
Alexa Plus Unity Documentation

Release August 2018

Austin Wilson

Oct 07, 2018

GETTING STARTED

1	AWS Configuration	1
1.1	Prerequisites	1
1.2	Obtain an Identity Pool ID using Amazon Cognito	1
1.3	Attach Policies to the Identity Pool default roles in AWS IAM	1
2	PubNub Configuration	3
2.1	Prerequisites	3
2.2	Create a New App on PubNub	3
3	Setup for Unity	5
3.1	Prerequisites	5
3.2	Integrating the Alexa Plus Unity SDK into your Unity project	5
4	Setup for Alexa Skills	7
4.1	Prerequisites	7
4.2	Integrating the Alexa Plus Unity SDK into your Alexa Skill	7
5	Introduction	9
6	Creating the Unity Project	11
6.1	Prerequisites	11
6.2	Creating the project	11
6.3	Integrating the Alexa Plus Unity Package	11
6.4	Adding the LightControl script into the Unity project	12
6.5	Adding the Alexa Manager GameObject in Unity	16
6.6	Wrapping Up	17
7	Creating the Alexa Skill	19
7.1	Prerequisites	19
7.2	Creating the skill	19
7.3	Example Skill Template Overview	19
7.4	Adding AlexaPlusUnity to the skill	20
7.5	Deploying the Skill	22
7.6	Wrapping Up	23
8	Conclusion	25
8.1	Testing the Light Control Demo	25

9	Unity C# Technical Docs	27
9.1	Classes	27
9.2	AlexaBaseData	27
9.3	AmazonAlexaManager	28
9.4	ConnectionStatusEventData	31
9.5	ErrorEventData	32
9.6	GetSessionAttributesEventData	33
9.7	HandleMessageEventData	34
9.8	MessageSentEventData	35
9.9	SetSessionAttributesEventData	36
10	Alexa Plus Unity	39
10.1	The Explorer Demo	39

For Alexa Plus Unity to function, we need to configure AWS to allow the Unity project to communicate to DynamoDB (The Alexa Skill's persistent attributes).

1.1 Prerequisites

- An [AWS Account](#)

1.2 Obtain an Identity Pool ID using Amazon Cognito

1. Log in to the [Amazon Cognito Console](#) and click **Create new identity pool**.
2. Enter a name for your Identity Pool and check the checkbox to enable access to unauthenticated identities. Click **Create Pool** to create your identity pool.
3. Click **Allow** to create the two default roles associated with your identity pool—one for unauthenticated users and one for authenticated users. These default roles provide your identity pool access to Cognito Sync and Mobile Analytics.

The next page displays code. Take note of the displayed **Identity Pool ID** and the **Region** you set up the Identity Pool in as you will need them when setting up Alexa Plus Unity.

1.3 Attach Policies to the Identity Pool default roles in AWS IAM

1. Log in to the [AWS IAM Console](#) and click **Roles** in the left navigation bar.
2. Find and click your **Unauthenticated** Identity Pool role. It should look similar to `Cognito_[YOUR IDENTITY POOL]Unauth_Role`.
3. Click **Attach Policies**.

4. Find and check the **AmazonDynamoDBFullAccess** policy.
5. Click **Attach Policy**.

Your Identity Pool is now configured to use the required AWS services for Alexa Plus Unity to function.

PubNub Configuration

For Alexa Plus Unity to function, we need to configure PubNub to allow the Unity project to communicate with the Alexa Skill.

2.1 Prerequisites

- A [PubNub Account](#)

2.2 Create a New App on PubNub

1. Log in to the [PubNub Admin Console](#) and click **Create new app**.
2. Enter a name for your new app. Click **Create** to create your new app.
3. Click on your new app in the admin console.

The next page displays your keysets. You can create as many as you keysets as you wish, but for our purposes, we can just use the Demo Keyset.

4. Click the Demo Keyset.
5. Make note of both the publish and subscribe keys as you will need them when setting up Alexa Plus Unity.
6. Enable the **Stream Controller** and the **Storage and Playback** Application add-ons.

Your PubNub App is now configured for Alexa Plus Unity.

Let's setup the Alexa Plus SDK for Unity!

3.1 Prerequisites

- [Unity](#) version 4.x or above.
- Configure AWS as explained [here <https://alexaplusunity.readthedocs.io/en/latest/AWS-SetUp.html>](https://alexaplusunity.readthedocs.io/en/latest/AWS-SetUp.html)‘_.

