USB Mass Storage Device Support on Xbox 360

By David Cook
Senior Software Development Engineer
Advanced Technology Group (ATG)

Published: October 26, 2010

Overview

Allowing users to take advantage of their own USB storage devices is a great feature for gamers, but the performance of these devices can vary widely. This paper provides specific advice and best practices about how you can ensure your game runs well on a wide variety of storage devices.

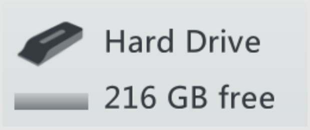
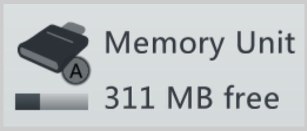
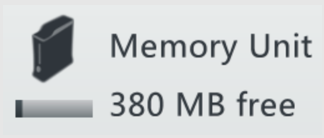
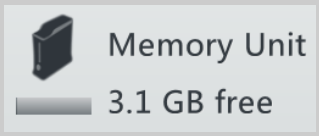
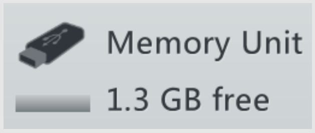
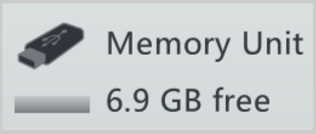
Starting with the Spring 2010 Xbox 360 system update, titles can mount attached USB devices, and save and load game data from them. Consumers can download XBLA games, Xbox LIVE Indie games, Games on Demand, Downloadable Content (DLC), title updates, and so on, to USB storage devices. Consumers can also install disc-based games to these devices.

At the Electronic Entertainment Expo 2010, Microsoft unveiled the Xbox 360 Slim console. This new model dispenses with the external memory unit (MU) ports, in favor of internal memory units (IMU) that act like USB storage devices. The IMUs serve as the primary option for user storage for the Arcade version of the Xbox 360 Slim. The new consoles are becoming increasingly prevalent in the consumer ecosystem. This warrants increased developer attention to USB storage.

Storage Media

Xbox 360 supports the types of offline storage shown in Table 1.

Table 1. Types of offline storage supported on Xbox 360 consoles.

Name & drive designator	Icon	Description
DVD DVD:\		The game disc itself, loaded by the optical disc drive (ODD).
HDD HDD:\	 Hard Drive 216 GB free	The attached Xbox 360 hard drive (HDD). The HDD comes in various SKUs, with order-of-magnitude differences in capacity and speed.
MU MU0:\ or MU1:\	 Memory Unit 311 MB free	The external memory units that plug into the slots on the front of older consoles. MUs come in various SKUs, with different capacities. Currently, they range from 64 MB to 512 MB.
Legacy IMU MUINT:\	 Memory Unit 380 MB free	The internal memory unit that exists in some older models of consumer and developer console. Legacy IMUs are similar in design to external MUs.
IMU INTUSB:\	 Memory Unit 3.1 GB free	The internal memory unit that ships with the Xbox 360 Slim models. These IMUs are USB devices that are designated solely for console storage. IMUs come in several varieties from different manufacturers.
Commercial USB Device USBMASS0MU:\ or USBMASS1MU:\	 Memory Unit 1.3 GB free	Any commercially available USB device capable of being used for Xbox 360 storage. Usable space ranges from 1 GB to 16 GB.
SanDisk Logo USB Stick USBMASS0MU:\ or USBMASS1MU:\	 Memory Unit 6.9 GB free	A particular 8 GB or 16 GB consumer USB device made by SanDisk, which is sold preformatted for use as Xbox 360 storage, and which has the Xbox 360 logo on it.

File Systems

Xbox 360 storage media uses the file systems shown in Table 2. For more details, see "Kernel File Systems" in the XDK documentation.

Table 2. File systems used by Xbox 360 storage media.

Name	Description	Accessible to title?
GDFX	The game disc file system (GDFX) used on DVD.	Yes
FAT32	The 32-bit file allocation table used on commercial USB storage devices.	No
FATX	The file allocation table for Xbox 360 used on MUs, IMUs, HDDs, and on the Xbox 360 partition of commercial USB storage devices.	No
SVOD	The secure virtual optical drive (SVOD) file system used by Games on Demand and by the Play From Hard Drive feature, by which DVD titles are installed to HDD, MU, IMU, and USB. It supports sizes larger than 2 GB.	Yes
STFS	The secure transacted file system (STFS) used for Xbox 360 Content Packages such as Xbox LIVE Arcade titles, game demos, downloadable content packages, title updates, title save games, and so on. It is limited to sizes of 2 GB or less.	Yes
STFC	The secure transacted file cache (STFC) used by the title utility partition on the HDD (and not on any other device type currently). It is limited to sizes of 2 GB or less.	Yes

USB Device Suitability

To qualify for use on Xbox 360, a USB mass storage device must have a capacity of at least 1 GB. Also, the first time the candidate device is plugged into an Xbox 360 USB slot, the system conducts various tests to determine the device's suitability. The tests consist of pseudo-random patterns of reads and writes, and both accuracy and speed are evaluated. The tests are not performed on IMUs or on unmodified SanDisk-branded logo USB sticks because those devices already are checked at the factory.

In an ideal situation, storage devices would be held to a strict performance bar. Experience has shown, however, that this approach is not feasible. Read and write performance fluctuates greatly based on factors invisible to the consumer—and sometimes to the developer as well. It is quite possible, for example, for a consumer to purchase two USB devices of the same model, at the same store, and have the system accept one and reject the other. Clearly, that would be a poor user experience! Rather than rejecting devices based on performance, therefore, the system simply provides a warning when the customer tries to format a slow device. The system only rejects USB devices outright if the devices fail data integrity tests.

Once a device is deemed acceptable, console system software creates a FATX container of the appropriate size, and divides it into a system partition and a consumer partition. The system partition occupies 512 MB of space. By default, the consumer partition occupies the remainder of the device capacity, or 16 GB, whichever is smaller.

Once the system partition is present, the system does not run any more performance tests. These test results are not entirely deterministic; however, once a device passes, it is considered acceptable automatically whenever a console encounters it.

Successfully formatted USB storage devices are treated by system software as analogous to MUs. The same set of operations is supported for both media. However, USB storage devices may have far greater memory capacity than MUs, and can support operations that were previously infeasible—operations such as installation of a full disc-based title.

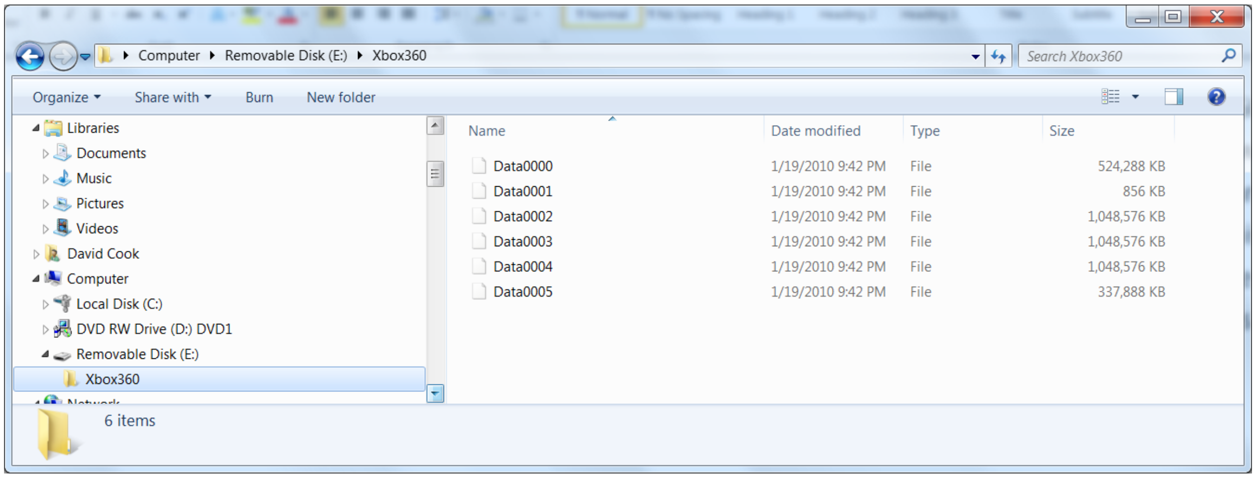
The console system software has a fixed limit of two external USB mass storage devices—whether they are formatted for Xbox 360 or not. This limit does not extend to the built-in hard drive or to standard MUs. The original Xbox 360 console has three USB ports (two in the front and one in the back), and the Xbox 360 Slim models have five USB ports (two in the front and three in the back—plus a dedicated port for Kinect). Nevertheless, if a user plugs in more than two USB mass storage devices, only the first two to be attached will be recognized. Sometimes, a USB device may have both an Xbox 360 partition and also other files recognizable to the console (for example, music tracks). Such a device still counts as only one device towards the previously described limit, but it will appear in the console UI as two separate devices.

