

User Management Sample

*\* This sample is compatible with the March 2016 Xbox One XDK or later.*

# Description

This sample follows the User and Controller events that are received as users and controllers are added, removed, and their pairing changed within the underlying system. It also demonstrates acquiring an active user, and refreshing this user when resuming from suspend.

# Using the Sample



|  |  |
| --- | --- |
| Action | Gamepad |
| Show Account Picker | Menu button |
| List Controllers | X button |
| List This Controller | Left Trigger + X button |
| List Users | Y button |
| Set / Reset Active User | A / B buttons |
| Adjust Sign Out Deferral Time (in seconds) | DPad Left/Right |
| Exit | View Button |

# Implementation Notes

See the Initialize() method in UserManagement.cpp for all the User and Controller event handling code, which is implemented inside lambda functions. Scoped locks keep all the console output from each event handler together.

You may also want to see the Process Lifetime Management (PLM) code in the OnResuming() method in UserManagement.cpp to learn how to check for and handle a stale active user when resuming from suspended state.

A note on “Active User”

This sample simulates an “engagement prompt” to acquire an “**active user**” by prompting the user to press the A button on a gamepad. The current active user is shown in the upper left of the screen in this sample.

This helps demonstrate the Xbox Requirement that if a title takes user profile-related actions (e.g. writing rich presence for a user, saving user progress or preferences, awarding achievements, etc.), the title must indicate the active user(s) before any action is taken on a user’s profile. There are multiple ways to determine who the active user is. For most games, it is recommended to employ a user acquisition screen on launch, where the user is prompted to press a button to continue, and through this process the game can capture the active user. If only one user is signed in, it can be assumed they are the active user. If only one controller is connected, it can be assumed it is being used and queried for a user paired with it. If necessary, the game can prompt for a user to be picked using **ShowAccountPickerAsync**. After initial user acquisition, if multiple users can participate in the game, additional users can be added using additional calls toShowAccountPickerAsync.

Additionally, for comparison and experimentation purposes, the Xbox **CurrentUser** property is also shown at the top of the screen in this sample. Current User is part of the Xbox application model and tells you whose profile was the active profile in Home when the game was launched or resumed (from suspend mode).

Tracking the active user through this property and the **CurrentUserChanged** event is another way to implement user management but is generally not recommended for games. This property may not always update when the game might expect. For example, in the current Xbox OS, when a user presses the Xbox button on their gamepad to bring up the Guide and selects “Switch profile”, this can change the controller pairing but it will *not* change the Current User. For this and other reasons, we recommend implementing your own concept of “active user(s)” that fits best with your game and adheres to the Xbox Requirements on user management.

These concepts are discussed further in the XDK documentation and the Xbox Requirements documentation (see in particular **XR-112**: Establishing an Initial User and Controller, and **XR-113**: Active User Management, as well as other documents listed in the Resources section below).

Sample Limitations

Note the following limitations in this sample:

1. The “active controller” paired to the active user is not tracked
2. Kinect is not used to set the active user

# Things to Try

Each of the following actions will help demonstrate several important scenarios related to user management. Observe the changes and events shown in the on-screen log.

1. **Action**: Use the Xbox button on a gamepad to bring up the Guide and choose Switch profile.  
   **Observe**: Notice that various user and controller events may fire, but the Current User remains the same.
2. **Action**: Use the Xbox button to Go Home (either Dev Home or Retail Home), bring up the Guide and switch profiles, then resume the sample.  
   **Observe**: The Current User updates under this scenario. It’s up to the title whether to clear the active user and show the engagement prompt. This sample does not track the active controller for the active user.
3. **Action**: After signing in and establishing the active user, sign in somewhere else to demonstrate single point of presence (SPOP).  
   **Observe**: The following sequence of events occur:
   1. SignOutStarted event (this is where you could initiate saving game state, for example, while holding on to the signout deferral object)
   2. Sign out deferral takes place (default duration in this sample is set to 5 seconds)
   3. ControllerPairingChanged event removes the user pairing from a controller
   4. UserRemoved event
   5. SignOutCompleted event (this is where the active user is reset)
4. **Action**: After signing in and establishing the active user, use Xbox One Manager or Dev Home to suspend the sample, sign out the active user, and then resume.  
   **Observe**: On resuming, the sample finds a stale active user and, with that user no longer signed in, clears the active user and displays the engagement prompt again. Meanwhile, observe that the Current User property has not changed, nor were any SignOut events received.
5. **Action**: After signing in and establishing the active user, use Xbox One Manager or Dev Home to suspend the sample, sign out the active user, sign that user back in while the sample is still suspended, and then resume.  
   **Observe**: On resuming, the sample finds a stale active user, but since that user is signed in, it can refresh the active user object. Notice also that no SignOut events were received.

# FAQ

1. Why are there two (or more) gamepad controllers listed when I only have one plugged in to my console?  
   **Answer**: If you have Xbox One Manager running and connected to your devkit, it can show up in the list of Controllers as a gamepad. Try quitting Xbox One Manager and you should see a Controller Removed event.
2. I’m not seeing any user or controller events firing. What happened?  
   **Answer:** This can happen on rare occasion. Try restarting your console to correct a fault with one of the system services. We also encourage you to report this by sending feedback using the Report-a-Problem tool in Xbox One Manager.

# Resources

* XR-112: Establishing an Initial User and Controller
* XR-113: Active User Management
* XR-115: Addition and Removal of Users and Controllers
* XR-116: Handling Users and Controllers When Resuming From Suspended and Constrained Modes
* [Xfest 2014](https://developer.xboxlive.com/en-us/platform/documentlibrary/events/Pages/Xfest2014.aspx) Talk: “Managing Players and Input: Who’s Playing My Game?”  
  [Video](http://go.microsoft.com/?linkid=9845004) | [.pptx](https://developer.xboxlive.com/_layouts/xna/download.ashx?file=InputAudioSG_Managing_Players_and_Input_Who_is_Playing_My_Game.pptx&folder=platform\xfest2014)
* [Whitepaper](https://developer.xboxlive.com/en-us/platform/development/education/Pages/WhitePapers.aspx): “[Users, Controllers, and Pairing – Identity on Xbox One](http://aka.ms/9838344)”

# Known issues

None

# Update history

**Initial Release:** Xbox One launch

**March 2018 Update:**

* Uses newer sample framework
* UI overhaul
* Implements the concept of an Active User, captured through an engagement prompt and updated automatically when the user signs out or the sample resumes from suspended
* Checks for stale active user and refreshes as needed
* Tracks the CurrentUser property set by the system
* Supports individual sign in and pairing from *multiple* controllers
* Supports displaying info about the controller that pressed the button (i.e. current controller), in addition to displaying info about all controllers
* More events are logged to the onscreen console, including visibility and PLM events to help you place when user and controller events occurred

# 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/).