3.2 Integrating the Alexa Plus Unity SDK into your Unity project

1. Download the `AlexaPlusUnity.unitypackage` from the [Releases](#) tab in the GitHub project.
2. Open your project in Unity.
3. Open your newly downloaded `AlexaPlusUnity.unitypackage` and import the package.

That's it! However, in order to handle the communication to and from Alexa, you need to create your own script to initialize the manager. See a Tutorial for a more in-depth implementation.

Setup for Alexa Skills

Now that we have configured our Unity workspace, it's time to set up the Alexa Skill!

4.1 Prerequisites

- An [NPM](#) project. For information on how to set up a NPM project, please see [this](#).
- A suitable Node.js development environment. The ASK SDK v2 for Node.js requires Node 4.3.2 or above.

4.2 Integrating the Alexa Plus Unity SDK into your Alexa Skill

1. Navigate to your skill function in an command prompt or terminal
2. Install the AlexaPlusUnity package into your skill's function: `npm install alexaplusunity`
3. Include the package in your skill with:

```
var alexaPlusUnityClass = require('alexaplusunity');
```

4. Create an instance of the class with:

```
var alexaPlusUnity = alexaPlusUnityClass("<YOUR_PUBNUB_PUB_KEY>", "<YOUR_PUBNUB_↪SUB_KEY>", true); //Third parameter enables verbose logging
```

That's it! However, in order to handle the communication to and from your game, you need to use the script's methods. See a [Tutorial](#) or the [package page](#) for more in-depth information.

CHAPTER 5

Introduction

Looking to learn how to use the the Alexa Plus Unity SDK with an in-depth example? You've come to the right place!

In this tutorial, we will walk through the steps to create a basic game in which we'll be able to manipulate a light with both our keyboard and voice, all while having Alexa remain aware of the game's state.

Creating the Unity Project

Let's setup the Unity Project!

If you get lost or want to see the final project, the light control demo scene is located in `Amazon Plus Unity\Examples`. You will need to create a project and import `AlexaPlusUnity.unitypackage` to gain access to this folder.

6.1 Prerequisites

- [Unity](#) version 4.x or above.
- Configure AWS and PubNub as explained [here](#).

6.2 Creating the project

1. Open Unity.
2. Create a new 3D project.
3. Add a cube to the SampleScene (GameObject -> 3D Object -> Cube).

6.3 Integrating the Alexa Plus Unity Package

1. Download Alexa Plus Unity from the [Unity Asset Store](#).
2. Add the asset package to your Unity project.
3. Make sure everything is checked and click **Import**.

You should now see a new folder in you Assets folder, **Alexa Plus Unity**.

6.4 Adding the LightControl script into the Unity project

1. With your project **open**, create a new script (Assets -> Create -> C# Script) and name it **LightControl**.
2. Open your LightControl script by double-clicking it.
3. Delete everything in the script.
4. Copy and paste the code below into your LightControl script:

```
using Amazon;
using Amazon.DynamoDBv2.Model;
using AmazonsAlexa.Unity.AlexaCommunicationModule;
using System.Collections.Generic;
using UnityEngine;

public class LightControl : MonoBehaviour
{
    //Step 5: Add variables below

    // Use this for initialization
    void Start () {
        //Step 6: Add AlexaPlusUnity initialization
    }

    public void ConfirmSetup(GetSessionAttributesEventData eventData)
    {
        //Step 7: Notify the skill that setup has completed by updating the
        ↳skills persistant attributes (in DynamoDB)
    }

    void Update() {
        //Step 8: Add Keyboard event listener (For "gameplay")
    }

    public void HandleSpacePress()
    {
        //Step 9: Handle the keyboard input
    }

    //Callback for when a message is recieved
    public void OnAlexaMessage(HandleMessageEventData eventData)
    {
        //Step 10: Listen for new messages from the Alexa skill
    }

    private void GetObjectInDirection(string type, string message)
    {
        //Step 11: Get the object in a specific direction (Note: For this demo,
        ↳there is only one object, the cube)
    }
}
```

(continues on next page)

(continued from previous page)

```

    }

    public void UpdateLight(string type, string value,
↪GetSessionAttributesEventData eventData)
    {
        //Step 12: Update the light based on the incoming message, then save the
↪state of the light through the skill's session attributes

    }

    public void SetAttributesCallback(SetSessionAttributesEventData eventData)
    {
        //Step 13: Callback for when session attributes have been updated

    }

    public void OnMessageDeleted(ErrorEventData eventData)
    {
        //Step 14: Callback for when a message is deleted

    }
}

```

The above code is our skeleton for our script. We will fill this skeleton step by step. The steps below correspond to the step numbers in the skeleton. Place the code for each of the below steps under their step number in the skeleton.

