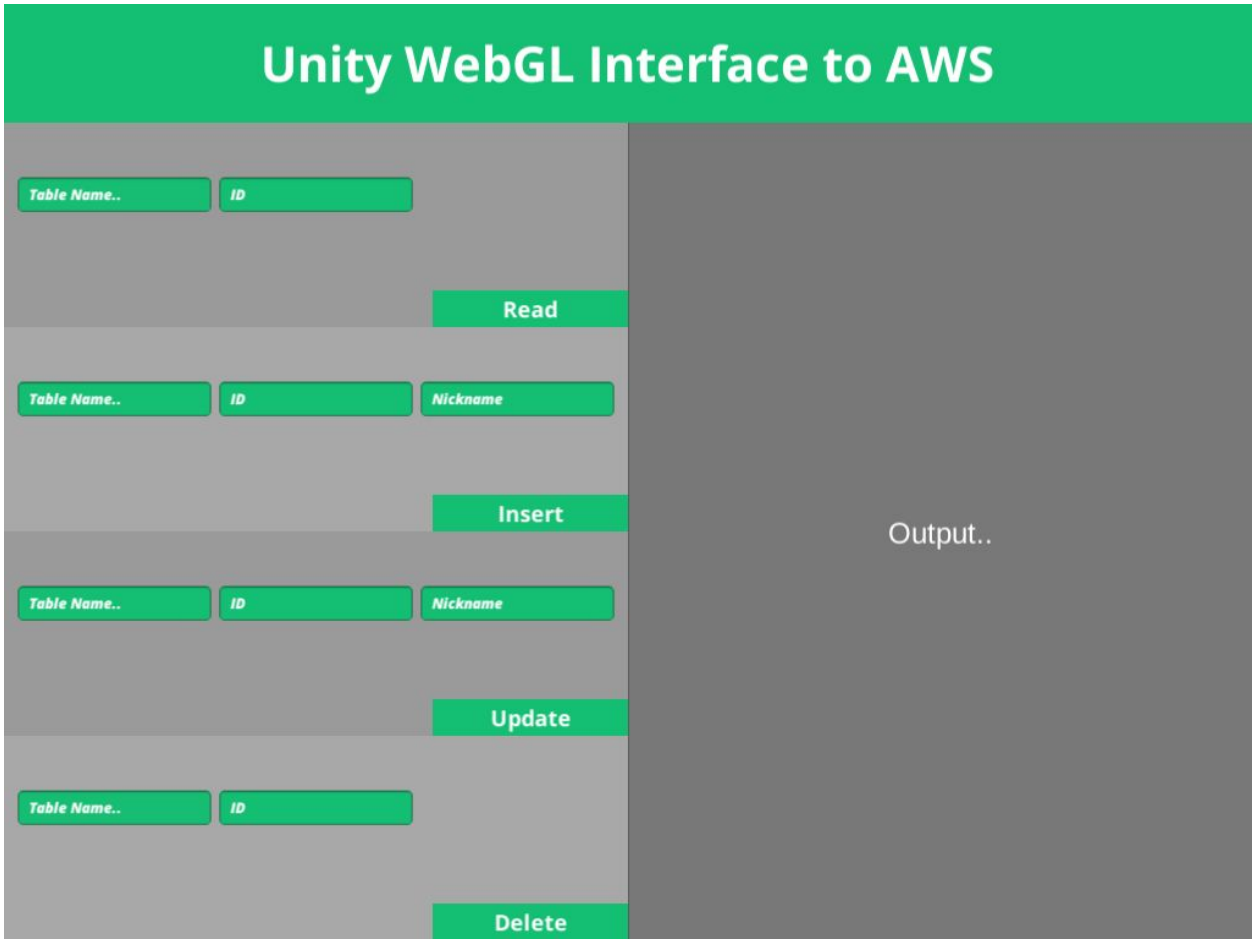
# AWS, WebGL and Unity

This document present an approach to use AWS's DynamoDB with an Unity WebGL application. At the moment Amazon doesn't offer an SDK that supports Unity's WebGL,  the only supported platforms are Android and iOS.

The methodology proposed needs to me further evaluated for bugs and performance.