Developer Experience

When a formatted USB device is read from a computer, the console data appears as an \Xbox360\ folder in the system root (the folder has the hidden attribute on Windows). This folder contains a number of files, including a 512-MB file corresponding to the system partition, and one or more files of 1 GB or less corresponding to the consumer partition. From the perspective of the console, this data forms a single unified file system. It is subject to the same signing and security measures that are applied to all Xbox 360 storage devices. External tampering with file contents is interpreted by the system as data corruption, and the affected content will fail to load.

Developers should never have to read, write, or modify the Xbox-formatted content on the device directly from a development computer. However, they can use either the console UI or Xbox 360 Neighborhood to modify the content.

Figure 1. Contents of an Xbox-formatted 4-GB USB stick attached to a computer, viewed as a local drive.



Name	Date modified	Type	Size
Data0000	1/19/2010 9:42 PM	File	524,288 KB
Data0001	1/19/2010 9:42 PM	File	856 KB
Data0002	1/19/2010 9:42 PM	File	1,048,576 KB
Data0003	1/19/2010 9:42 PM	File	1,048,576 KB
Data0004	1/19/2010 9:42 PM	File	1,048,576 KB
Data0005	1/19/2010 9:42 PM	File	337,888 KB

6 items

Figure 2. Contents of an Xbox-formatted 4-GB USB stick attached to a development kit, viewed from Xbox 360 Neighborhood.

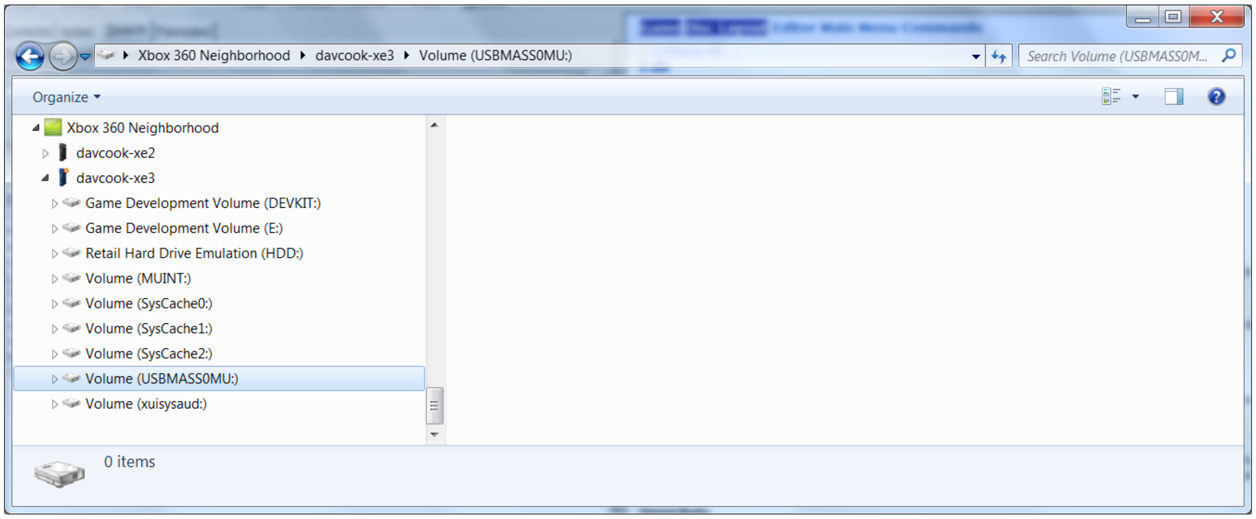
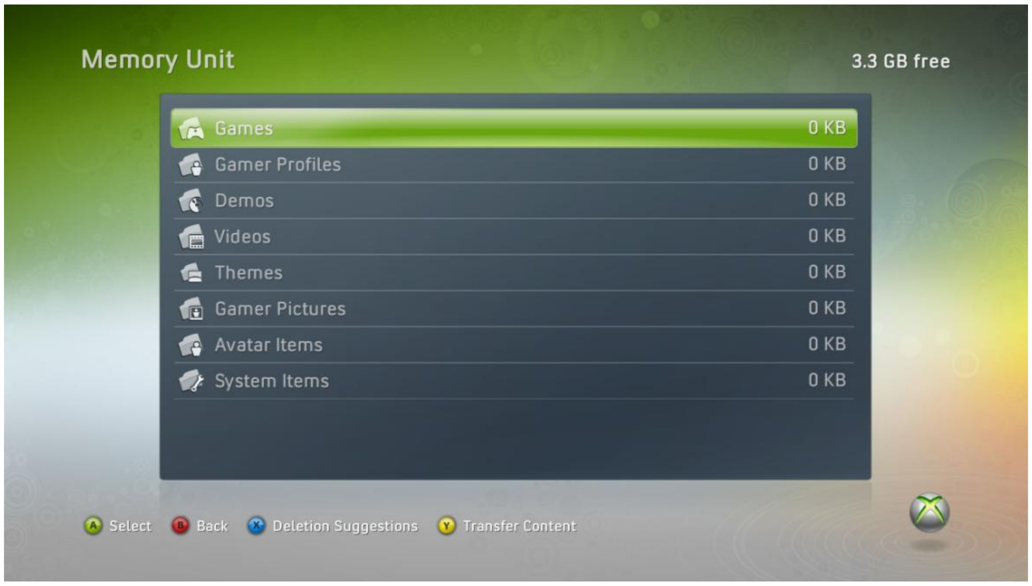

Xbox 360 Neighborhood ▸ davcook-xe3 ▸ Volume (USBMASS0MU:)
Search Volume (USBMASS0M...
Organize
Xbox 360 Neighborhood
davcook-xe2
davcook-xe3
Game Development Volume (DEVKIT:)
Game Development Volume (E:)
Retail Hard Drive Emulation (HDD:)
Volume (MUINT:)
Volume (SysCache0:)
Volume (SysCache1:)
Volume (SysCache2:)
Volume (USBMASS0MU:)
Volume (xuisysaud:)
0 items

Figure 3. Contents of an Xbox-formatted 4-GB USB, stick, viewed from the Xbox Dashboard.



Memory Unit — 3.3 GB free

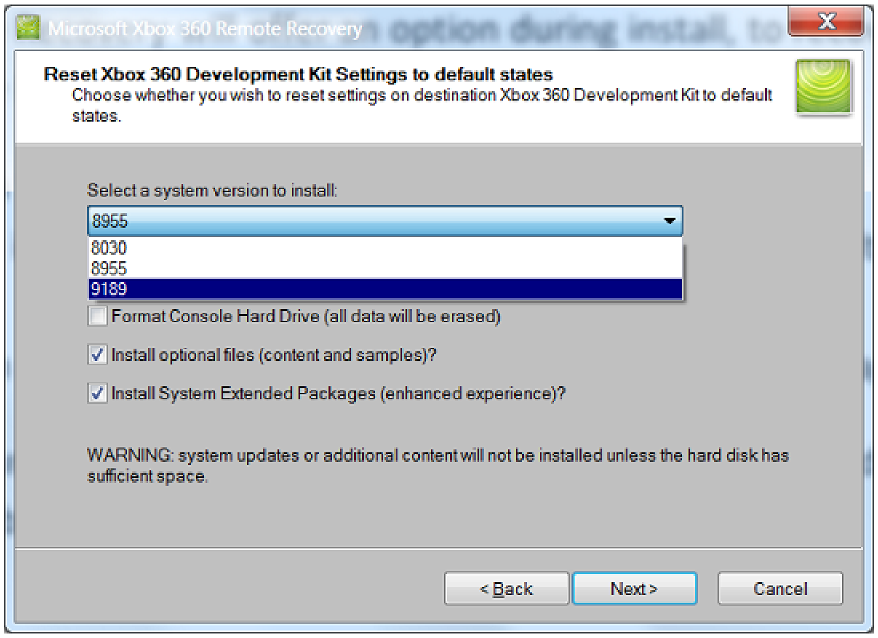
Games	0 KB
Gamer Profiles	0 KB
Demos	0 KB
Videos	0 KB
Themes	0 KB
Gamer Pictures	0 KB
Avatar Items	0 KB
System Items	0 KB

Consumer Experience

The following images show the current UI progression of selecting and preparing a mass market USB stick, which either passes or fails the performance checks. As always, aspects of the UI are subject to change in future system updates.