Note: There may be IDE errors as we continue, but those will be resolved at the end when the skeleton is complete.

5. Define the class variables:

```

public string publishKey;
public string subscribeKey;
public string channel;
public string tableName;
public string identityPoolId;
public string AWSRegion = RegionEndpoint.USEast1.SystemName;
public bool debug = false;
public GameObject lightCube;
public GameObject camera;

private Dictionary<string, AttributeValue> attributes;
private AmazonAlexaManager alexaManager;

```

These variables are necessary to preform initialization and enable reusability of the Alexa Manager within our LightControl script.

6. Initialize the Alexa Manager:

```

UnityInitializer.AttachToGameObject(gameObject);
AWSConfigs.HttpClient = AWSConfigs.HttpClientOption.UnityWebRequest;
alexamanager = new AmazonAlexaManager(publishKey, subscribeKey, channel,
↪tableName, identityPoolId, AWSRegion, this.gameObject, OnAlexaMessage, null,
↪debug); //Initialize the Alexa Manager

```

7. Tell the skill that the game has completed setup and is ready to play:

```
attributes = eventData.Values;
attributes["SETUP_STATE"] = new AttributeValue { S = "COMPLETED" }; //Set SETUP_
↪STATE attribute to a string, COMPLETED
alexamanager.SetSessionAttributes(attributes, SetAttributesCallback);
```

8. Listen for a spacebar keypress:

```
if (Input.GetKeyDown(KeyCode.Space))
{
    Debug.Log("Space pressed");
    HandleSpacePress();
}
```

9. Update the light to blue when the spacebar is pressed:

```
if (!PlayerPrefs.HasKey("alexidUserDynamoKey")) //If the AlexaUserId has not been_
↪recieved from Alexa (If the user has not opened the skill)
{
    Debug.LogError("'alexidUserDynamoKey' not found in PlayerPrefs. We must_
↪establish connection from Alexa to set this. Please open the skill to set the
↪'AlexaUserId' PlayerPrefs.");
} else {
    alexamanager.GetSessionAttributes((result) =>
    {
        if (result.IsError)
            Debug.LogError(result.Exception.Message);
        UpdateLight("Color", "blue", result);
    });
}
```

10. Listen for new messages from the Alexa skill:

```
Debug.Log("OnAlexaMessage");

Dictionary<string, object> message = eventData.Message;

//Get Session Attributes with in-line defined callback
switch (message["type"] as string)
{
    case "AlexaUserId":
        Debug.Log("AlexaUserId: " + message["message"]);
        alexamanager.alexidUserDynamoKey = message["message"] as string;
        break;
}

alexamanager.GetSessionAttributes((result) =>
{
    if (result.IsError)
        Debug.LogError(eventData.Exception.Message);

    switch (message["type"] as string)
    {
        case "AlexaUserId":
            ConfirmSetup(result);
            break;
        case "Color":
            Debug.Log("Requested Light Color: " + message["message"]);
```

(continues on next page)

(continued from previous page)

```

        UpdateLight(message["type"] as string, message["message"] as string,
↪result);
        break;
    case "State":
        Debug.Log("Requested Light State: " + message["message"]);
        UpdateLight(message["type"] as string, message["message"] as string,
↪result);
        break;
    case "GetObject":
        Debug.Log("Requested object direction: " + message["message"]);
        GetObjectInDirection(message["type"] as string, message["message"] as
↪string);
        break;
    default:
        break;
    }
});

```

11. Get object in a direction:

```

RaycastHit hit;
Dictionary<string, string> messageToAlexa = new Dictionary<string, string>();
Vector3 forward = camera.transform.forward * 10;
messageToAlexa.Add("object", "nothing");

if (Physics.Raycast(camera.transform.position, forward, out hit, (float)15.0))
{
    if (hit.rigidbody)
    {
        messageToAlexa.Remove("object");
        messageToAlexa.Add("object", hit.rigidbody.name);
    }
}

alexamanager.SendToAlexaSkill(messageToAlexa, OnMessageSent);

```

12. Update the light:

```

attributes = eventData.Values;
if (type == "Color")
{
    attributes["color"] = new AttributeValue { S = value }; //Set color attribute
↪to a string value
}
else if (type == "State")
{
    attributes["state"] = new AttributeValue { S = value }; //Set state attribute
↪to a string value
}

switch (value)
{
    case "white":
        lightCube.GetComponent<Renderer>().material.color = Color.white;
        break;
    case "red":
        lightCube.GetComponent<Renderer>().material.color = Color.red;

