

PlayFab 클라우드 스크립팅

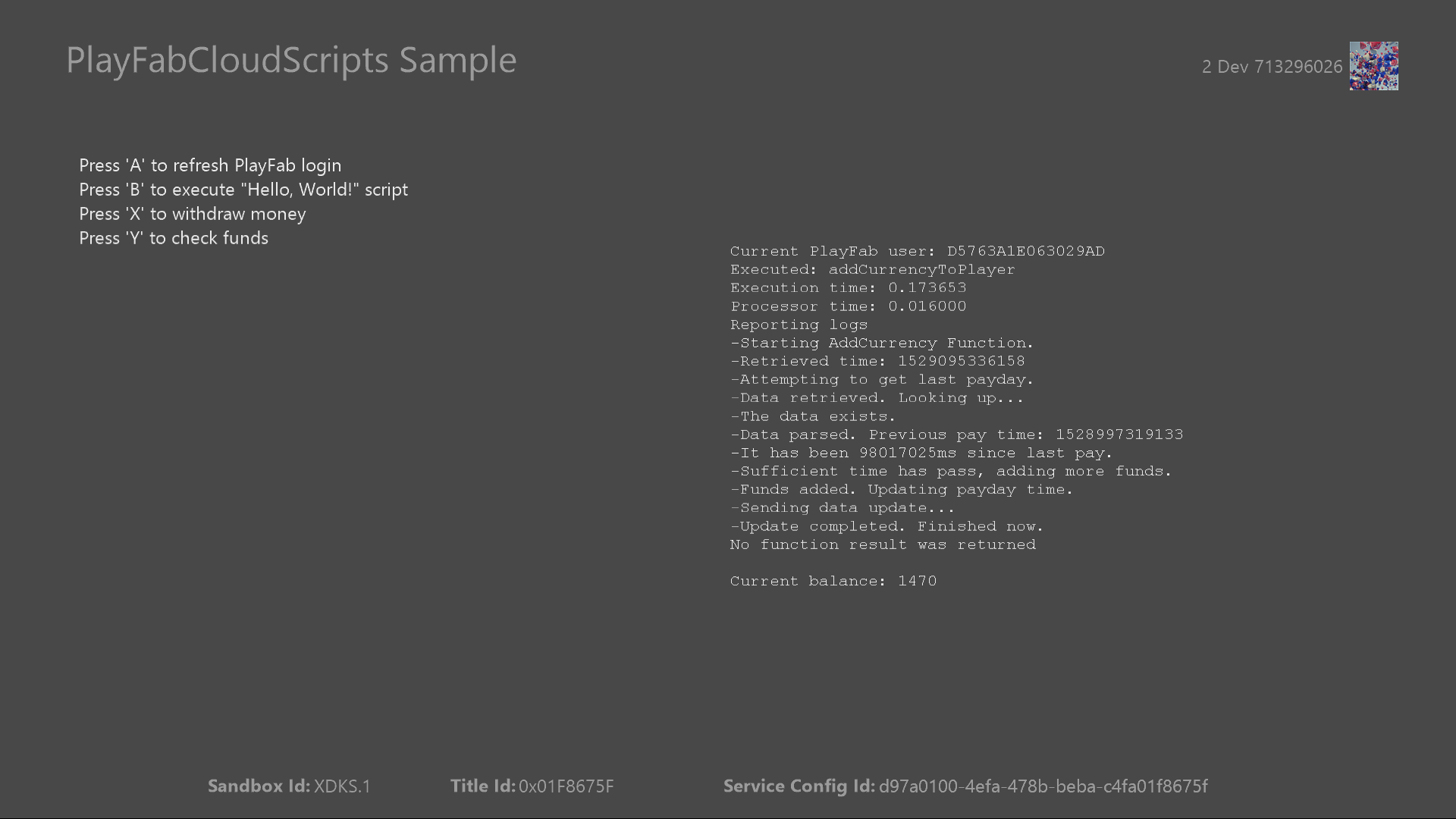
*\*이 샘플은 2018년 4월 XDK와 호환됩니다.*

# 설명

이 샘플은 클라우드 스크립트 호출을 보여줍니다.

# 샘플 사용하기

이 샘플은 XDKS.1 샌드 박스에서 작동하도록 구성되었습니다.



|  |  |
| --- | --- |
| 액션 | 게임 패드 |
| PlayFab 인증 토큰 새로 고침 | A 버튼 |
| "안녕, 세계"클라우드 스크립트 실행 | B 버튼 |
| 클라우드 스크립트를 실행하여 플레이어 자금 추가 | X 버튼 |
| 플레이어 가상 통화 쿼리 | Y 버튼 |
| 계정 선택기 열기 | 메뉴 버튼 |
| 종료 | 보기 버튼 |

# 구현 정보

이 버전의 XDK에서 샘플은 Xbox One 콘솔에서 XDKS.1 **SandboxID**을 사용해야 합니다. 이것을 전환하려면 **SandboxID**, Xbox One XDK 명령 프롬프트에서 다음을 사용하십시오.

xbconfig sandboxid=XDKS.1

PlayFab 타이틀 ID는 샘플 초기화 코드에서 설정해야 합니다. 이 샘플에서는 PlayFab 제목 ID "DC0"을 사용합니다. **PlayFabSettings::titleId** 정적 필드.

# 클라우드 스크립팅

PlayFab 서비스를 사용하면 PlayFab 대시 보드를 통해 JavaScript를 사용하여 사용자 정의 스크립트를 만들 수 있습니다. 이 스크립트는**ExecuteCloudScript** API 를 사용하여 또는 대시 보드에서 트리거 설정을 통해 게임 클라이언트에서 호출할 수 있습니다.