## Results

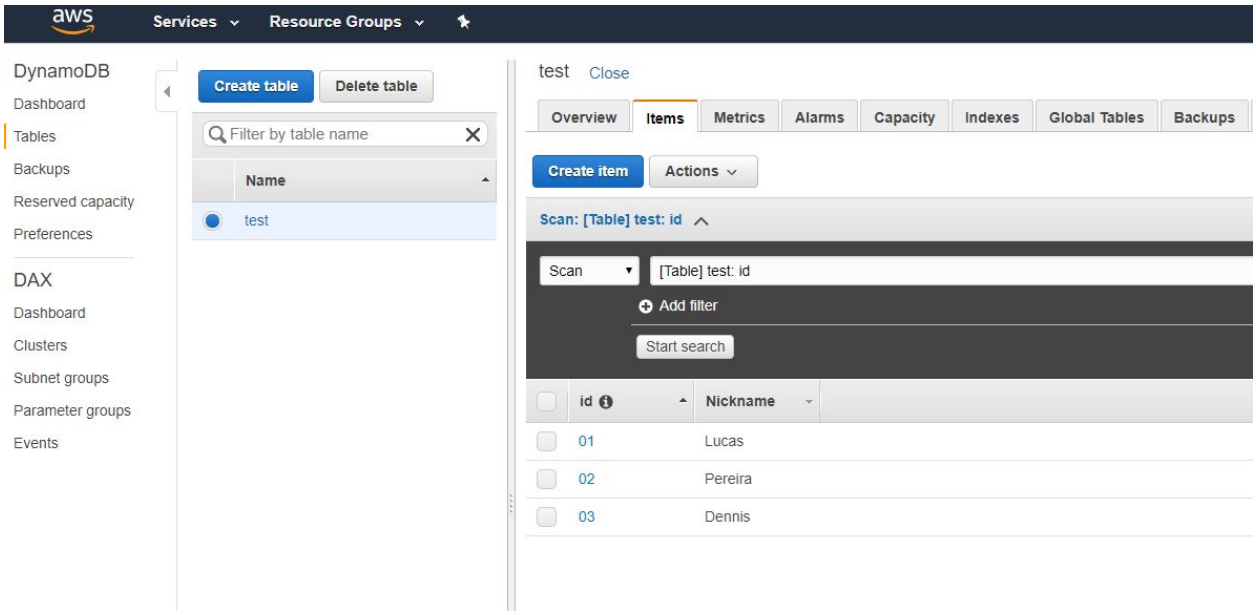
This guide will result in an Unity WebGL application that can: **read, update**, **insert** and **delete** data into an remote DynamoDB table.



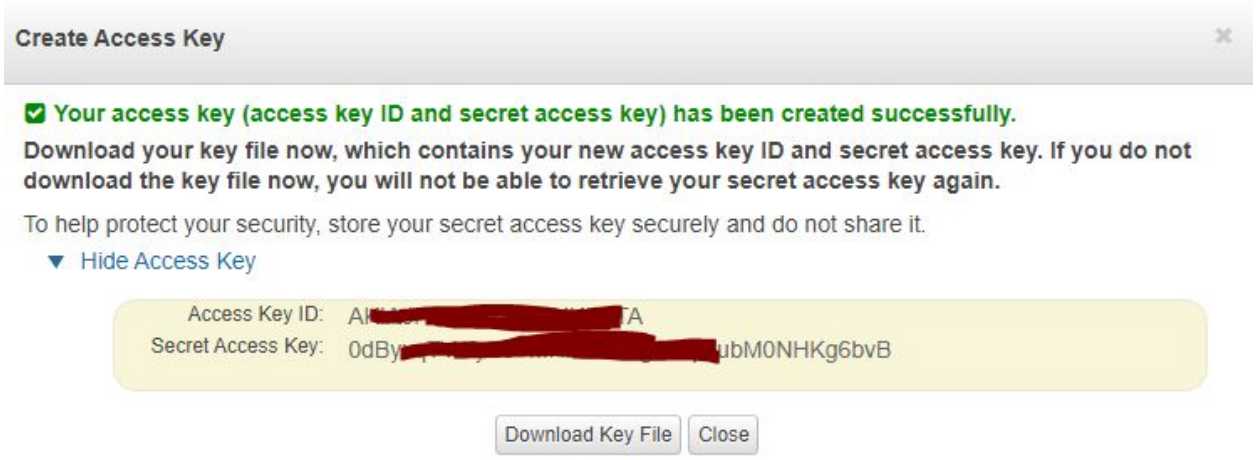
# AWS Database Creation

This guide requires a DynamoDB database set up. It's possible to create one at https://aws.amazon.com/dynamodb/.

Once the database set up, create the table where all operations will take place. Name the table 'test', add two columns 'id' and 'Nickname'. For testing purposes there are three items added to the table, it looks like the image above on the console screen.