```

(continues on next page)

(continued from previous page)

```

        break;
    case "green":
        lightCube.GetComponent<Renderer>().material.color = Color.green;
        break;
    case "yellow":
        lightCube.GetComponent<Renderer>().material.color = Color.yellow;
        break;
    case "blue":
        lightCube.GetComponent<Renderer>().material.color = Color.blue;
        break;
    case "on":
        lightCube.GetComponent<Renderer>().enabled = true;
        break;
    case "off":
        lightCube.GetComponent<Renderer>().enabled = false;
        break;
}
alexamaManager.SetSessionAttributes(attributes, SetAttributesCallback); //Save
↳Attributes for Alexa to use

```

13. Let's be notified when there is a error setting the attributes:

```

Debug.Log("OnSetAttributes");
if (eventData.IsError)
    Debug.LogError(eventData.Exception.Message);

```

14. Let's be notified when there is a error deleting a message:

```

Debug.Log("OnMessageSent");
if (eventData.IsError)
    Debug.LogError(eventData.Exception.Message);

```

15. Be sure to save this file!

6.5 Adding the Alexa Manager GameObject in Unity

1. Create a new **Empty GameObject** (GameObject -> Create Empty) and name it **Amazon Alexa**.
2. With your new GameObject selected, click **Add Component**, type **LightControl** and select the LightControl script.
3. Fill the Publish Key with the PubNub publish key you made note of during configuration.
4. Fill the Subscribe Key with the PubNub subscribe key you made note of during configuration.
5. Fill the Channel with the code sent from the Alexa skill when it launches.

Note: You will have to fill this in later, as we have not set up the Alexa skill yet.

6. Fill the Table Name with **AlexaPlusUnityTest**.
7. Fill the Identity Pool Id with the one you created during configuration.
8. Fill the AWS Region with the one you made note of during configuration.
9. Check the box next to Debug to enable detailed logging.

10. Drag the **Cube** from the hierarchy into the box next to `Light Cube`.
11. Drag the **Main Camera** from the hierarchy into the box next to `Camera`.

6.6 Wrapping Up

Aside from a few minor updates, we have finished the Unity project! Next Step: The Alexa Skill!

Creating the Alexa Skill

Let's setup the Alexa Skill!

If you get lost, stuck, or just want a working demo right now, you can clone the complete sample project from [here](#).

7.1 Prerequisites

- Node.js (v4.5 or above)
- Have an [AWS Account](#)
- Have an [Amazon Developer Account](#)
- Install the [ASK CLI](#)

7.2 Creating the skill

1. Clone the example skill template: `git clone https://github.com/AustinMathuw/AlexaPlusUnityExampleSkillTemplate.git`
2. Open the template in a editor such as Visual Studio code
3. Open a command prompt or terminal and navigate to <Template Location>/lambda/custom/
4. In the command prompt or terminal, run `npm install`

7.3 Example Skill Template Overview

Our Example Skill has the following Intents:

- FlipSwitchIntent
 - Handles turning our light on and off

- `ChangeColorIntent`
 - Handles changing the color of our light
- `GetColorIntent`
 - Returns our lights current color
- `AMAZON.HelpIntent`
 - Returns help message to guide the user
- `AMAZON.CancelIntent`
 - Closes the skill
- `AMAZON.StopIntent`
 - Closes the skill

7.4 Adding AlexaPlusUnity to the skill

The steps below correspond to the step numbers in the skeleton. Place the code for each of the below steps under their step number in the template code.

Note: There may be IDE errors as we continue, but those will be resolved at the end when the skeleton is complete.

1. Our cloned template already has the `alexaplusunity` node module, but we still need to include it. Open `index.js` under `lambda/custom/` and add the following:

```
var alexaPlusUnityClass = require('alexaplusunity');
var alexaPlusUnity = alexaPlusUnityClass("<YOUR_PUBNUB_PUB_KEY>", "<YOUR_PUBNUB_
↪SUB_KEY>", true); //Third parameter enables verbose logging
```

2. In our example skill, we will use state management to confirm that the user has connected to our game. Insert the code below inside of the `LaunchRequestHandler`:

```
if(attributes.SETUP_STATE == "STARTED") {
    var launchSetUpResult = await launchSetUp(speechText, reprompt, handlerInput, ↪
↪attributes);
    attributes = launchSetUpResult.attributes;
    response = launchSetUpResult.response;
}
```

In the code block above, we are checking to see if we are still in the `SETUP_STATE`. If we are, then run the method `launchSetUp()` and build our response. If we are not in the `SETUP_STATE`, then use the response we already built.