이 스크립트는 플레이어 통계 업데이트 또는 가상 화폐 부여와 같이 개별 또는 소규모 API 호출 그룹을 안전하게 실행할 수 있도록 하기 위한 것입니다. 클라우드에서 길거나 연속적인 스크립트를 실행하기 위한 것이 아닙니다. 실행 시간, 스크립트의 전체 크기 및 단일 스크립트에서 작성할 수 있는 API 호출 수를 포함하여 실행할 수 있는 클라우드 스크립트에 제한이 있습니다. 이 제한은 구독 수준에 따라 달라지며 대시보드에서 **설정** 탭의 **제한** 섹션에 있습니다.

이 샘플은 네트워크 활동을 플레이어 입력에 직접 연결합니다. 플레이어가 요청을 너무 자주 하지 못하도록 코드 가드가 포함되어 있습니다. 이것은 특정 XR에는 필요하지 않지만 필요 이상으로 네트워크가 포화되는 것을 방지하는 것이 좋습니다.

# "안녕, 세계"클라우드 스크립트

PlayFab은 참조 용으로 사용할 수 있는 많은 스크립트를 제공합니다. PlayFab 대시 보드에서 다음과 같이 볼 수 있습니다:

// This is a Cloud Script function. "args" is set to the value of the "FunctionParameter"

// parameter of the ExecuteCloudScript API.

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

// "context" contains additional information when the Cloud Script function is called from a PlayStream action.

handlers.helloWorld = function (args, context) {

// The pre-defined "currentPlayerId" variable is initialized to the PlayFab ID of the player logged-in on the game client.

// Cloud Script handles authenticating the player automatically.

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

// You can use the "log" object to write out debugging statements. It has

// three functions corresponding to logging level: debug, info, and error. These functions

// take a message string and an optional object.

log.info(message);

var inputValue = null;

if (args && args.inputValue)

inputValue = args.inputValue;

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

// The value you return from a Cloud Script function is passed back

// to the game client in the ExecuteCloudScript API response, along with any log statements

// and additional diagnostic information, such as any errors returned by API calls or external HTTP

// requests. They are also included in the optional player\_executed\_cloudscript PlayStream event

// generated by the function execution.

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

return { messageValue: message };

};

이것은 단순히 게임 클라이언트와 클라우드 스크립트간에 데이터를 전달하는 방법을 보여줍니다. 이 스크립트는 많은 부분에서 주석 처리되었으며 정적 로그 객체뿐 아니라 함수에 인수를 전달하는 방법을 설명합니다. 스크립트가 완료되면 **ExecuteCloudScriptResult** 객체는 스크립트 자체에서 오류가 발생하면 로그의 내용은 물론 함수의 반환 값 및/또는 오류 정보를 받습니다.

# 클라우드 스크립트에서 API 호출하기

통계 업데이트 및 트랜잭션 처리와 같은 작업을 수행하기 위해 클라이언트에 크게 의존하여 게임이 열리는 표면 영역을 악성 공격에 추가합니다. 이 샘플에서는 **addCurrencyToPlayer** 클라우드 스크립트에서 이러한 작업을 수행하여 공격의 가능성을 줄일 수 있는 방법을 보여줍니다. . 이 스크립트의 본문은 다음과 같습니다:

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.");

};

이 스크립트는 주기적으로 실행되어 플레이어의 인벤토리에 가상 화폐를 추가할 수 있습니다. 관리자 또는 서버 코드에서만 액세스 할 수 있는 **GetUserInternalData** 과 **UpdateUserInternalData** 를 사용하여 요청이 얼마나 자주 사용했는지 추적합니다.. 게임 클라이언트에서 직접 액세스할 수는 없습니다. 이렇게 하면 클라이언트가 원하는 것보다 더 자주 가상 통화를 추가하는 요청을 스푸핑 할 수 없게 합니다. 이 경우 서버는 10초마다 한 번만 요청을 허용합니다.

# 사전 정의된 이벤트를 기반으로 클라우드 스크립트 트리거

클라이언트에서 클라우드 스크립트를 호출하는 것과 함께 특정 이벤트가 발생할 때 실행되는 스크립트를 정의할 수도 있습니다. 이러한 이벤트는 PlayFab 대시 보드의 **자동화** 탭의 **규칙** 페이지 에서 설정합니다. 이 샘플에서는 플레이어가 타이틀에 로그인할 때마다 실행되도록 간단한 스크립트가 설정되었습니다. 스크립트는 다음과 같습니다.

handlers.incrementLoginStat = function (args, context) {

var updateRequest = {

PlayFabId: currentPlayerId, Statistics: [{

StatisticName: "LoginCount", Value: 1

}]};

server.UpdatePlayerStatistics(updateRequest);

};

규칙에 트리거 이벤트 **com.playfab.player\_logged\_in**가 발생했습니다. **액션** 유형 **Execute Cloud Script** 가 **incrementLoginStat** 이벤트가 발생할 때마다 실행을 위해 추가되었습니다. 선택적으로 클라우드 스크립트에 인수를 제공할 수 있습니다.

# 업데이트 기록

**초판:** 2018년 6월

# 개인정보처리방침

샘플을 컴파일하고 실행할 때 샘플 실행 파일의 이름이 Microsoft로 보내져 샘플 사용을 추적 할 수 있습니다. 이 데이터 수집을 거부하려면 Main.cpp에서 "샘플 사용 텔레메트리"라는 코드 블록을 제거하면 됩니다.

Microsoft의 개인 정보 취급 방침에 대한 일반적인 내용은 [Microsoft 개인 정보 취급 방침](https://privacy.microsoft.com/ko-kr/privacystatement/)을 참조하십시오.