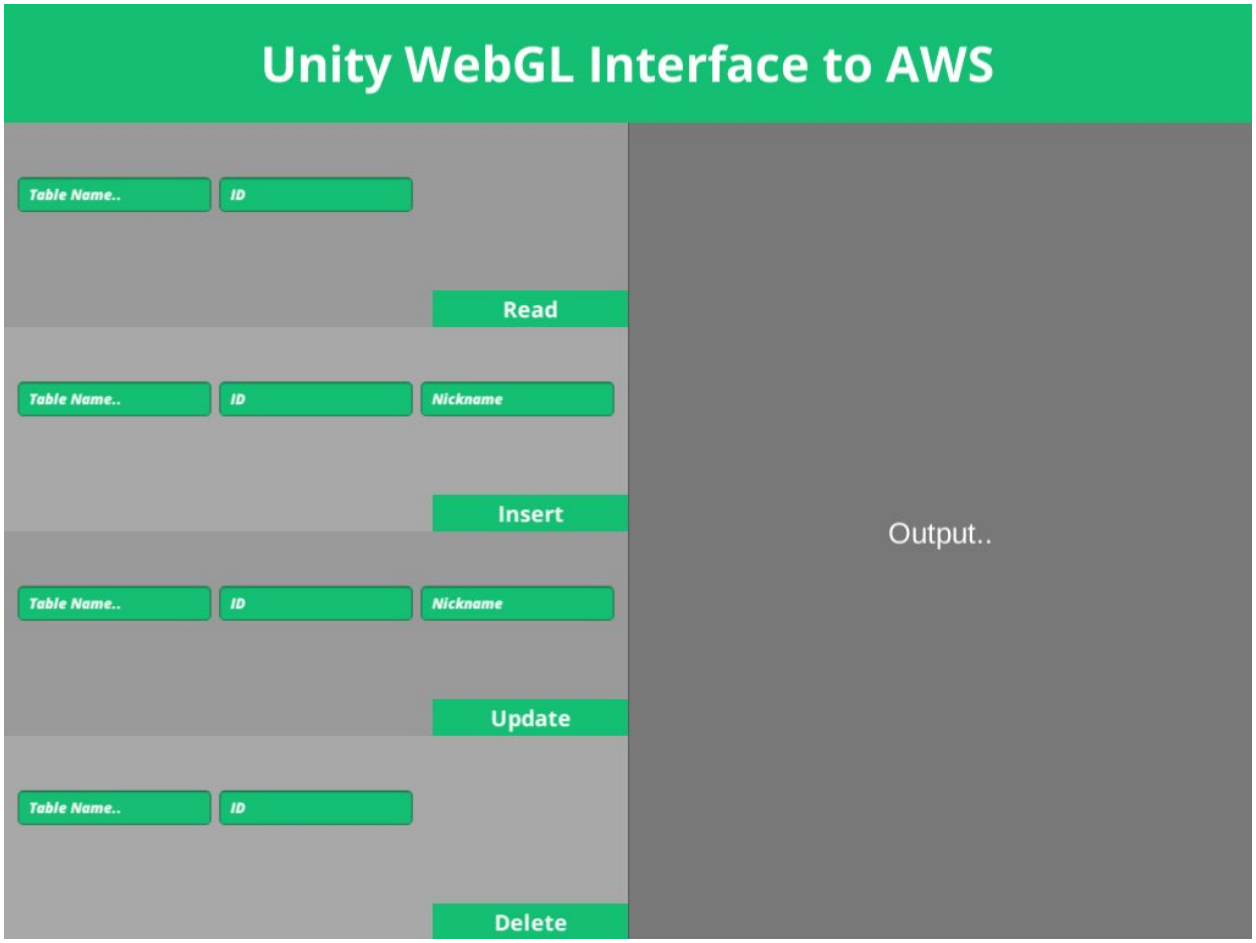



After creating the table set up security credentials to be able to access the database. It's possible to do it on the IAM page under Services on the Amazon Console (use the search box). At the end of the process it will have generated 2 keys, like the image below.



# WebGL Application

Once the database is set up, create a minimal Unity WebGL application. I will use Unity 2018.2.4f1 for this. Start by creating a simple UI like the one below. If you are struggling to create it, you can get it from the attached project.




## Calling Javascript from C#

In order to communicate with DynamoDB we will use AWS's Javascript SDK for browsers, making calls to Javascript functions from the C# code.