3. In the next steps, we will add `AlexaPlusUnity` to the `FlipSwitchIntent` (`CompletedFlipSwitchIntentHandler`). We will start by creating a payload object for the intent:

```
var payloadObj = {
    type: "State",
    message: state
};
```

4. Then we will send the payload to our game and reply with a success or error if the payload failed to send:


```
var response = await alexaPlusUnity.publishMessage(payloadObj, attributes.PUBNUB_
↪CHANNEL).then((data) => {
    return handlerInput.responseBuilder
        .speak(speechText + reprompt)
        .reprompt(reprompt)
        .getResponse();
}).catch((err) => {
    return ErrorHandler.handle(handlerInput, err);
});
```

- Now, we will add AlexaPlusUnity to the ChangeColorIntent (CompletedChangeColorIntentHandler). We will create our payload object:

```
var payloadObj = {
    type: "Color",
    message: color
};
```

- Send the payload to our game and reply with a success or error if the payload failed to send:

```
var response = await alexaPlusUnity.publishMessage(payloadObj, attributes.PUBNUB_
↪CHANNEL).then((data) => {
    return handlerInput.responseBuilder
        .speak(speechText + reprompt)
        .reprompt(reprompt)
        .getResponse();
}).catch((err) => {
    return ErrorHandler.handle(handlerInput, err);
});
```

- Add AlexaPlusUnity to the GetObjectInDirectionIntent (CompletedGetObjectInDirectionIntentHandler). We will create our payload object:

```
var payloadObj = {
    type: "GetObject",
    message: direction
};
```

- Send a the payload to our game and reply with a success or error if the payload failed to send:

```
var response = await alexaPlusUnity.publishMessageAndListenToResponse(payloadObj, ↪
↪attributes.PUBNUB_CHANNEL, 4000).then((data) => {
    const speechText = 'Currently, ' + data.message.object + ' is ' + direction +
↪' you!';
    const reprompt = ' What\'s next?';
    return handlerInput.responseBuilder
        .speak(speechText + reprompt)
        .reprompt(reprompt)
        .getResponse();
}).catch((err) => {
    return ErrorHandler.handle(handlerInput, err);
});
```

- Create the user's unique channel:

```
var response = await alexaPlusUnity.addChannelToGroup(attributes.PUBNUB_CHANNEL,
↪"AlexaPlusUnityTest").then(async (data) => {
```

(continues on next page)

(continued from previous page)

```
var responseToReturn = responseBuilder
    .speak(speechText)
    .reprompt(reprompt)
    .withSimpleCard('Alexa Plus Unity', "Here is your Player ID: " + attributes.
        ↪PUBNUB_CHANNEL)
    .getResponse();

var userId = handlerInput.requestEnvelope.session.user.userId;
return await sendUserId(userId, attributes, handlerInput, responseToReturn);
}).catch((err) => {
    return ErrorHandler.handle(handlerInput, err);
});
```

10. Now, we need to send the game the user's Alexa ID so we can access their persistent session attributes. Create our payload object:

```
var payloadObj = {
    type: "AlexaUserId",
    message: userId
};
```

11. Send the payload to the game:

```
return await alexaPlusUnity.publishMessage(payloadObj, attributes.PUBNUB_CHANNEL).
    ↪then((data) => {
        return response;
    }).catch((err) => {
        return ErrorHandler.handle(handlerInput, err);
    });
```

12. Lastly, we need to initialize the skills attributes

```
attributes.SETUP_STATE = "STARTED";
var newChannel = await alexaPlusUnity.uniqueQueueGenerator("AlexaPlusUnityTest");

if(newChannel != null) {
    attributes.PUBNUB_CHANNEL = newChannel;
} else {
    return null;
}
```

7.5 Deploying the Skill

1. Open a command prompt or terminal and navigate to <Template Location>
2. Type `ask deploy` to deploy the skill.

Note: This will only work if you set up the [ASK CLI](#) correctly!

7.6 Wrapping Up

At this point, you should be able to test the skill by saying, “Alexa, open unity light”.

We have finished the Alexa Skill!

Congratulations! You have completed the Alexa Plus Unity Light Control Demo!