Consumers are offered two choices: *Configure now* or *customize*. When the consumer selects **Configure Now**, the entire device capacity, up to the maximum of 512 MB plus 16 GB, is formatted for use by the console. When selecting **Customize**, the consumer can configure the total amount of space used from 768 MB to the maximum. By selecting **Customize**, the consumer can preserve pre-existing, non-console data on the device.

Figure 4. Choice of flash version in the February 2010 Remote Recovery (actual flash numbers may vary).



Reset Xbox 360 Development Kit Settings to default states

Choose whether you wish to reset settings on destination Xbox 360 Development Kit to default states.

Select a system version to install:

8955

8030
8955
9189

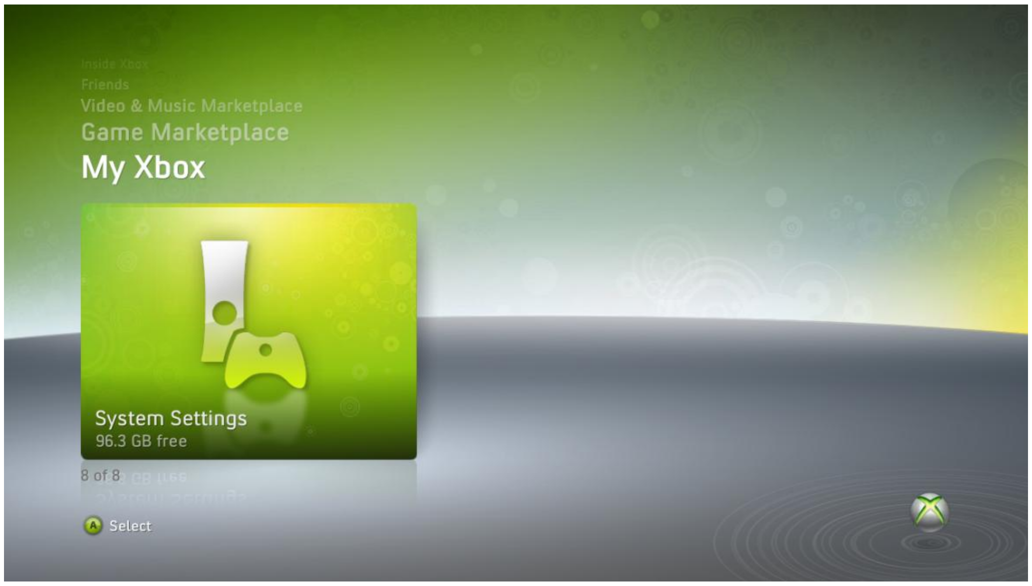
☐ Format Console Hard Drive (all data will be erased)

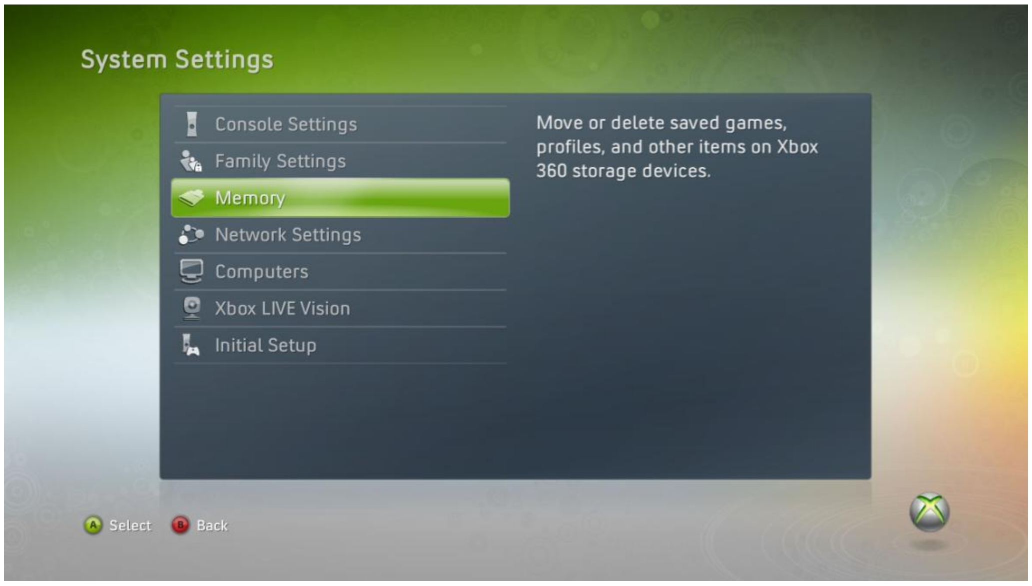
✓ Install optional files (content and samples)?

✓ Install System Extended Packages (enhanced experience)?

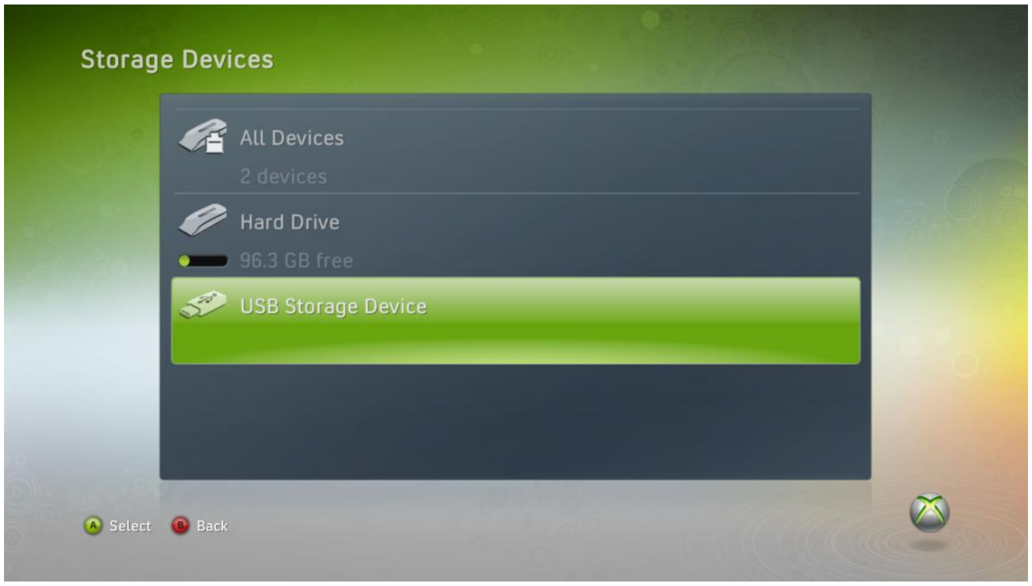
WARNING: system updates or additional content will not be installed unless the hard disk has sufficient space.