It's fairly simple to do so, we define the headers of the Javascript functions we will create on a C# script on the format below.

```
    [DllImport("__Internal")]
    private static extern void ReadData(string tableName, string itemId);

    [DllImport("__Internal")]
    private static extern void InsertData(string tableName, string itemId,
string nickname);

    [DllImport("__Internal")]
    private static extern void UpdateData(string tableName, string itemId,
string nickname);

    [DllImport("__Internal")]
    private static extern void DeleteData(string tableName, string itemId);
```

We can call those functions normally throughout the C# script they were defined. On this guide each UI button call one of the Javascript functions. Therefore we implement each button click like the example below.

```
    /// <summary>
    /// Read data button click..
    /// </summary>
    public void ReadButtonClick()
    {
        ReadData(ReadTable.text, ReadID.text);
    }

    /// <summary>
    /// Insert data button click..
    /// </summary>
    public void InsertButtonClick()
    {
        InsertData(InsertTable.text, InsertID.text, InsertNickname.text);
    }

    /// <summary>
    /// Update data button click..
    /// </summary>
    public void UpdateButtonClick()
    {
        UpdateData(UpdateTable.text, UpdateID.text, UpdateNickname.text);
```

```
    }

    /// <summary>
    /// Delete data button click..
    /// </summary>
    public void DeleteButtonClick()
    {
        DeleteData(DeleteTable.text, DeleteID.text);
    }
```

Finally each Javascript function needs to be programmed and saved on a file with the extension**.jslib**, that functions must be enclosed on the **mergeInto** function:

```
mergeInto(LibraryManager.library,
{
    ReadData: function (tableName, itemId) { code goes here }
    InsertData: function (tableName, itemId, nickname) {code goes here }
    DeleteData: function (tableName, itemId) { code goes here }
    UpdateData: function (tableName, itemId, nickname) { code goes here }
});
```

This file must be added to a folder named **Plugins** under the **Assets** folder.

# Javascript Code

Now we need to create each Javascript function that we will use, those are: **UpdateData**, **DeleteData**, **ReadData** and **InputData**. Since the functions are very similar, we will analyse only the ReadData Javascript function. The other functions code is at the end of the document.

```
// ##### Function to Read Data from the DynamoDB #####
    ReadData: function (tableName, itemId)
    {
        var awsConfig =
        {
                region: "us-east-1",
                endpoint: "http://dynamodb.us-east-1.amazonaws.com",
                accessKeyId: "AKIAJAENBHYTHITUKR2TA",
                secretAccessKey: "0dByxqT8Ka90+MHznAsWgeuHdep9scM0NHKg6bvB"
        };

        AWS.config.update(awsConfig);
        var returnStr = "Error";
        var docClient = new AWS.DynamoDB.DocumentClient();
        var params =
        {
            TableName: Pointer_stringify(tableName),
            Key:{ "id": Pointer_stringify(itemId) }
        };

        docClient.get(params, function (err, data)
        {
            if (err)
            {
                returnStr = "Error:" + JSON.stringify(err, undefined, 2);
                SendMessage('DynamoInterface', 'StringCallback',
returnStr);}
            else
            {
                 returnStr = "Data Found:" + JSON.stringify(data,
undefined, 2);
                 SendMessage('DynamoInterface', 'StringCallback',
returnStr);}
        });
    }
```

Initially we update our AWS settings to be able to connect to the database. We define the database's **accessKeyId**, **secretAccessKey**, **region** and **endpoint** (remember the keys where created on the IAM page on the AWS console). We set the keys by calling the **AWS.config.update** function.

```
var awsConfig =
{
        region: "us-east-1",
        endpoint: "http://dynamodb.us-east-1.amazonaws.com",
        accessKeyId: "AKIAJAENBHYTHITUKR2TA",
        secretAccessKey: "0dByxqT8Ka90+MHznAsWgeuHdep9scM0NHKg6bvB"
};

AWS.config.update(awsConfig);
```

Now we can create a new *DocumentClient* and get the data by calling the **get** function.

```
var docClient = new AWS.DynamoDB.DocumentClient();
var params =
{
    TableName: Pointer_stringify(tableName),
    Key:{ "id": Pointer_stringify(itemId) }
};

docClient.get(params, function (err, data)
{ … }
```

Note that the before calling **get** the **params** variable is set with the table name and primary key, those values come from the C# function.

The **get** function receives an object with the parameters (**params**) and a callback function to execute upon completion of the task.