8.1 Testing the Light Control Demo

1. Open the skill by saying “Alexa, open unity light”.
2. Make note of your player ID.
3. Make sure your Unity project is open.
4. Click on the **Amazon Alexa** GameObject.
5. Fill the `Channel` field in the Light Control script with the player ID returned by the Alexa Skill.
6. Press the **Play Button** in Unity to enter **Play Mode**.
7. Once in **Play Mode**, re-launch your Alexa Skill.

You should notice that your skill’s welcome message changed. This means that the game is successfully communicating to your Alexa Skill. You can now say commands like “Change light to blue”, or “What is in front of me?” to interact with the game with your voice.

9.1 Classes

AlexaBaseData

AmazonAlexaManager

ConnectionStatusEventData

ErrorEventData

GetSessionAttributesEventData

HandleMessageEventData

MessageSentEventData

SetSessionAttributesEventData

9.2 AlexaBaseData

Syntax

```
public abstract class AlexaBaseData : BaseEventData
```

9.2.1 Constructors

AlexaBaseData(EventSystem)

Declaration

```
public AlexaBaseData(EventSystem eventSystem)
```

Parameters

Type	Name	Description
EventSystem	eventSystem	

9.2.2 Properties

Exception

Declaration

```
public Exception Exception { get; }
```

Property Value

Type	Description
System.Exception	

IsError

Declaration

```
public bool IsError { get; }
```

Property Value

Type	Description
System.Boolean	

9.2.3 Methods

BaseInitialize(Boolean, Exception)

Declaration

```
protected virtual void BaseInitialize(bool isError, Exception exception)
```

Parameters

Type	Name	Description
System.Boolean	isError	
System.Exception	exception	

9.3 AmazonAlexaManager

Syntax

```
public class AmazonAlexaManager
```


9.3.1 Constructors

AmazonAlexaManager(String, String, String, String, String, String, GameObject, Action<HandleMessageEventData>, Action<ConnectionStatusEventData>, Boolean)

AmazonAlexaManager Constructor.

Declaration

```
protected virtual void BaseInitialize(bool isError, Exception exception)
```

Parameters

Type	Name	Description
System.String	publishKey	Your PubNub publish key.
System.String	subscribeKey	Your PubNub subscribe key.
System.String	channel	Your player's channel. (Should be unique to the player)
System.String	tableName	Name of your skill's DynamoDB table where the persistent attributes are stored.
System.String	identityPoolId	Identifier of your AWS Cognito identity pool.
System.String	AWSRegion	The AWS Region where your DynamoDB table and Cognito identity pool are hosted.
GameObject	gameObject	The GameObject you are attaching this manager instance to.
System.Action<HandleMessageEventData>	handleMessageCallback	The callback for when a message is recieved from your Alexa Skill.
System.Action<ConnectionStatusEventData>	handleConnectionStatusCallback	
System.Boolean	debug	(Optional) True to debug.

9.3.2 Fields

handleConnectionStatusCallback

The connection status recieved callback.

Declaration

```
public Action<ConnectionStatusEventData> handleConnectionStatusCallback
```

Field Value

Type	Description
System.Action<ConnectionStatusEventData>	

handleMessageCallback

The message recieved callback.

Declaration

```
public Action<HandleMessageEventData> handleMessageCallback
```

Field Value

Type	Description
System.Action<HandleMessageEventData>	

9.3.3 Properties

alexaUserDynamoKey

Gets or Resets the player's DynanoDB table key.

Declaration

```
public string alexaUserDynamoKey { get; set; }
```

Property Value

Type	Description
System.String	The alexa user dynamo key.

channel

Resets your player's channel. (Should be unique to the player)

Declaration

```
public string channel { set; }
```

Property Value

Type	Description
System.String	The channel.

9.3.4 Methods

GetSessionAttributes(Action<GetSessionAttributesEventData>)

Gets the Skill's persistant session attributes from DynamoDB.

Declaration

```
public void GetSessionAttributes (Action<GetSessionAttributesEventData> callback)
```

Parameters

Type	Name	Description
System.Action<GetSessionAttributesEventData>	callback	The callback.

SendToAlexaSkill(Object, Action<MessageSentEventData>)

Sends a message to Alexa Skill. NOTE: Skill will only receive the message if it is listening for a response.

Declaration

```
public void SendToAlexaSkill(object message, Action<MessageSentEventData> callback)
```

Parameters

Type	Name	Description
System.Object	message	The message.
System.Action<MessageSentEventData>	callback	The callback.

SetSessionAttributes(Dictionary<String, AttributeValue>, Action<SetSessionAttributesEventData>)

Sets the Skill's persistent session attributes in DynamoDB.

