**WordGame Game Save Sample**

This sample requires the Xbox One XDK March 2016 or later,

or the Windows 10 October 2018 Update SDK (17763)

# Description

This sample demonstrates game save functionality for Xbox One **XDK** *and* for **UWP** using the Connected Storage APIs to build a complete end-to-end game experience.

This sample provides options for the following game save scenarios.

* **Using “full sync” or “sync-on-demand” mode**

For demonstration purposes, on sample launch you have the option of choosing to use either the full sync API (which syncs all game save data between console and the title storage service) or the sync-on-demand API (which syncs game save data only as you need it).  
NOTE: If you’ve already performed a full sync and you want to experiment with sync-on-demand, you should either sign in with a different user or clear the local cache of game save data.

* + To clear the local cache for the Xbox One, you would run the “xbstorage.exe reset /force” command from the XDK command prompt.
  + To clear the local cache for a Windows 10 PC, you would run the “gamesaveutil.exe reset” command from an administrator command prompt. Note: **gamesaveutil.exe** can be found in the following directory once you’ve installed the Windows 10 Fall Creators Update SDK:  
    %ProgramFiles(x86)%\Windows Kits\10\Extension SDKs\XboxLive\1.0\Bin\x64
* **Load, Save, and Delete game save data**

Use the menu options to load game boards (using either the Get or Read APIs), save them, and delete them. You can save up to 9 different boards.

* **List containers and blobs**

Use the menu options to enumerate containers and blobs. The output is displayed in the scrollable debug output region of the game screen.

* **View last modified date and remaining quota**

This info is displayed just below the title on the game screen.

* **Auto save on user sign out**

If the current game board has not yet been saved, it will be automatically saved if and when the user signs out.

* **Auto save on suspend**In reaction to a suspending event, if the current game board has not yet been saved, it will be automatically saved.

# Building the sample

**XDK Build Requirements**

* Visual Studio 2015
* Xbox One XDK

**UWP Build Requirements**

* Visual Studio 2017 (15.7 update) or later
* Windows 10 April 2018 Update SDK (17134)
* Xbox Live SDK version 2017.11
  + NOTE: The Xbox Live SDK NuGet package is restored automatically from nuget.org when building this sample from Visual Studio

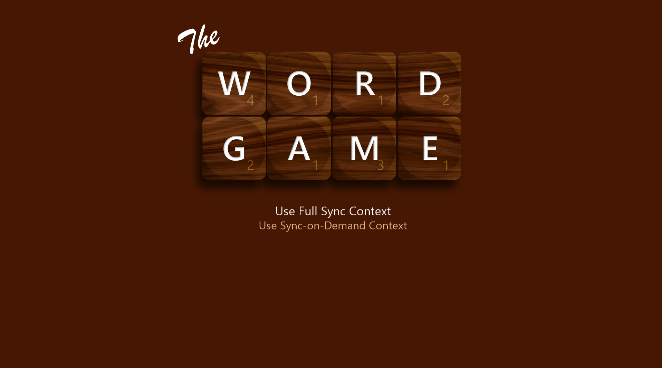
# Using the sample

**Xbox Live Sandbox Requirements**

* Xbox One devkit: set the console’s sandbox to XDKS.1
* Windows 10: you can use the default retail sandbox OR set it to XDKS.1 to experiment with game saves roaming between PC and console (see the docs included with the Xbox Live SDK for instructions on how to change the Xbox Live sandbox in Windows)

**Launch Menu**

|  |  |  |
| --- | --- | --- |
| Action | Gamepad | Keyboard / Mouse |
| Select between “full sync” mode and “sync-on-demand” mode | LS or D-Pad | Arrow keys |
| Select menu item | A button | Enter or  Mouse left-click |
| Toggle full screen |  | Alt+Enter |



**Game Board**

|  |  |  |
| --- | --- | --- |
| Action | Gamepad | Keyboard / Mouse |
| Move cursor | LS or D-Pad | Arrow keys |
| Select menu item | A button | Enter or  Mouse left-click |
| Select game save slot | LB/RB button | 1 – 9 keys or  Mouse |
| Change letter tile under cursor | RS Left/Right | A – Z keys |
| Clear letter under cursor | X button | Delete or Space |
| Scroll debug output | RS Up/Down | Page Up/Dn  and  Home/End |
| Toggle full screen |  | Alt+Enter |



# Game Menu notes

* **Get Board**Uses the **GetAsync** API to load the game board for the current game save slot.
* **Read Board**  
  Uses the **ReadAsync** API (an alternate method to GetAsync) to load the game board for the current game save slot.
* **Save Board**  
  Uses the **SubmitUpdatesAsync** API to save the game board for the current game save slot.
* **Reset Board**  
  Clears the board of all letters. If the board has been previously saved, this will mark the board “dirty”. If the board has not yet been saved, the board will not be marked “dirty”.
* **Delete Board**Uses the **DeleteContainerAsync** API to delete the game board for the current game save slot.
* **Delete Board Blob**  
  Uses the **SubmitUpdatesAsync** API to delete just the blobs for the current game board, leaving an empty container.
* **List Containers**  
  Uses the **GetContainerInfo2Async** API to enumerate all containers and list them in the debug output area of the game screen.
* **List Containers & Blobs**Uses the **GetContainerInfo2Async** and **GetBlobInfoAsync** APIs to enumerate all containers and blobs and list them in the debug output area of the game screen.

Game Play notes

**Game Play**

The game is played on a 5 x 5 grid. You can place letters anywhere on the grid. Consecutive letters that form a recognized English word, either across or down the board, will score points based on the total of the point values for each letter in the word. The objective is to maximize your score. You have a limited number of each letter to place on each board. The remaining count is tracked just above the game board.

