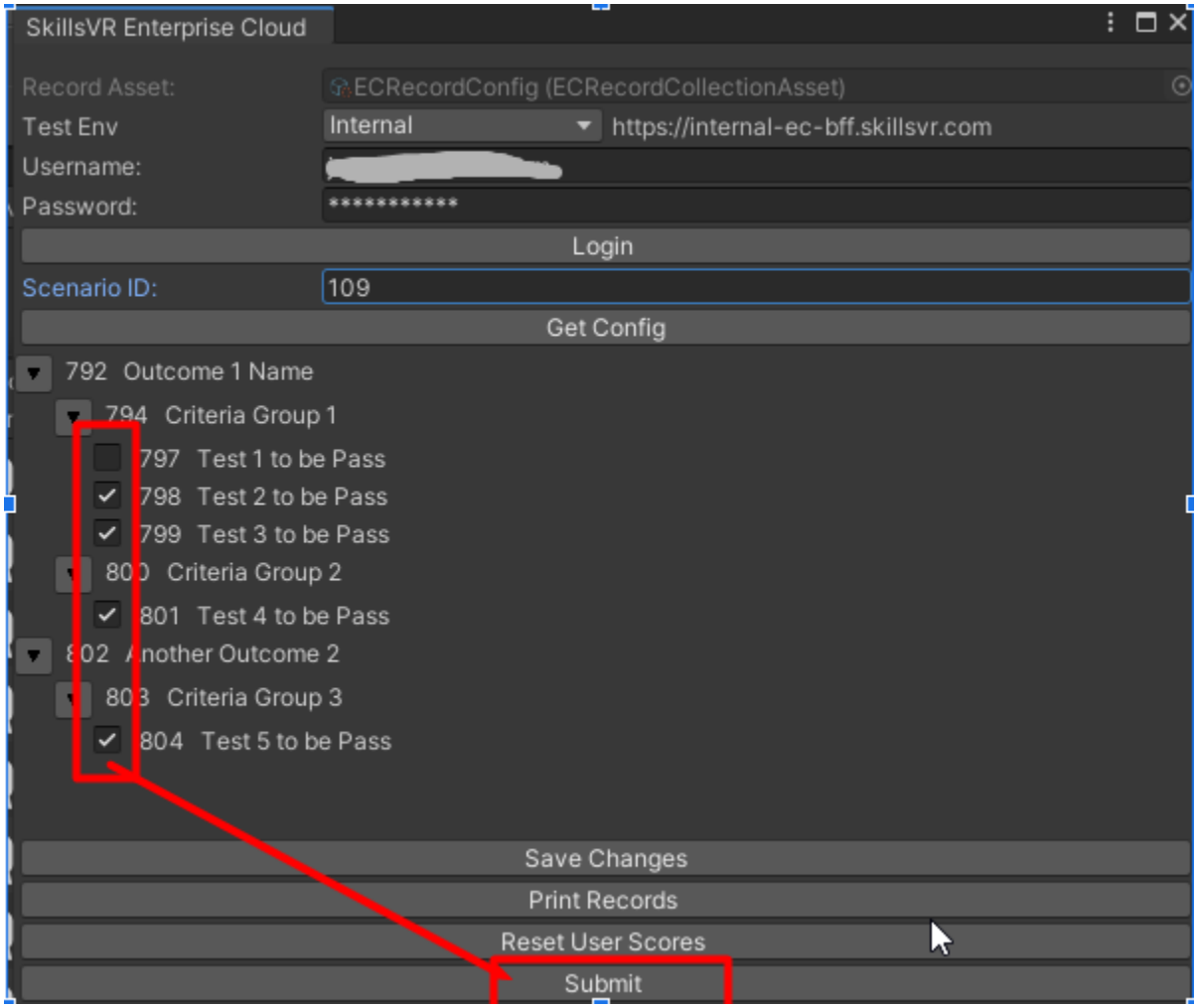
1. Make sure Step 3: Set Up SDK Configuration inside Unity is completed first;
2. Open SDK editor window;



