

PlayFab のクラウド スクリプト

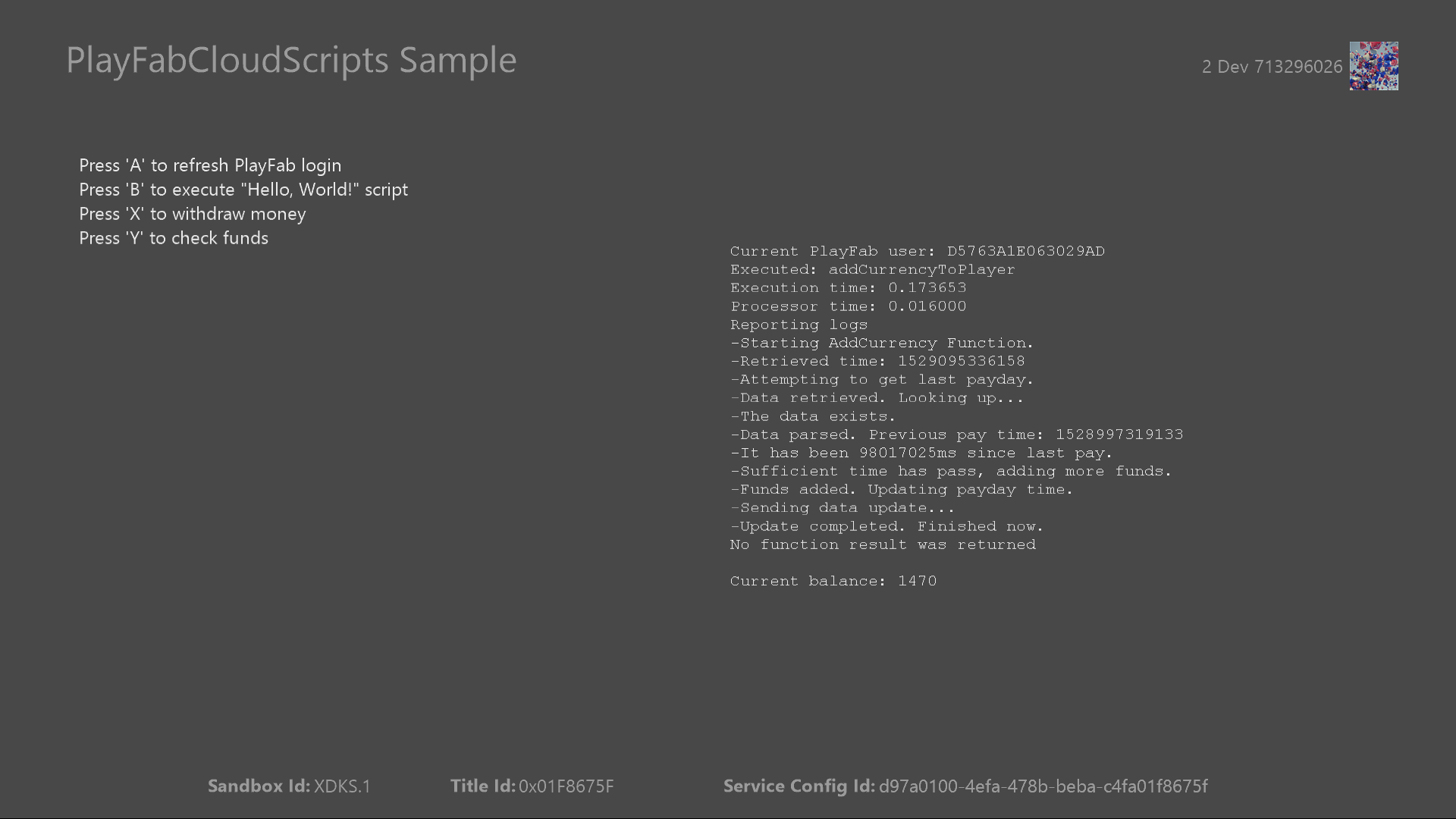
*\* このサンプルは、2018 年 4 月の XDK と互換性があります*

# 概要

このサンプルでは、クラウド スクリプト呼び出しのデモを行っています。

# サンプルの使用

このサンプルは、XDKS.1 サンドボックスで機能するように構成されています。



|  |  |
| --- | --- |
| 作用 | ゲームパッド |
| PlayFab 認証トークンを更新する | Aボタン |
| 「Hello、World」クラウド スクリプトを実行する | B ボタン |
| プレーヤーの資金を追加するためのクラウド スクリプトを実行する | X ボタン |
| プレーヤ-架空の通貨を照会する | Y ボタン |
| 口座開設ピッカー | メニューボタン |
| 終了 | 表示ボタン |

