**WordGame ゲーム保存サンプル**

このサンプルには、2016 年 3 月以降の Xbox One XDK、

または 2018 年 4 月 10 日アップデート SDK（17134） の Windows 10 が必要です。

# 概要

このサンプルは、完全なエンドツーエンドのゲームエクスペリエンスを構築するために Connected Storage API を使用した、Xbox One **XDK** *および* **UWP** のゲーム保存機能を紹介します。

このサンプルでは 以下のゲームのセーブ シナリオのオプションを提供します。

* **「完全同期」または「同期オンデマンド」モードを使用する**

デモンストレーションの目的で、サンプル起動時に、完全同期 API (コンソールとタイトル保存サービス間で、すべてのゲームのセーブ データを同期する)　または　同期オンデマンド API (ゲームのセーブ データのみを同期する)　のいずれかを使用することを、必要に応じて選択できます。  
注: 既に完全同期を実行していて、オンデマンド同期を試したい場合は、別のユーザーでサインインするか、ゲームのセーブ データのローカルキャッシュを消去する必要があります。

* + Xbox One のローカルキャッシュを消去するには、XDK コマンドプロンプトから「xbstorage.exe reset/force」コマンドを実行します。
  + Windows 10 PC のローカルキャッシュをクリアするには、管理者コマンドプロンプトから「gamesaveutil.exe reset」コマンドを実行します。注: Windows 10 Fall Creators Update SDK をインストールすると、次のディレクトリに **Gamesaveutil.exe** があります。  
    %ProgramFiles(x86)%\Windows Kits\10\Extension SDKs\XboxLive\1.0\Bin\x64
* **ゲームのセーブ データのロード、保存、削除**

メニューオプションを使用して (Get または Read API を使用して) ゲーム ボードをロードし、保存してから削除します。最大 9 枚のボードを保存できます。

* **コンテナと Blobs をリストする**

メニューオプションを使用して、コンテナと Blobs を列挙します。出力はゲーム画面のスクロール可能なデバッグ出力領域に表示されます。

* **最終変更日と残りのクォータを表示する**

この情報はゲーム画面のタイトルのすぐ下に表示されます。

* **ユーザーサインアウト時に自動保存する**

現在のゲーム ボードがまだ保存されていない場合は、ユーザーがサインアウトしたときに自動的に保存されます。

* **中断時に自動保存する**一時停止イベントに反応して、現在のゲーム ボードがまだ保存されていない場合、自動的に保存されます。

# サンプルをビルドする

**XDK ビルド要件**

* Visual Studio 2015
* Xbox One XDK

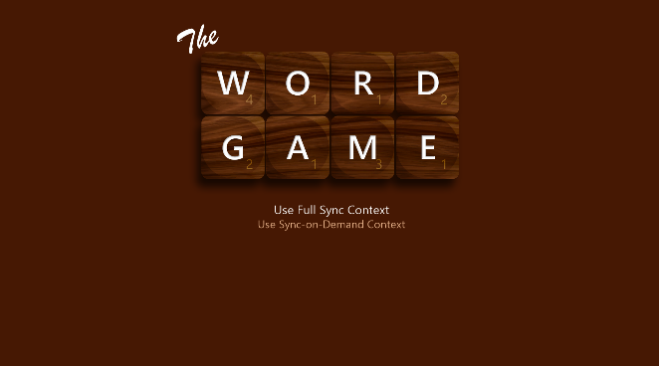
**UWPビルド要件**

* Visual Studio 2017 (15.7アップデート) もしくはそれ以降
* 2018 年 4 月 10 日アップデート SDK（17134） の Windows 10
* Xbox Live SDK バージョン 2017.11
  + 注: このサンプルを Visual Studio からビルドすると、Xbox Live SDK NuGet パッケージは nuget.org から自動的に復元されます。

# サンプルの使用

**Xbox Live サンドボックスの要件**

* Xbox One の開発キット: 本体のサンドボックスを XDKS.1 に設定します。
* Windows 10：デフォルトのリテールサンドボックスを使用するか、XDKS.1 に設定して PC とコンソール間でローミングするゲーム保存で試すことができます（Windows での Xbox Live サンドボックスの変更方法については、Xbox Live SDK に付属のドキュメントを参照してください）。

**スタート メニュー**

|  |  |  |
| --- | --- | --- |
| 作用 | ゲームパッド | キーボード/マウス |
| 「完全同期」 モードと 「同期オンデマンド」 モードの間で選択 | LS または D-Pad | 矢印キー |
| メニュー項目を選択 | Aボタン | Enterまたは  マウスを左クリック |
| 全画面表示切り替え |  | Alt Enter |

|  |  |  |
| --- | --- | --- |
| 作用 | ゲームパッド | キーボード/マウス |
| カーソルを移動 | LS または D-Pad | 矢印キー |
| メニュー項目を選択 | Aボタン | Enterまたは  マウスを左クリック |
| ゲーム セーブ スロットを選択 | LB/RB ボタン | 1～9 キーまたは  マウス |
| カーソルの下の文字のタイルを変更 | RS 左/右 | A～Z キー |
| カーソルの下の文字を消去する | X ボタン | 削除またはスペース |
| デバッグ出力をスクロールする | RS 上/下 | Page Up/Dn  および  Home/End |
| 全画面表示切り替え |  | Alt Enter |