3. Login user;



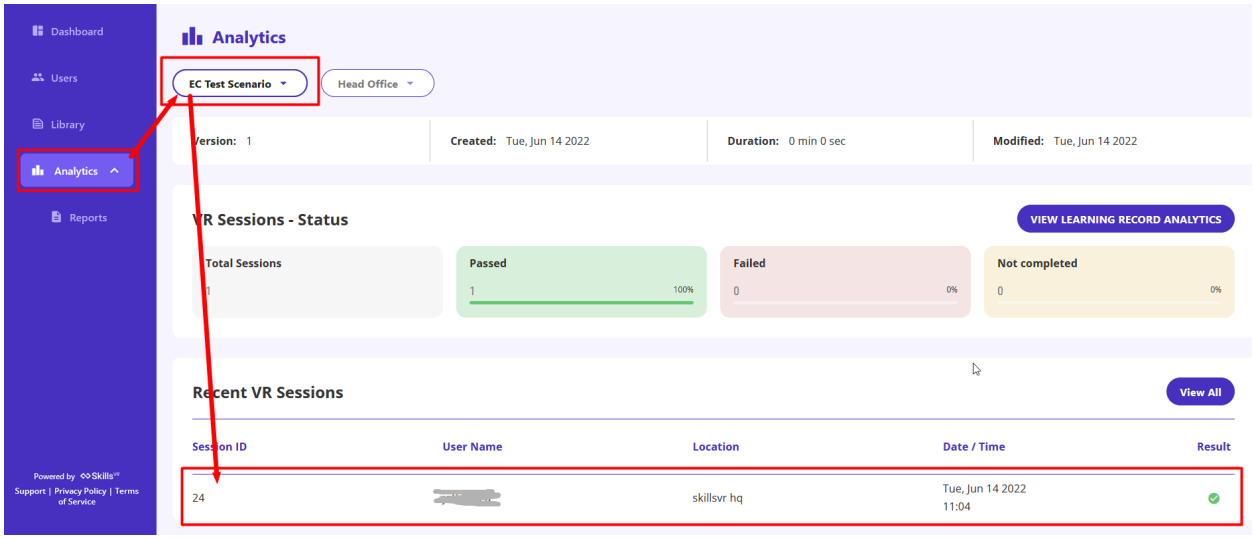
4. tick toggles at any of the records, then click the “Submit’.

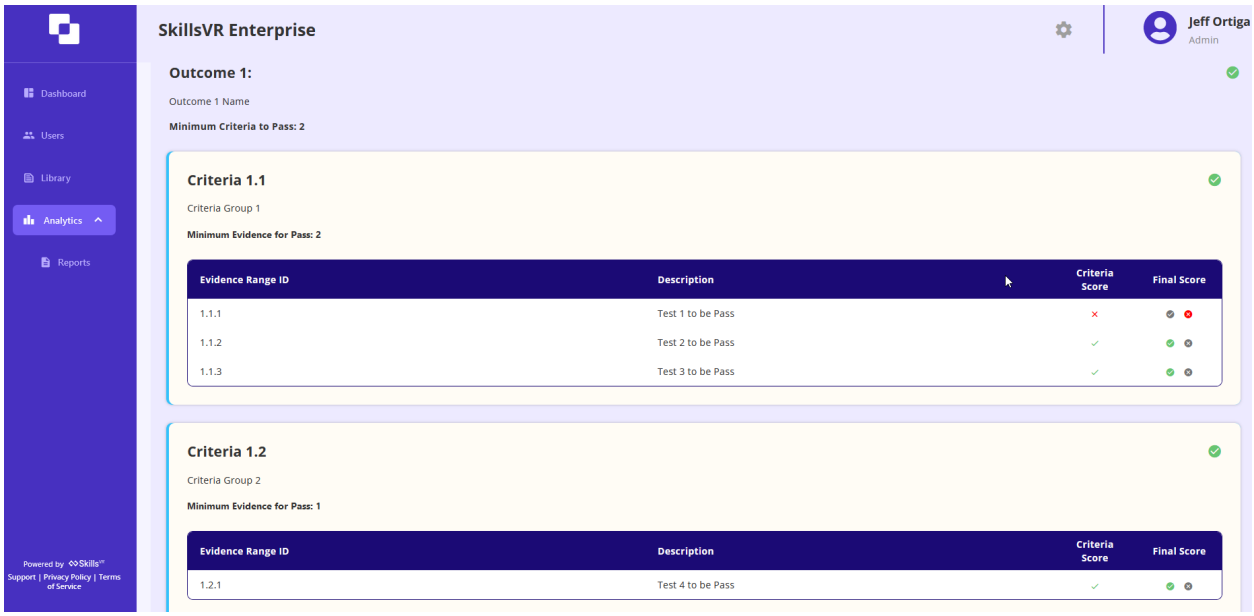


To view the submitted results,

1. go to the portal,
2. then click “Analytics”,
3. select the scenario name from the dropdown (in this tutorial select the “EC Test Scenario”),

4. then click one of the sessions listed in the “Recent VR Sessions”.





Extras about the SDK

Any extra information needed to know about the SDK

Troubleshooting and Extras

Links or information on things that might go wrong when importing and using the package