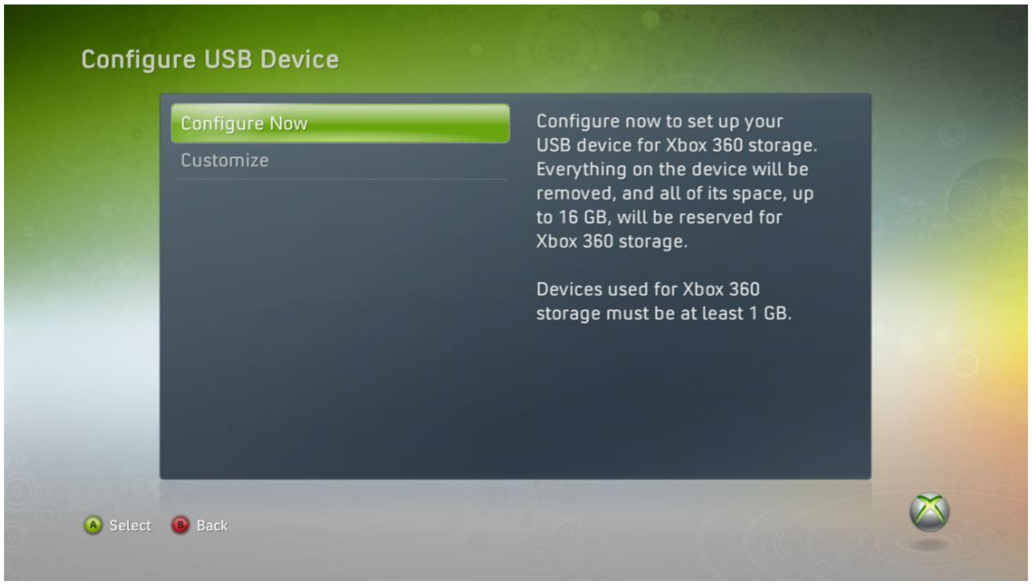
Figure 5. Result of failure to format a USB mass-storage device (six screens).





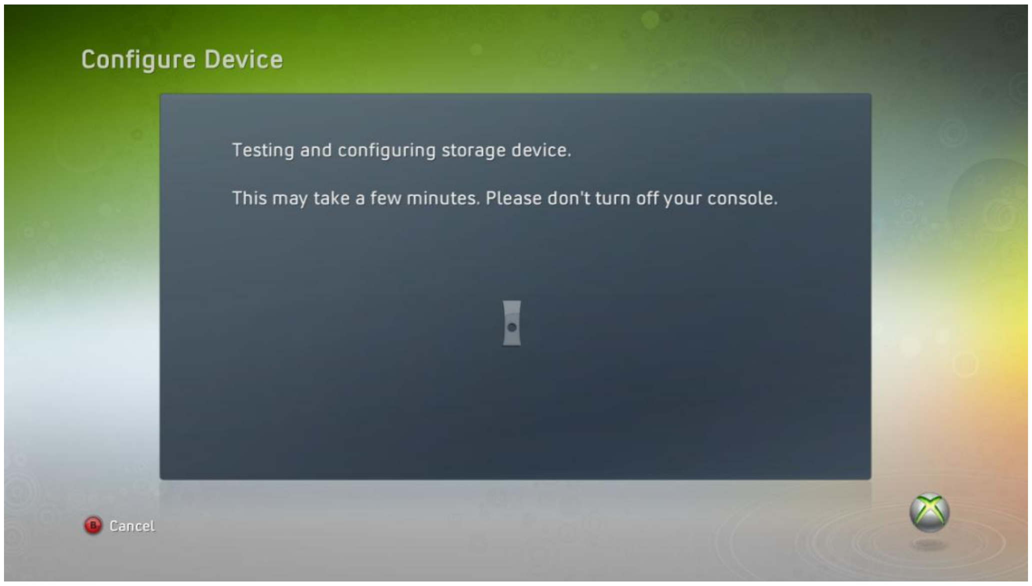
Move or delete saved games, profiles, and other items on Xbox 360 storage devices.




Configure USB Device

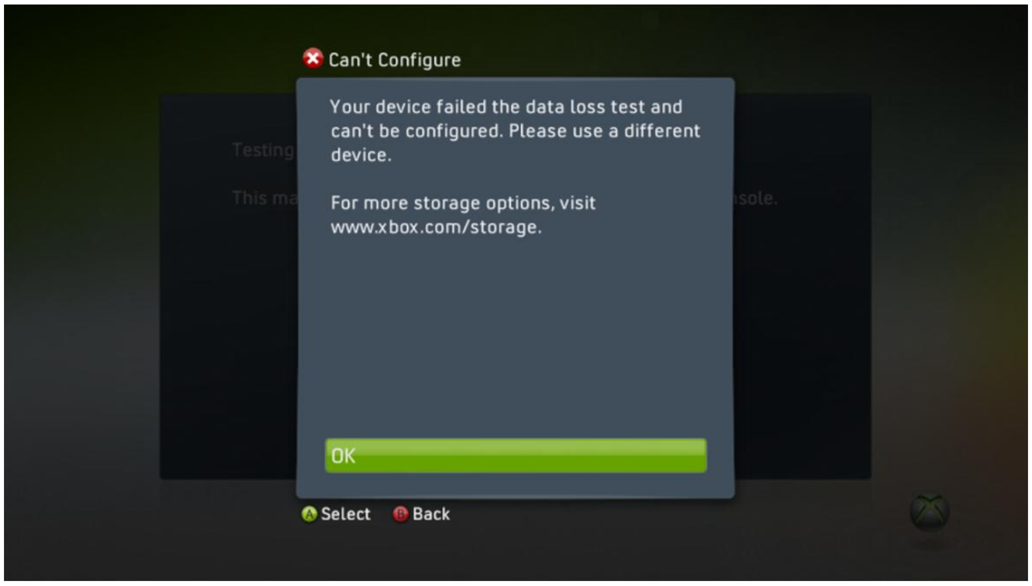
Configure now to set up your USB device for Xbox 360 storage. Everything on the device will be removed, and all of its space, up to 16 GB, will be reserved for Xbox 360 storage.

Devices used for Xbox 360 storage must be at least 1 GB.


Configure Device

Testing and configuring storage device.

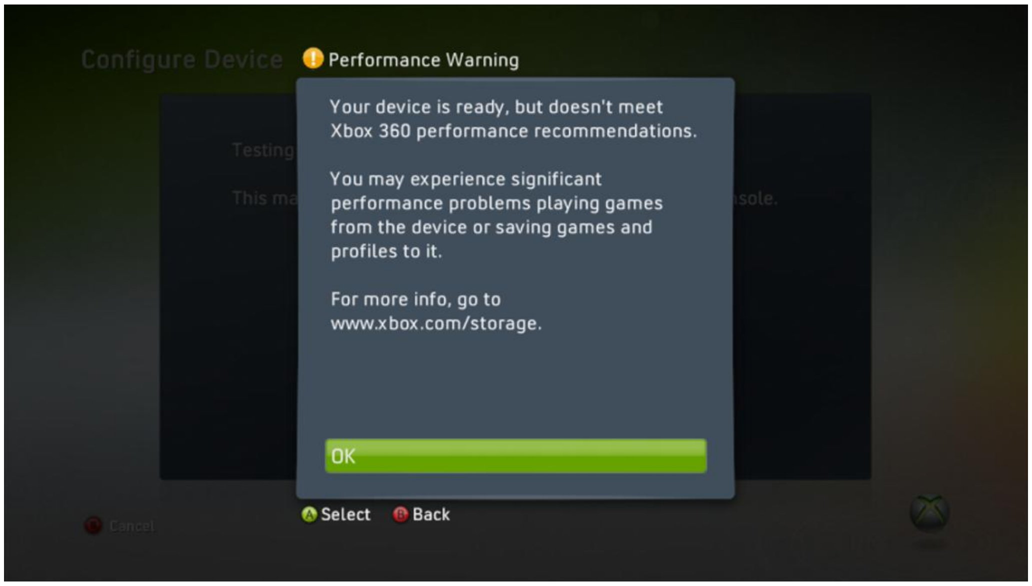
This may take a few minutes. Please don't turn off your console.


Can't Configure

Your device failed the data loss test and can't be configured. Please use a different device.

For more storage options, visit www.xbox.com/storage.

