**WordGame Game Save Sample (C++/WinRT)**

This sample requires the Xbox One XDK March 2017 or later.

# Description

This sample demonstrates game save functionality for Xbox One XDK using the Connected Storage APIs to build a complete end-to-end game experience. It demonstrates the use of Connected Storage using the **C++/WinRT** projection headers. This sample includes a simple implementation of the **IBuffer** interface that is compatible the C++/WinRT projections, and demonstrates the use of **CoRoutines** for parallel programming.

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.
* **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 2017 (15.3 update) or later
* Xbox One XDK

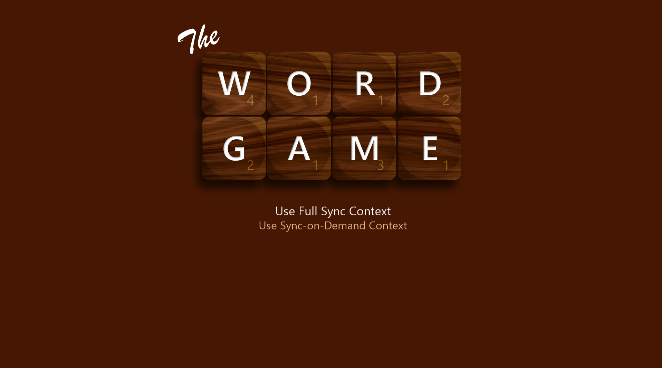
# Using the sample

**Xbox Live Sandbox Requirements**

* Set the sandbox of your devkit to **XDKS.1**

**Launch Menu**

|  |  |
| --- | --- |
| Action | Gamepad |
| Select between “full sync” mode and “sync-on-demand” mode | LS or D-Pad |
| Select menu item | A button |



**Game Board**

|  |  |
| --- | --- |
| Action | Gamepad |
| Move cursor | LS or D-Pad |
| Select menu item | A button |
| Select game save slot | LB/RB button |
| Change letter tile under cursor | RS Left/Right |
| Clear letter under cursor | X button |
| Scroll debug output | RS Up/Down |



# 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: *October 2017***

Released with the October 2017 XDK

# 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/).