Declaration

```
public void SetSessionAttributes(Dictionary<string, AttributeValue> attributes, Action<SetSessionAttributesEventData> callback)
```

Parameters

Type	Name	Description
System.Collections.Generic.Dictionary<System.String, AttributeValue>	attributes	The attributes to set.
System.Action<SetSessionAttributesEventData>	callback	The callback.

9.4 ConnectionStatusEventData

Syntax

```
public class ConnectionStatusEventData : AlexaBaseData
```

9.4.1 Constructors

ConnectionStatusEventData(EventSystem)

Declaration

```
public ConnectionStatusEventData(EventSystem eventSystem)
```

Parameters

Type	Name	Description
EventSystem	eventSystem	

9.4.2 Properties

Category

Declaration

```
public PNStatusCategory Category { get; }
```

Property Value

Type	Description
PNStatusCategory	

9.4.3 Methods

Initialize(Boolean, PNStatusCategory, Exception)

Declaration

```
public void Initialize(bool isError, PNStatusCategory category, Exception exception =  
↪ null)
```

Parameters

Type	Name	Description
System.Boolean	isError	
PNStatusCategory	category	
System.Exception	exception	

9.5 ErrorEventData

Syntax

```
public class ErrorEventData : AlexaBaseData
```

9.5.1 Constructors

ErrorEventData(EventSystem)

Declaration

```
public ErrorEventData(EventSystem eventSystem)
```

Parameters

Type	Name	Description
EventSystem	eventSystem	

9.5.2 Methods

Initialize(Boolean, PNStatusCategory, Exception)

Declaration

```
public void Initialize(Exception exception, bool isError = true)
```

Parameters

Type	Name	Description
System.Exception	exception	
System.Boolean	isError	

9.6 GetSessionAttributesEventData

Syntax

```
public class GetSessionAttributesEventData : AlexaBaseData
```

9.6.1 Constructors

GetSessionAttributesEventData(EventSystem)

Declaration

```
public GetSessionAttributesEventData(EventSystem eventSystem)
```

Parameters

Type	Name	Description
EventSystem	eventSystem	

9.6.2 Properties

Values

Declaration

```
public Dictionary<string, AttributeValue> Values { get; }
```

Property Value

Type	Description
System.Collections.Generic.Dictionary<System.String, AttributeValue>	

9.6.3 Methods

Initialize(Boolean, Dictionary<String, AttributeValue>, Exception)

Declaration

```
public void Initialize(bool isError, Dictionary<string, AttributeValue> values,   
↳Exception exception = null)
```

Parameters

Type	Name	Description
System.Exception	exception	
System.Collections.Generic.Dictionary<System.String, AttributeValue>	Dictionary	
System.Boolean	isError	

9.7 HandleMessageEventData

Syntax

```
public class HandleMessageEventData : AlexaBaseData
```

9.7.1 Constructors

HandleMessageEventData(EventSystem)

Declaration

```
public HandleMessageEventData(EventSystem eventSystem)
```

Parameters

Type	Name	Description
EventSystem	eventSystem	

9.7.2 Properties

Message

Declaration

```
public Dictionary<string, object> Message { get; }
```

Property Value

Type	Description
System.Collections.Generic.Dictionary<System.String, System.Object>	

9.7.3 Methods

Initialize(Boolean, Dictionary<String, Object>, Exception)

Declaration

```
public void Initialize(bool isError, Dictionary<string, object> message, Exception_
↳exception = null)
```

Parameters

Type	Name	Description
System.Boolean	isError	
System.Collections.Generic.Dictionary<System.String, System.Object>	message	
System.Exception	exception	

9.8 MessageSentEventData

Syntax

```
public class MessageSentEventData : AlexaBaseData
```

9.8.1 Constructors

MessageSentEventData(EventSystem)

Declaration

```
public MessageSentEventData(EventSystem eventSystem)
```

Parameters

Type	Name	Description
EventSystem	eventSystem	

9.8.2 Properties

Message

Declaration

```
public object Message { get; }
```

Property Value

Type	Description
System.Object	

9.8.3 Methods

Initialize(Boolean, Object, Exception)

Declaration

```
public void Initialize(bool isError, object message, Exception exception = null)
```

Parameters

Type	Name	Description
System.Boolean	isError	
System.Object	message	
System.Exception	exception	

9.9 SetSessionAttributesEventData

Syntax

```
public class SetSessionAttributesEventData : AlexaBaseData
```

9.9.1 Constructors

SetSessionAttributesEventData(EventSystem)

Declaration

```
public SetSessionAttributesEventData(EventSystem eventSystem)
```