**Game Board Loading**

For demonstration purposes, game boards **do not load automatically** when the game board screen appears, or when you switch to a new game save slot. This gives you, the developer, a chance to choose the load method (Get or Read) and have full control of load and save operations while on a particular game save slot.

**Changed Game Boards & Auto-Save**

When a letter has been changed on the game board, or when you use the Reset menu command, the game board will be marked “dirty” (indicated by an asterisk after the board name at the top of the screen). Dirty game boards will be auto-saved under the following conditions:

* Switching to a different game board (gamepad LB/RB)
* User signout
* Game suspending

# Implementation notes

The **GameSaveManager** class manages game save operations for the game. The **InitializeForUser()** method sets up a Connected Storage save context for a player. There are also methods for loading, saving, enumerating, and deleting save data. See the comments in the header file for usage notes on each method in the class.

There are 2 types of game data structures used by the game: an index and a game board. The templatized **GameSave** class provides methods for use by the GameSaveManager for loading and saving data generically for any type of game data. The index, defined by the **GameBoardIndex** struct in GameSaveManager.h, is used primarily to keep track of the last save slot used by the player (the “active board”). The game board data is represented by the **GameBoard** struct in GameBoard.h.

# Known issues

None.

# Update history

**Initial Release: *February, 2016***

Released with the February 2016 XDK, for Xbox One XDK only

**March 2016 Update:**

New Features:

* UWP version of the game added (separate Visual Studio projects are now available for XDK and UWP), with gamepad, keyboard, and mouse input support
* Resuming from suspend in UWP version shows any containers that were changed since last sync (if they were loaded in memory while suspended, they should be reloaded before saving any further changes)
* UWP version uses Xbox Live SDK version 1602 and the Xbox Live Platform Extensions for UWP

Changes related to the game save process:

* Refactored the GameSaveManager Suspend() method and added an OnSignOut() method
* A user signout will now save the game board index if it is dirty, in addition to saving a dirty active game board

Miscellaneous changes:

* The word list file is considerably smaller, and the game board will display while the word list continues to load in the background

Bug fixes:

* Fixed crash bug when resuming game and user had signed out while suspended (null deref)

**April 2016 Update:**

Changes:

* Separated GameSaveManager implementation into UWP and XDK source files for easier side-by-side comparison
* Updated DeviceResources (UWP and XDK versions) to latest version
* Improved PLM suspend handling:
  + Game won’t try to load (and possibly reset) the game save index nor will it try to load the game save metadata if a suspending event occurs while performing the initial initialization and sync
  + Game won’t try to switch the game state to InGame at the end of the async game save initialization task if a suspending event occurred during the task execution

Bug fixes:

* Fixed bug where game could get stuck in a “Loading” menu state when there’s an error during game save initialization

**May 2016 Update:**

Changes:

* UWP version updated to use Xbox Live SDK version 1604 (still code-compatible with version 1602 and 1603, however)
* For XDK version, debug log file is now saved to D: instead of T: so that it is available when title is not running
* Added warning in UWP GameSaveManager’s InitializeForUser() if SCID passed in is null (game save context will be offline only)
* Code quality improvements

**March 2017 Update:**

Changes:

* UWP version now uses C++ XSAPI NuGet binaries instead of WinRT (C++/CX) XSAPI NuGet binaries
* UWP version defaults to using Xbox Live SDK version 1703, though is compatible with most earlier versions

**Note**: the NuGet binaries for the Xbox Live SDK are now available for download from nuget.org. They should get downloaded / restored automatically for you when you build this sample from Visual Studio.

* Switched to using the ATG standard “Xbox Live Tool Kit” for user management
* Added support for switching to multi-user (MUA) version of UWP sample (activatable by uncommenting changes in pch.h and appx manifest, once C++ XSAPI fully supports this)
* Removed the Switch Account feature from single-user UWP version to be compatible with the removal of the switch\_account() API in the 1703 version of the Xbox Live SDK for UWP
* Input control improved: game cursor and menus are controlled via any combination of Left Stick, D-Pad, OR keyboard cursor arrows
* Includes improvements to user management, UI flow, and resume from suspend behavior

**April 2017 Update:**

Changes:

* UWP version is now a multi-user app (MUA) by default when running on Xbox
* UWP version can now run in the default (RETAIL) sandbox on a PC

**September 2017 Update:**

Changes:

* Updated UWP project to Visual Studio 2017 (15.1 update) and Windows 10 Creators SDK (15063)
* UWP build requires new Xbox Live Extensions SDK 15063

**October 2017 Update:**

Changes:

* Improved error handling and enhanced debug output

**December 2017 Update:**

Changes:

* Updated UWP project to Visual Studio 2017 (15.4 update) and Windows 10 Fall Creators Update SDK (16299)
* UWP build no longer requires the Xbox Live Extensions SDK because the winmd for the Connected Storage API is now included with the Windows 10 Fall Creators Update SDK
* UWP version updated to reference Xbox Live SDK version 2017.11 NuGet package
* Add logging of sync/load/save async call duration

Bug Fixes:

* Fixed crash in UWP version if GetRemainingBytesInQuotaAsync() throws internally
* Fixed bug where wrong save slots can be displayed at the top of the screen when first loaded

**May 2018 Update:**

Changes:

* Updated UWP project to Visual Studio 2017 (15.7 update) and Windows 10 April 2018 SDK (17134)

# Privacy statement

When compiling and running a sample, the file name of the sample executable will be sent to Microsoft to help track sample usage. To opt-out of this data collection, you can remove the block of code in Main.cpp labeled “Sample Usage Telemetry”.

For more information about Microsoft’s privacy policies in general, see the [Microsoft Privacy Statement](https://privacy.microsoft.com/en-us/privacystatement/).