# 実装上の注意

このバージョンの XDK では、このサンプルでは Xbox One 本体に XDKS.1 **SandboxID**を使用する必要があります。この **SandboxID**に切り替えるには Xbox One XDK コマンドプロンプトで、次のように入力します。

xbconfig sandboxid=XDKS.1

PlayFab のタイトル ID は、サンプル初期化コードで設定する必要があります。このサンプルでは、 **PlayFabSettings::titleId** 静的フィールドを割り当てた、PlayFabのタイトルID「DC0」を使用しています。

# クラウド スクリプト

PlayFab サービスでは、PlayFab ダッシュボードから JavaScript を使用して独自のカスタム スクリプトを作成できます。これらのスクリプトは、**ExecuteCloudScript** API を使用するか、ダッシュボードから設定されたトリガーを介して、ゲーム クライアントから呼び出すことができます。

これらのスクリプトは、プレーヤー統計の更新や架空の通貨の付与など、それぞれのまたは少数グループの API 呼び出しを安全に実行できるようにします。クラウド内での、長いもしくは連続したスクリプトの実行は意図していません。実行できるクラウド スクリプトには、実行時間、スクリプトの合計サイズ、単一のスクリプトから実行できる API 呼び出しの数などの制限があります。これらの制限はご自身の購読レベルに基づいており、ダッシュボードの [**設定**] タブの [**制限**] セクションにあります。

このサンプルは、ネットワーク アクティビティを直接プレイヤーの入力に結び付けます。プレイヤーがリクエストを頻繁に出すのを防ぐためのコードガードが含まれています。これは特定の XRには必要ありませんが、必要以上にネットワークが飽和状態にならないようにすることをお勧めします。

# 「Hello、World」クラウド スクリプト

PlayFab は参照として使用するためのスクリプトをいくつか提供します。PlayFab ダッシュボードには次のように表示されます。

// これはクラウド スクリプト関数です。「args」 は 「FunctionParameter」 の値に設定されます。

// ExecuteCloudScript API のパラメータです。

// (https://api.playfab.com/Documentation/Client/method/ExecuteCloudScript)

// 「context」 は、クラウド スクリプト関数が PlayStream アクションから呼び出されたときの追加情報を含みます。

handlers.helloWorld = function（args、context）{

// 定義済みの 「currentPlayerId」 変数はゲーム クライアントにログインしているプレイヤーの PlayFab ID で初期化されます。

// クラウド スクリプトはプレーヤーの認証を自動的に処理します。

var message = "Hello " + currentPlayerId + "!";

// 「log」オブジェクトを使用してデバッグ ステートメントを書き出すことができます。そこには

// ログ レベルに対応する　3　つの関数、　debug、info および error があります。これらの機能で

//　メッセージ文字列とオプションのオブジェクトを受け取ります。

log.info （メッセージ）;

var inputValue = null;

if (args && args.inputValue)

inputValue = args.inputValue;

log.debug("helloWorld:", { input: args.inputValue });

// クラウド スクリプト関数から返された値は

// ExecuteCloudScript API レスポンスで log ステートメントやその他の診断情報である

// API 呼び出しや外部 HTTP 要求によって返されたエラーなどと共にゲームクライアントに

// 返されます。これらは、関数の実行によって生成されるオプションの player\_executed\_cloudscript PlayStream イベントにも

// 含まれています。

// (https://api.playfab.com/playstream/docs/PlayStreamEventModels/player/player\_executed\_cloudscript)

{messageValue：message}　を返します。

};

これは単にゲーム クライアントとクラウド スクリプトの間でデータの受け渡しを示しています。このスクリプトにはとても多くのコメントがあり、静的ログ オブジェクトと同様に関数に引数を渡すことを説明します。スクリプトの完了まで実行すると、**ExecuteCloudScriptResult** オブジェクトは、ログの内容、関数の戻り値およびスクリプト自体でエラーが発生した場合に、エラー情報を受け取ります。