**ゲーム ボード**



# ゲーム メニューのメモ

* **ボードを入手  
  GetAsync** API を使用して、現在のゲーム セーブ スロット用のゲーム ボードをロードします。
* **ボードを読む**  
  **ReadAsync** API (GetAsync の代替メソッド) を使用して、現在のゲーム セーブ スロット用のゲーム ボードをロードします。
* **ボードを保存**  
  **SubmitUpdatesAsync** API を使用して、現在のゲーム セーブ スロット用にゲームボードを保存します。
* **ボードをリセット**  
  ボードからすべての文字を消去します。ボードが以前に保存されている場合、ボードは「ダーティ」とマークされます。ボードがまだ保存されていない場合、ボードは「ダーティ」とマークされません。
* **ボードを削除  
  DeleteContainerAsync** API を使用して、現在のゲーム セーブ スロットのゲーム ボードを削除します。
* **ボード Blob を削除**  
  **SubmitUpdatesAsync** API を使用して、現在のゲーム ボードの Blobs だけを削除し、空のコンテナを残します。
* **コンテナの一覧表示**  
  **GetContainerInfo2Async** API を使用して、すべてのコンテナを列挙し、それらをゲーム画面のデバッグ出力領域に一覧表示します。
* **コンテナと Blobs の一覧表示  
  GetContainerInfo2Async** および **GetBlobInfoAsync** API を使用して、すべてのコンテナと Blobs を列挙し、それらをゲーム画面のデバッグ出力領域に一覧表示します。

ゲーム プレイのノート

**ゲーム プレイ**

ゲームは 5x5 のグリッドで行われます。グリッド上の任意の場所に文字を配置できます。認識された英語の単語を形成する連続した文字は、その単語の各文字のポイント値の合計に基づいて、ポイントを獲得します。目的はあなたのスコアを最大にすることです。各ボード上に配置する各文字の数は限られています。残りの数はゲーム ボードのすぐ上で追跡されます。

**ゲーム ボードのロード**

デモンストレーションの目的で、ゲーム ボード画面が表示されたとき、または新しいゲームセーブ スロットに切り替えたときに、ゲーム ボードは**自動的にはロードされません**。これにより開発者は、特定のゲーム セーブ スロットでロード方法 (Get または Read) を選択し、ロードおよびセーブ操作を完全に制御することができます。

**ゲーム ボードと自動保存の変更**

ゲーム ボードで文字が変更されたときや、[リセット] メニュー コマンドを使用した場合、ゲーム ボードは「ダーティ」とマークされます (画面上部のボード名の後にアスタリスクが表示されます)。ダーティー ゲームボードは以下の条件で自動保存されます。

* 別のゲームボードに切り替える (ゲームパッド LB/RB)
* ユーザー サインアウト
* ゲーム中断

# 実装上の注意

**GameSaveManager** のクラスはゲームのセーブ操作を管理します。**InitializeForUser（）** メソッドはプレーヤーの Connected Storage 保存コンテキストを設定します。セーブ データのロード、保存、列挙および削除のためのメソッドもあります。クラス内のメソッドの使用上の注意については、ヘッダーファイル内のコメントを参照してください。

ゲームで使用されるゲーム データ構造には、インデックスとゲームボード の 2 種類があります。テンプレート化された **GameSave** クラスは、GameSaveManager の使用する、任意の種類のゲーム データの一般的なロードおよび保存のためのメソッドを提供します。GameSaveManager.h の **GameBoardIndex**　構造体によって定義されたインデックスは、主にプレイヤーによって使用された最後のセーブ スロット (「アクティブ ボード」) を追跡するために使用されます。ゲーム　ボードのデータは、GameBoard.h の **GameBoard**構造体によって表されます。

# 既知の問題点

なし

# 更新履歴

**初回リリース：*2016 年 2 月***

Xbox One XDK 専用に 2016 年 2 月 XDK とともにリリース

**2016年3月更新：**

新要素：

* ゲームパッド、キーボード、およびマウス入力をサポートする UWP 版のゲームが追加されました（個別の Visual Studio プロジェクトが XDK と UWP で利用可能になりました）。
* UWP バージョンのサスペンドから再開すると、前回の同期以降に変更されたコンテナ-が表示されます（一時停止中にメモリにロードされた場合は、それ以降の変更を保存する前にリロードする必要があります）
* UWP バージョンは Xbox Live SDK バージョン 1602 と Xbox Live プラットフォーム拡張機能 （UWP） を使用します。