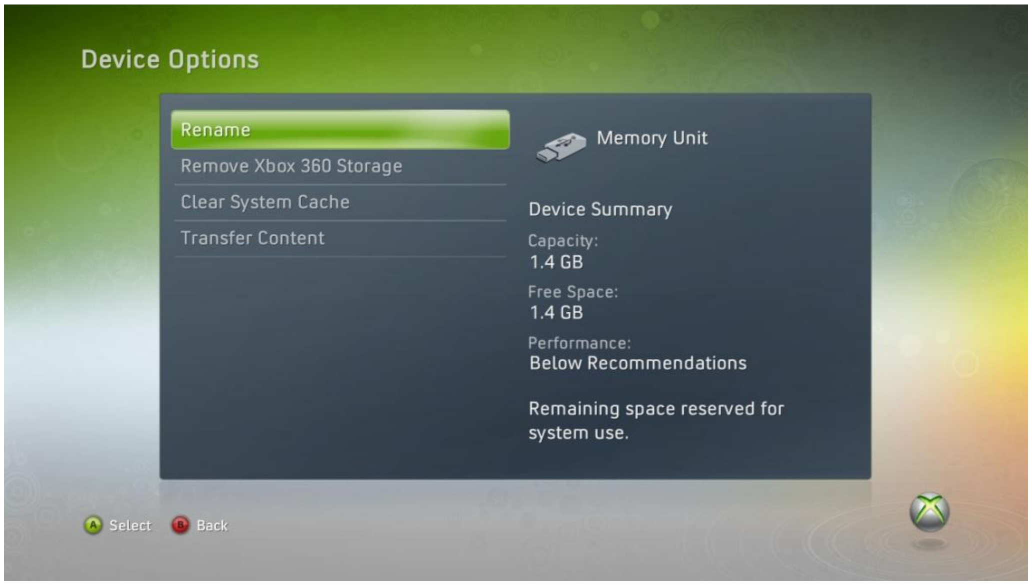
Figure 6. Two screens that show the result of successfully formatting a non-recommended USB storage device using the March 2010 preview.


Performance Warning

Your device is ready, but doesn't meet Xbox 360 performance recommendations.

You may experience significant performance problems playing games from the device or saving games and profiles to it.

For more info, go to www.xbox.com/storage.



Device Summary

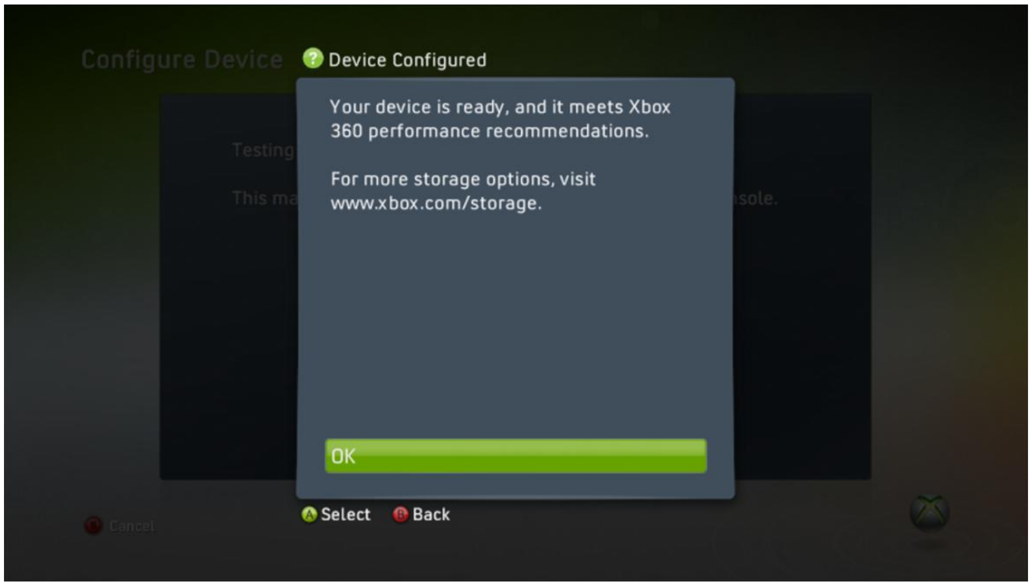
Capacity:
1.4 GB

Free Space:
1.4 GB

Performance:
Below Recommendations

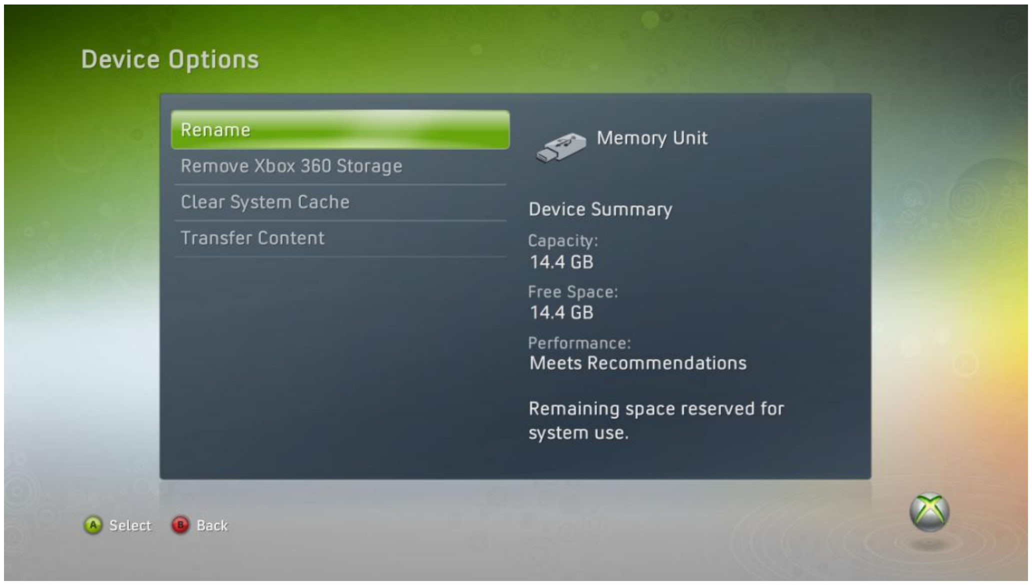
Remaining space reserved for system use.

Figure 7. Two screens that show the result of successfully formatting a recommended USB storage device using the March 2010 preview.



Your device is ready, and it meets Xbox 360 performance recommendations.

For more storage options, visit www.xbox.com/storage.



Device Summary
Capacity:
14.4 GB
Free Space:
14.4 GB
Performance:
Meets Recommendations

Remaining space reserved for system use.

Performance Factors for Storage Devices

The spectrum of USB storage devices is quite broad. As opposed to the ODD, whose behavior is sufficiently predictable to allow for software emulation, access times to other storage devices can vary by orders of magnitude, based on many different factors. The following subsections describe some of these variations.

Hardware

USB storage comes in many shapes and sizes, from high-capacity disc drives, to small keychain devices, to multipurpose devices such as phones and MP3 players. The underlying hardware has different performance characteristics. Moreover, newer and more expensive devices do not necessarily perform better than older devices. Often, cost is a function of capacity rather than speed. In addition, manufacturing advances are often designed to reduce cost or physical size, rather than increase performance.

It can be confusing to discover that different instances of the exact same device behave differently, based on random fluctuations in the manufacturing process. For example, two USB sticks of Brand X, used in identical fashion on the same game, can operate at different speeds.

Firmware

USB devices have fixed firmware sitting above the raw hardware. Some devices have on-board caches that can mask latency from reads and writes. Most devices can only read or write at a certain block size. For these devices, a non-aligned write turns into a read-modify-write operation, which incurs bandwidth hits in both directions. Devices write file metadata along with file contents. This metadata may need to be modified and flushed when certain events occur, such as a change in file timestamp or size.

Storage devices can become fragmented, causing a single read or write requests from the console to be broken down into multiple requests internally. Some devices have garbage collectors that occasionally defragment the set of used blocks. While it occurs, garbage collection can significantly reduce read and write speeds seen by the title.

Competing Device Activity

A data request to a USB storage device that is interrupted by other request to the same device will, not surprisingly, take longer to satisfy. A key scenario in this vein is a game installed to a device using SVOD, and simultaneously touching a package such as a save file elsewhere on the device. Not only must bandwidth be shared across the two packages, but the act of transitioning between the two physical locations may have additional adverse effects. Whenever the active file changes, either the device firmware or the software file systems may flush the associated file metadata.