The callback function is extremely useful because the **get** function is asynchronous, that means the rest of the code don't wait for it to finish executing. Once the **get** function executes it calls the callback function that notifies the C# script.

```
docClient.get(params, function (err, data)
{
   if (err)
   {
       returnStr = "Error:" + JSON.stringify(err, undefined, 2);
       SendMessage('DynamoInterface', 'StringCallback', returnStr);
   }
    else
    {
        returnStr = "Data Found:" + JSON.stringify(data, undefined, 2);
        SendMessage('DynamoInterface', 'StringCallback', returnStr);
     }
  });
```

## Calling C# from Javascript

In order to notify the C# script we use the *SendMessage* function, passing as parameters:

- The GameObject name that contains the C# script function we want to execute (‘DynamoInterface’ in my case).
- The function name we want to call (‘StringCallback’ in my case).
- An object we want to send.

# Final Build and Details

Once all Javascript functions are created and the UI is linked with the C# script. We can go ahead and build the application. No error should appear in the Unity console.

Once the application created, navigate to it's folder and open the **index.html** file with a text editor. Add the following line on the **<head>** tag:

```
<script src="https://sdk.amazonaws.com/js/aws-sdk-2.7.16.min.js"></script>
```

This is a reference to the AWS browser SDK, and it's necessary for your Javascript to run. The final html file should look like this.

```
<!DOCTYPE html>
<html lang="en-us">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Unity WebGL Player | VIS_WebApp</title>
    <link rel="shortcut icon" href="TemplateData/favicon.ico">
    <link rel="stylesheet" href="TemplateData/style.css">
    <script src="https://sdk.amazonaws.com/js/aws-sdk-2.7.16.min.js"></script>
    <script src="TemplateData/UnityProgress.js"></script>
    <script src="Build/UnityLoader.js"></script>
    <script>
      var gameInstance = UnityLoader.instantiate("gameContainer", "Build/Builds.json", {onProgress: UnityProgress});
    </script>
  </head>
  <body>
    <div class="webgl-content">
      <div id="gameContainer" style="width: 1024px; height: 768px"></div>
      <div class="footer">
        <div class="webgl-logo"></div>
        <div class="fullscreen" onclick="gameInstance.SetFullscreen(1)"></div>
        <div class="title">VIS_WebApp</div>
      </div>
    </div>
  </body>
</html>
```

# Appendix - The code

## C# Class

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.Runtime.InteropServices;
using UnityEngine.UI;

public class DynamoInterface : MonoBehaviour
{
    [DllImport("__Internal")]
    private static extern void ReadData(string tableName, string itemId);

    [DllImport("__Internal")]
    private static extern void InsertData(string tableName, string itemId,
string nickname);

    [DllImport("__Internal")]
    private static extern void UpdateData(string tableName, string itemId,
string nickname);

    [DllImport("__Internal")]
    private static extern void DeleteData(string tableName, string itemId);

    public Text Output;

    [Header("Read Fields")]
    public InputField ReadTable;
    public InputField ReadID;

    [Space(10)]
    [Header("Insert Fields")]
    public InputField InsertTable;
    public InputField InsertID;
    public InputField InsertNickname;

    [Space(10)]
    [Header("Update Fields")]
    public InputField UpdateTable;
```

```
    public InputField UpdateID;
    public InputField UpdateNickname;

    [Space(10)]
    [Header("Delete Fields")]
    public InputField DeleteTable;
    public InputField DeleteID;

    /// <summary>
    /// Read data button click..
    /// </summary>
    public void ReadButtonClick()
    {
        ReadData(ReadTable.text, ReadID.text);
    }

    /// <summary>
    /// Insert data button click..
    /// </summary>
    public void InsertButtonClick()
    {
        InsertData(InsertTable.text, InsertID.text, InsertNickname.text);
    }

    /// <summary>
    /// Update data button click..
    /// </summary>
    public void UpdateButtonClick()
    {
        UpdateData(UpdateTable.text, UpdateID.text, UpdateNickname.text);
    }

    /// <summary>
    /// Delete data button click..
    /// </summary>
    public void DeleteButtonClick()
    {
        DeleteData(DeleteTable.text, DeleteID.text);
    }

    #region Callbacks
```

```
    public void StringCallback(string info)
    {
        Output.text = info;
    }