Parameters

Type	Name	Description
EventSystem	eventSystem	

9.9.2 Methods

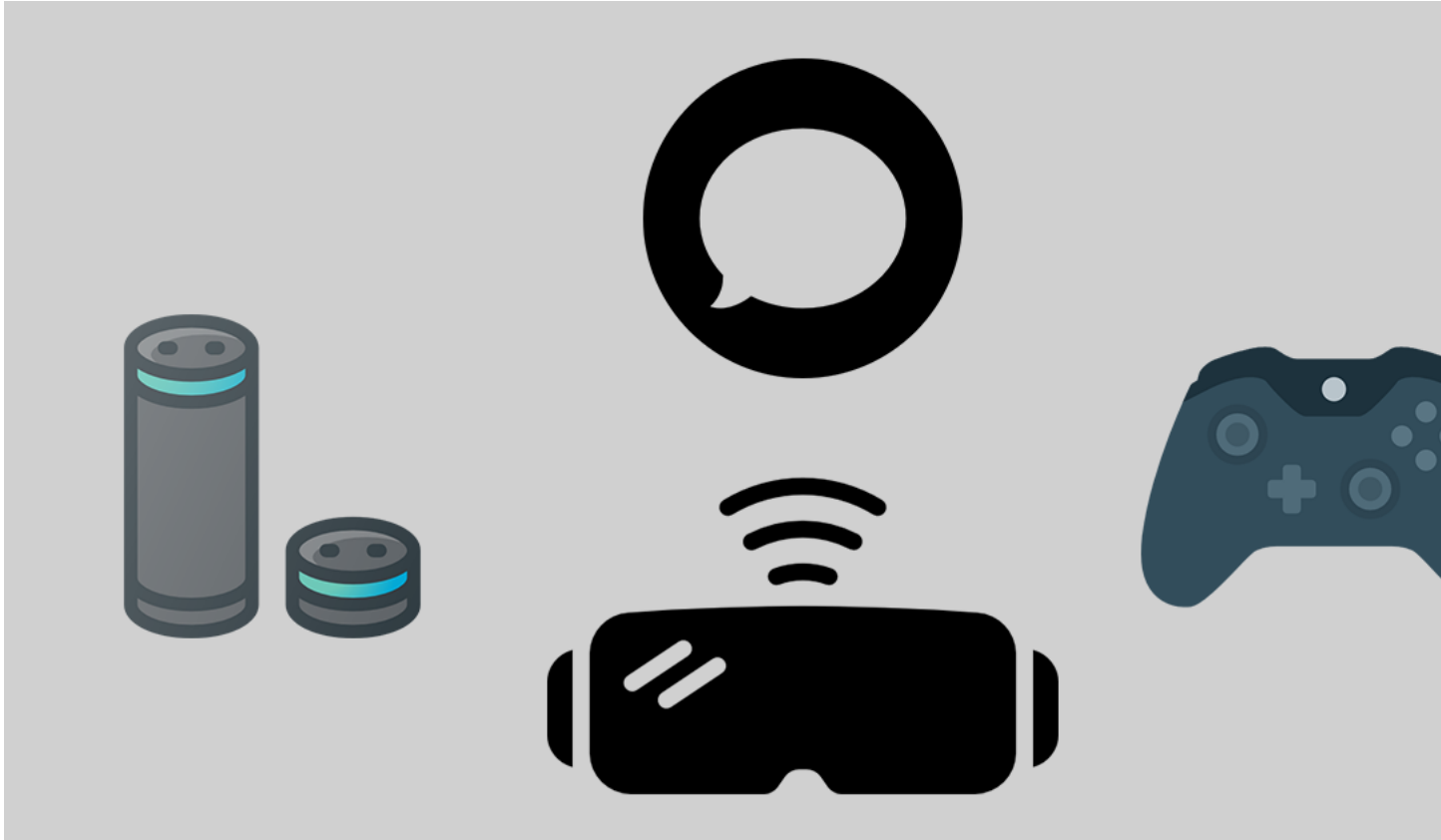
Initialize(Boolean, Exception)

Declaration

```
public void Initialize(bool isError, Exception exception = null)
```

Parameters

Type	Name	Description
System.Boolean	isError	
System.Exception	exception	



CHAPTER 10

Alexa Plus Unity

The Alexa Plus Unity SDK was built to help with the expansion of Amazon's Alexa into video games, bringing a whole new layer of immersion into gaming.

Ready to get started? Follow the guides in the navigation bar to learn how to use the platform!

10.1 The Explorer Demo