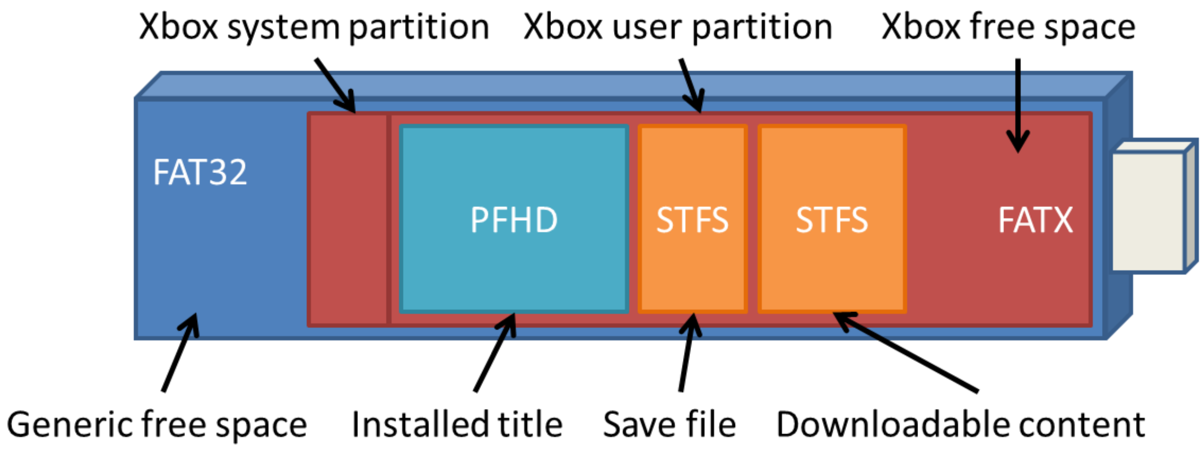
Competing Bus Activity

All current Xbox 360 consoles have two internal USB busses. Under heavy usage, one USB device can affect another device on the same bus. For example, any of the following activities might affect bandwidth of a separate USB device: playing a Kinect title, playing custom music tracks, executing a background download over the wireless network, and so on.

File System

Consumer USB storage devices use the FAT32 file system. As mentioned previously, the Xbox 360 accessible portion of the storage is formatted as a FATX container underneath the FAT32 root system. However, Xbox-only devices such as external or internal MUs, do not have the outer FAT32 layer. Game content on the device is stored in another file system underneath FATX. The secondary system can be SVOD for DVD-based titles installed to the device. It can also be STFS, for downloadable titles, for saved games, for downloadable content packages, or other content types. These file systems have different software code paths, different file caches, different security measures, and so on. Operations to access game data from storage might go through three nested layers of interface, each with its own associated overhead.

Figure 8. Sample file system hierarchy on commercial USB storage device.



SVOD

The SVOD file system mimics the GDFX layout of the game on DVD, with one substantial difference. When a title makes buffered reads from GDFX, the system expands the requests to 32-KB alignment, which is the size of an error correcting code (ECC) block on the disc. A system cache saves the resulting padding data. Often the cached data can be used to satisfy subsequent requests without additional transfer over the SATA bus. Buffered reads insulate the title from details of the hardware. SVOD has no ECC block cache. Thus, it can exhibit worse behavior on small reads.

STFS

Up until the time of this article, the STFS file system broke down all writes into 4 KB chunks. This approach worked well for older MUs, which have a native block size of 4 KB. It also worked well for HDDs, which have on-board caches to coalesce reads and writes. By contrast, many USB storage devices, including SanDisk-branded USB stick and the USB internal MUs, perform poorly for 4 KB reads and writes. Current IMUs, for example, use an 8 KB block size. In this situation, a 4 KB write translates to an 8 KB read-modify-write, potentially requiring 16 KB of bandwidth.

To address this problem, the Fall 2010 system update modifies STFS to write data at larger granularity. Tests have shown an approximate 2× to 3× improvement in write speed for the most-affected devices. Developers of titles that are sensitive to STFS write performance will want to consider whether their likely customers will have this fix or not—and also whether the fix is present on their development and QA hardware.

You can determine whether the STFS improvement is present on a particular console according to the criteria described in Table 3.

Table 3. STFS write improvement by user type.

User type	STFS write improvement becomes available…	Improvement applies to…
LIVE-connected consumer	When the **Fall 2010 system update** is released, and the consumer applies the update (which they must do in order to remain connected to LIVE).	All games retroactively
Non-LIVE-connected consumer	When the consumer first plays a title that is submitted to Xbox 360 Certification after the **Fall 2010 system update** was approved. (This approval is scheduled to occur in November 2010.)	All games retroactively
Developer	When the developer installs the **Fall 2010 system update** (or later version) on their development console. This system update will be available to developers in the November 2010 XDK.	All titles played while such a flash remains installed

Performance Measurements

In light of the many caveats of the preceding section, it is difficult to define and to measure representative performance numbers for USB storage. Nevertheless, developers still need to have a rough idea of the relative speeds of different devices. With that in mind, we show results of some tests that are conducted across different media types. Be aware that these tests were performed only on a single instance of a single model of each device type. Each entry is the result of multiple trials, each reading or writing a total of 8 MB, using **ReadFile** and **WriteFile** calls. The tables report a range of one standard deviation to either side of the mean.

FATX Throughput

The following test measures the speed of reading and writing data to the raw FATX container (after pre-sizing the file in the write case). A game running on a release flash cannot access the FATX root directly, but an application running on a debug flash can. This test separates the effects of the STFS/SVOD file systems from those of lower level file systems, and of the hardware itself. The graphs separate the HDD from the other devices because it is so much faster.

It is interesting to note that the performance of the non-recommended USB stick is comparable to and, in some cases, better than the performance of the approved devices. This test measures only sequential operations, however. By contrast, system recommendations are based on random access patterns.

Table 4. Measured read speeds (MB/s) on FATX.

Read 8 MB at block size	FATX: HDD	FATX: MU	FATX: Legacy IMU	FATX: IMU	FATX: Non-recommended USB	FATX: SanDisk USB
4 KB	15.98–18.54	3.18–3.22	10.75–10.84	2.77–2.85	4.07–4.2	4.02–4.07
8 KB	26.2–27.91	4.88–4.91	11.02–11.2	4.92–5.34	6.57–6.73	7.05–7.23
32 KB	49.03–50.41	7.99–8	11.18–11.28	10.02–10.52	11.13–11.8	12.08–12.24
512 KB	49.17–60.64	9.43–9.45	11.25–11.3	16.32–16.51	13.61–13.67	14.5–14.55

Figure 9. FATX read performance by block size.