# クラウド スクリプトで API を呼び出す

統計の更新やトランザクションの処理などのアクションを実行するためにクライアントに大きく依存していると、ゲームが悪意のある攻撃にさらされやすい領域が増えます。このサンプルは、**addCurrencyToPlayer** クラウド スクリプトでこれらの操作を実行することで、攻撃の可能性を減らす方法を示します。このスクリプトの本文は次のとおりです。

handlers.addCurrencyToPlayer = function (args, context) {

log.debug("Starting script to add currency.");

var timeNow = Date.now();

log.debug("Current server time: " + timeNow);

var getRequest = {

PlayFabId: currentPlayerId, Keys: [ "lastPayDay" ] };

var getResult = server.GetUserInternalData(getRequest);

var lastPayTime = getResult.Data["lastPayDay"];

if(lastPayTime)

{

var lastTimeInMs = lastPayTime.Value;

var timeLapse = timeNow - lastTimeInMs;

log.debug("Time of previous payout: " + lastTimeInMs);

if(timeLapse > 10000)

{

log.debug("Sufficient time has pass, adding more funds.");

var currencyRequest = {

Amount: 10, PlayFabId: currentPlayerId, VirtualCurrency: "DU"

};

server.AddUserVirtualCurrency(currencyRequest);

var updateRequest = {

PlayFabId: currentPlayerId, Data: {

"lastPayDay": timeNow

}};

server.UpdateUserInternalData(updateRequest);

}

else

{

log.debug("Aborting! Funds can only be added once every 10 seconds.");

}

}

else

{

log.debug("This is the first time money is being requested.");

var currencyRequest = {

Amount: 50, PlayFabId: currentPlayerId, VirtualCurrency: "DU"

};

server.AddUserVirtualCurrency(currencyRequest);

var updateRequest = {

PlayFabId: currentPlayerId, Data: {

"lastPayDay": timeNow

}};

server.UpdateUserInternalData(updateRequest);

}

log.debug("Script completed.");

};

このスクリプトを定期的に実行して、プレーヤーのインベントリに架空の通貨を追加することができます。管理コードまたはサーバー コードからのみアクセスできる **GetUserInternalData** および **UpdateUserInternalData** を使用して要求が行われた頻度を追跡します。ゲーム クライアントから直接アクセスすることはできません。これにより、クライアントが架空の通貨を追加する要求をスプーフィングして必要以上に頻繁にスプールすることを防ぎます。この場合、サーバーは成功するために 10 秒に 1 回の頻度で要求を許可します。

# 定義済みのイベントに基づいてクラウド スクリプトを起動する

クライアントからクラウド スクリプトを呼び出すだけでなく、特定のイベントが発生したときに実行されるスクリプトを定義することもできます。これらのイベントは、PlayFab ダッシュボードの **オートメーション** タブの **規則** ページで設定されています。このサンプルでは、プレーヤーがタイトルにログインするたびに実行される簡単なスクリプトが設定されています。スクリプトは次のとおりです:

handlers.incrementLoginStat = function (args, context) {

var updateRequest = {

PlayFabId: currentPlayerId, Statistics: [{

StatisticName: "LoginCount", Value: 1

}]};

server.UpdatePlayerStatistics(updateRequest);

};

この規則には、トリガー イベント **com.playfab.player\_logged\_in** が割り当てられています。イベントが発生するたびに　**incrementLoginStat**　を実行する、**クラウド　スクリプト実行**タイプの**アクション**が追加されました。オプションでクラウド　スクリプトに引数を指定できます。

# 更新履歴

**初回リリース：**2018 年 6 月

# プライバシーステートメント

サンプルをコンパイルして実行すると、サンプルの実行可能ファイルの名前がMicrosoftに送信され、サンプルの使用状況の追跡に役立ちます。このデータ収集を無効にするには、Main.cppの「Sample Usage Telemetry」というラベルの付いたコードブロックを削除します。

マイクロソフトのプライバシーポリシー全般に関する詳細については、 [Microsoftのプライバシーステートメント](https://privacy.microsoft.com/ja-jp/privacystatement/)をご参照ください。