    #endregion
}
```

## Javascript File

```
mergeInto(LibraryManager.library,
{

    // ##### Function to Read Data from the DynamoDB #####
    ReadData: function (tableName, itemId)
    {
        var params =
        {
            TableName: Pointer_stringify(tableName),
            Key:
            {
                "id": Pointer_stringify(itemId)
            }
        };

        var awsConfig =
        {
                region: "us-east-1",
                endpoint: "http://dynamodb.us-east-1.amazonaws.com",
                accessKeyId: "AKIAJFENBSGPATUKR2TA",
                secretAccessKey: "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"
        };

        AWS.config.update(awsConfig);

        var docClient = new AWS.DynamoDB.DocumentClient();

        var returnStr = "Error";
```

```
        docClient.get(params, function (err, data)
        {
            if (err)
            {
                returnStr = "Error:" + JSON.stringify(err, undefined, 2);
                SendMessage('DynamoInterface', 'StringCallback',
returnStr);
            }
            else
            {
                 returnStr = "Data Found:" + JSON.stringify(data,
undefined, 2);
                 SendMessage('DynamoInterface', 'StringCallback',
returnStr);
            }
        });
    },



    // ##### Function to Insert Data on the DynamoDB #####
    InsertData: function (tableName, itemId, nickname)
    {
        var params =
        {
            TableName: Pointer_stringify(tableName),
            Item:
            {
                "id": Pointer_stringify(itemId),
                "Nickname": Pointer_stringify(nickname)
            }
        };

        var awsConfig =
        {
                region: "us-east-1",
                endpoint: "http://dynamodb.us-east-1.amazonaws.com",
                accessKeyId: "AKIAJFENBSGPATUKR2TA",
                secretAccessKey: "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"
        };
```

```
        AWS.config.update(awsConfig);

        var docClient = new AWS.DynamoDB.DocumentClient();

        var returnStr = "Error";

        docClient.put(params, function(err, data)
        {
            if (err)
            {
                returnStr = "Error: " + JSON.stringify(err, undefined, 2);
                SendMessage('DynamoInterface', 'StringCallback',
returnStr);
            }
            else
            {
                 returnStr = "Inserted: " + JSON.stringify(data, undefined,
2);
                 SendMessage('DynamoInterface', 'StringCallback',
returnStr);
            }
        });
    },

     // ##### Function to Delete Data from the DynamoDB #####
    DeleteData: function (tableName, itemId)
    {
        var params =
        {
            TableName: Pointer_stringify(tableName),
            Key:
            {
                "id": Pointer_stringify(itemId)
            }
        };

        var awsConfig =
        {
                region: "us-east-1",
                endpoint: "http://dynamodb.us-east-1.amazonaws.com",
```

```
                accessKeyId: "AKIAJFENBSGPATUKR2TA",
                secretAccessKey: "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"
        };

        AWS.config.update(awsConfig);

        var docClient = new AWS.DynamoDB.DocumentClient();

        var returnStr = "Error";

        docClient.delete(params, function(err, data)
        {
            if (err)
            {
                returnStr = "Error: " + JSON.stringify(err, undefined, 2);
                SendMessage('DynamoInterface', 'StringCallback',
returnStr);
            }
            else
            {
                 returnStr = "Deleted: " + JSON.stringify(data, undefined,
2);
                 SendMessage('DynamoInterface', 'StringCallback',
returnStr);
            }
        });
    },


    // ##### Function to Update Data on the DynamoDB #####
    UpdateData: function (tableName, itemId, nickname)
    {
        var params =
        {
            TableName: Pointer_stringify(tableName),
            Key:
            {
                "id": Pointer_stringify(itemId),
            },
            UpdateExpression: "set Nickname = :newName",
            ExpressionAttributeValues:
```

```
            {
                ":newName" : Pointer_stringify(nickname)
            },
             ReturnValues: "UPDATED_NEW"
        };

        var awsConfig =
        {
                region: "us-east-1",
                endpoint: "http://dynamodb.us-east-1.amazonaws.com",
                accessKeyId: "AKIAJFENBSGPATUKR2TA",
                secretAccessKey: "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"
        };

        AWS.config.update(awsConfig);

        var docClient = new AWS.DynamoDB.DocumentClient();

        var returnStr = "Error";

        docClient.update(params, function(err, data)
        {
            if (err)
            {
                returnStr = "Error: " + JSON.stringify(err, undefined, 2);
                SendMessage('DynamoInterface', 'StringCallback',
returnStr);
            }
            else
            {
                 returnStr = "Updated: " + JSON.stringify(data, undefined,
2);
                 SendMessage('DynamoInterface', 'StringCallback',
returnStr);
            }
        });
    },


});
```