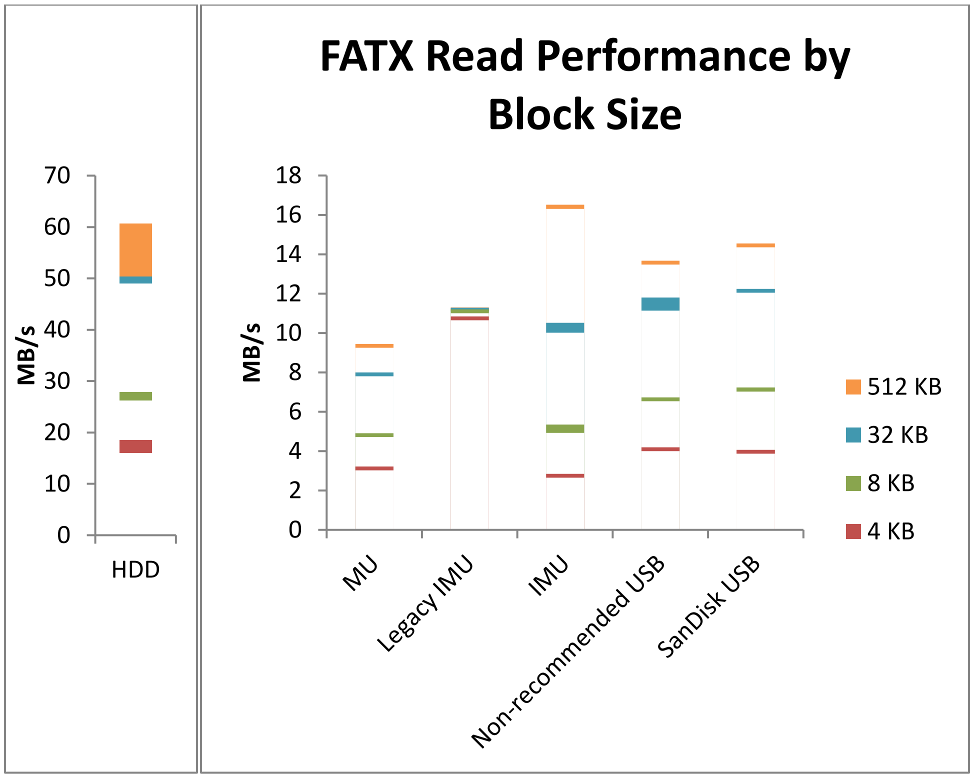
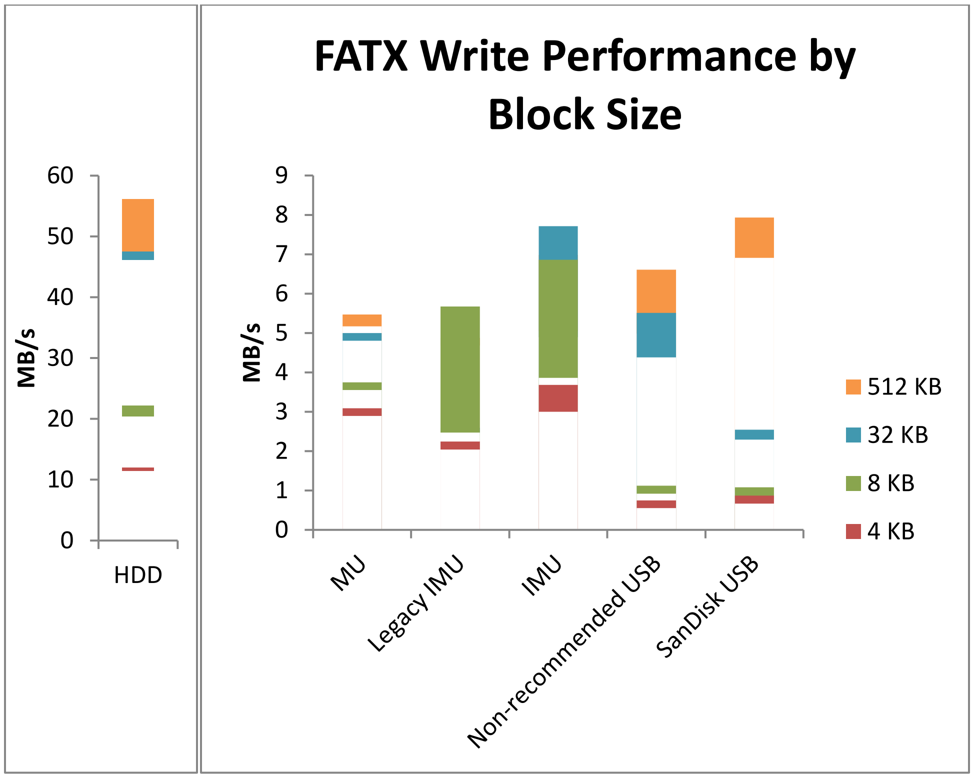


Table 5. Measured write speeds (MB/s) on FATX.

Write 8 MB at block size	FATX: HDD	FATX: MU	FATX: Legacy IMU	FATX: IMU	FATX: Non-recommended USB	FATX: SanDisk USB
4 KB	11.39–12.03	2.99–3.09	2.24–2.24	3–3.68	0.72–0.75	0.74–0.87
8 KB	20.4–22.17	3.69–3.75	2.47–5.67	3.86–6.86	1.07–1.12	0.77–1.08
32 KB	46.06–47.48	4.88–5	2.54–2.59	4.89–7.71	4.38–5.51	2.29–2.54
512 KB	47.5–56.09	5.17–5.47	4.77–4.88	5.05–6.98	4.98–6.61	6.91–7.93

Figure 10. FATX write performance by block size.



STFS Throughput

The following test measures the speed of reading and writing 8 MB of data to a newly created STFS container on a mostly empty device (so that fragmentation is unlikely). These results reflect best-case performance that an actual game would see when saving to or loading from content packages. The write tests are repeated using the legacy version and the upgraded version of STFS, which are described previously. The highlighted numbers show cases of substantial improvement.

Table 6. Measured STFS speeds (MB/s), before and after Fall 2010 system update.

Operation	STFS: HDD	STFS: MU	STFS: Legacy IMU	STFS: IMU	STFS: Non-recommended USB	STFS: SanDisk USB
Read 8 MB	26.86–29.1	8.67–8.7	9.39–9.54	13.48–13.55	12.97–13.37	11.35–11.61
Write 8 MB (July 2010)	6.86–7.34	2.05–2.11	3.69–4.22	0.75–0.87	0.62–0.65	0.68–0.81
Write 8 MB (>Fall 2010)	22.04–29.21	2.49–2.56	4.28–4.47	1.58–1.76	1.38–1.52	1.89–2.06

Figure 11. STFS read performance.

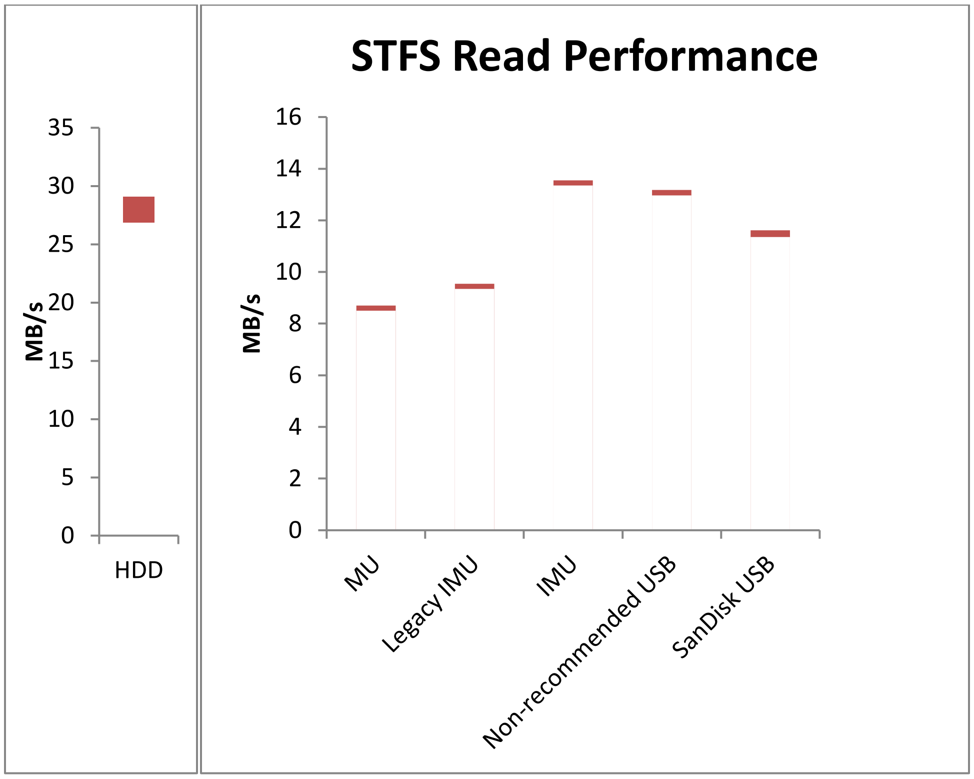
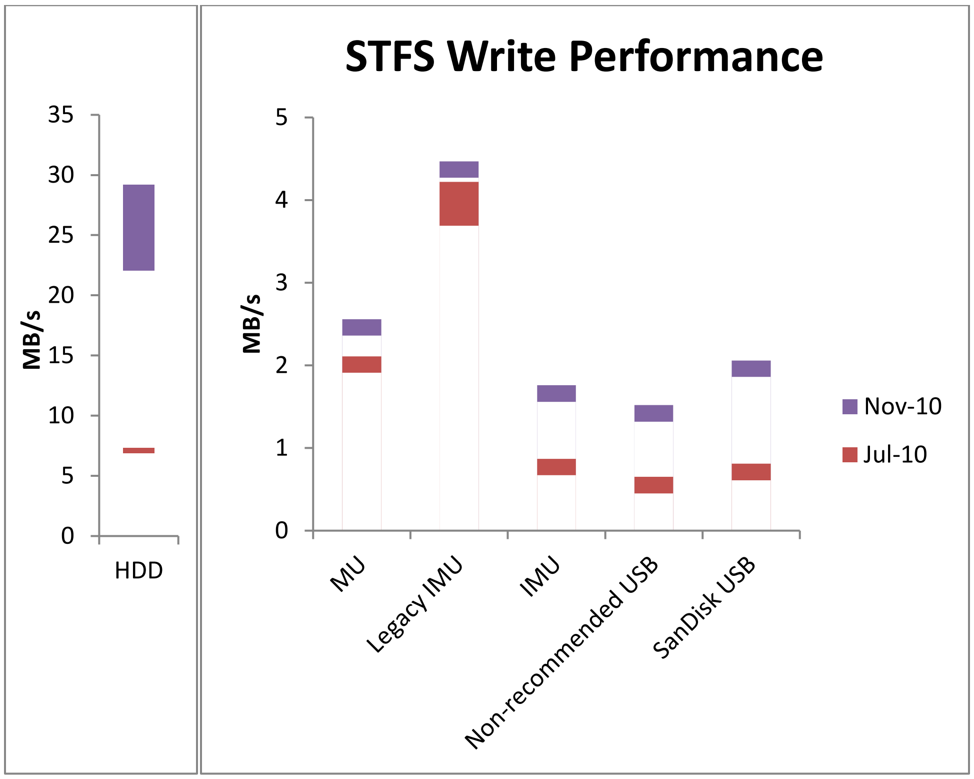