ゲームの保存プロセスに関する変更点：

* GameSaveManager の Suspend（） メソッドをリファクターし、OnSignOut（） メソッドを追加しました
* ダーティなアクティブゲームボードを保存することに加えて、それがダーティである場合、ユーザサインアウトによりゲームボードインデックスが保存されます。

その他の変更点：

* 単語リストファイルはかなり小さく、単語リストがバックグラウンドでロードされ続けている間、ゲームボードが表示されます

バグの修正：

* 一時停止中にゲームを再開し、ユーザーがサインアウトしたときのクラッシュバグを修正しました （null deref）

**2016 年 4 月更新：**

変更：

* 並べて比較しやすいように、GameSaveManager の実装を UWP と XDK のソースファイルに分離
* DeviceResources （UWP および XDK バージョン）を最新バージョンに更新
* PLM 中断処理の改善
  + 初期の初期化と同期の実行中に中断ているイベントが発生した場合、ゲームはゲームセーブインデックスをロード（およびおそらくリセット）しようとせず、ゲームセーブメタデータをロードしようともしません。
  + タスク実行中に中断しているイベントが発生した場合、ゲームは非同期ゲームセーブ初期化タスクの終了時にゲーム状態を InGame に切り替えようとしません

バグの修正：

* ゲームの保存の初期化中にエラーが発生したときに、ゲームが「ロード中」のメニュー状態で動かなくなることがあるバグを修正しました

**2016 年 5 月更新：**

変更：

* Xbox Live SDK のバージョン 1604 を使用するように UWP のバージョンが更新されました（ただし、バージョン 1602 および 1603 とコード互換性があります）。
* XDK バージョンでは、デバッグログファイルは T：の代わりに D：に保存されるようになり、タイトルが実行されていないときに利用できるようになりました。
* 渡された SCID が null の場合の、UWP GameSaveManager の InitializeForUser（） に警告が追加されました（ゲーム保存コンテキストはオフラインのみとなります）。
* コード品質の向上

**2017年3月更新：**

変更：

* UWP バージョンは WinRT（C / CX）XSAPI NuGet バイナリの代わりに C XSAPI NuGet バイナリを使用するようになりました
* UWP バージョンはデフォルトで Xbox Live SDK バージョン 1703 を使用しますが、以前のほとんどのバージョンと互換性があります。

**注**：Xbox Live SDK の NuGet バイナリは nuget.org からダウンロードできるようになりました。このサンプルを Visual Studio からビルドすると、自動的にダウンロード/復元されます。

* ユーザー管理に ATG 標準の「Xbox Live ツールキット」を使用するように変更
* マルチユーザー（MUA）版の UWP サンプルへの切り替えのサポートを追加しました（C++ XSAPIがこれを完全にサポートした後は、pch.h および appx マニフェストの変更をコメント解除することでアクティブ化できます）。
* UWP 用 Xbox Live SDK の 1703 バージョンで switch\_account（）API が削除されたことと互換性があるように、シングルユーザー UWP バージョンからアカウントの切り替え機能を削除しました。
* 入力制御が改善されました。ゲームのカーソルとメニューは、Left Stick、D-Pad、またはキーボードのカーソル矢印の任意の組み合わせで制御されます。
* ユーザー管理、UI フロー、および一時停止動作からの再開の改善が含まれています

**2017 年 4 月更新：**

変更：

* Xbox で実行している場合、UWP バージョンはデフォルトでマルチユーザーアプリ（MUA）になりました
* UWP バージョンが PC のデフォルト（RETAIL）サンドボックスで実行できるようになりました

**2017 年 9 月更新：**

変更：

* UWP プロジェクトを Visual Studio 2017（15.1 更新）および Windows 10 Creators SDK（15063）に更新しました。
* UWP ビルドには新しい Xbox Live 拡張機能 SDK 15063 が必要です

**2017 年 10 月更新：**

変更：

* 改善されたエラー処理と強化されたデバッグ出力

**2017 年 12 月更新：**

変更：

* UWP プロジェクトを Visual Studio 2017（15.4更新）および Windows 10 Fall Creators Update SDK（16299）に更新しました。
* UWP ビルドで Xbox Live Extensions SDK が不要になりました。これは、Connected Storage API の winmd が Windows 10 Fall Creators Update SDK に含まれているためです。
* Xbox Live SDK バージョン 2017.11 NuGet パッケージを参照するように UWP バージョンを更新
* 同期/ロード/非同期通話時間のログを追加

バグの修正：

* GetRemainingBytesInQuotaAsync（） が内部的にスローした場合の UWP バージョンでのクラッシュを修正しました
* 最初のロード時に誤った保存スロットが画面の上部に表示されることがあるバグを修正しました

**2018 年 5 月更新：**

変更：

* UWP プロジェクトを Visual Studio 2017（15.7更新）および Windows 10 April 2018 SDK（17134）に更新しました。

# 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。