Figure 12. STFS write performance.



GDFX/SVOD Throughput

The last test measures the speed of reading 8 MB from a SVOD package. These results reflect the loading performance of a title installed to various media types. The results verify that SVOD essentially has the same read throughput as STFS on all media.

For completeness, we include timings for reading 8 MB from GDFX off the DVD, although GDFX does not operate with any USB media. These numbers were derived from DVD emulation, on a default game disc layout. As with other media, the ODD comes in different models from different manufactures. DVD emulation gives a lower-end estimate of performance.

SVOD and GDFX are read-only file systems, so only read rates are measured here.

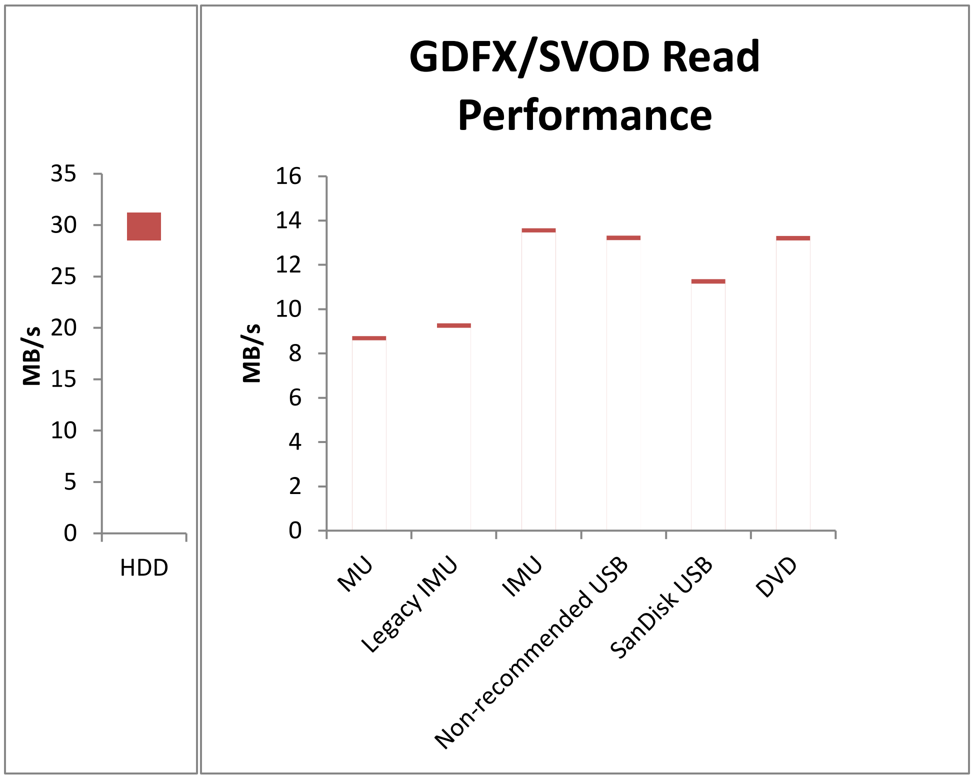
Table 7. Measured GDFX speeds (MB/s).

Operation	GDFX: DVD (emulated)
Read 8 MB	13.29–13.31

Table 8. Measured SVOD speeds (MB/s).

Operation	SVOD: HDD	SVOD: MU	SVOD: Legacy IMU	SVOD: IMU	SVOD: Non-recommended USB	SVOD: SanDisk USB
Read 8 MB	28.52–31.23	8.77–8.79	9.32–9.36	13.62–13.66	13.29–13.32	11.33–11.36

Figure 13. GDFX/SVOD read performance.



TCRs, Certification, and QA Recommendations

None of the Xbox 360 TCRs apply exclusively to USB mass storage devices, but all storage TCRs apply generically. For example, STR Game Data Storage, TCR 50, states that games must allow any available storage device to be used for saved data. This requirement applies to USB mass storage devices, as well as to traditional storage devices.

In spirit and in practice, though, the TCRs focus on the most prevalent devices, which are still the ODD and HDD. For example, *BAS Load Times*, TCR 7, states that overall load times must not exceed 60 seconds. This TCR is designed with the ODD and HDD in mind. Currently, no development hardware exists with USB IMUs. Clearly, it is not possible for developers to guarantee load times on devices that are not available to them during development. In the meantime, a title would not fail compliance testing based purely on performance with these unavailable IMUs, as long as the title met the requirements on standard media. Also, titles are not expected to satisfy time constraints on storage devices that are rated non-recommended by the system.

Titles are still expected to run without crashing, even with poor-performing devices. In the rare case where functional testing reveals other problems such as frequent hitches that make a title unplayable, Microsoft will work with the publisher to determine the best course of action.

The Xbox 360 certification team will test USB storage support on all titles, but due to time constraints this testing is not exhaustive. (For the up-to-date list of test cases, see Xbox 360 Compliance Test Cases and Xbox 360 Functional Test Cases.) Therefore, developers should not rely exclusively on the certification process to identify storage pitfalls. QA teams should acquire a number of mass market USB devices, and incorporate these devices into normal test processes. It should not be necessary to repeat every test case against every type of device. Instead, simply check that the title has no unexpected dependencies or behavior changes based on media type. Testing the saving and loading of game content with USB storage is the same as for traditional storage devices—just select the USB device as the target from the game's UI. Testing SVOD or content packages is analogous to testing from HDD, except that the content package must be copied onto USB media with the same relative path, as in this example:

```
xbcp -yrst hdd:\content\0000000000000000\00BC614E\00007000\*
usbmass0mu:\content\0000000000000000\00BC614E\00007000\
```

For general instructions on testing SVOD, see to "Testing Play from Hard Drive Support" in the XDK documentation.

Programming Recommendations

Engineers need to be aware of the ever-broadening possibilities for title and content media. No longer is it acceptable to target performance to a single media specification, or even to a narrow class of specifications. The following sections describe key situations to consider when designing your title's storage access patterns.

Title Saves to USB Storage

Many titles employ a save system that records a portion of the game state at a particular time. It is tempting to try a synchronous implementation first. In other words, pause the game, save all relevant data, resume the game, and see whether the pause is of reasonable duration. This method is guaranteed atomic, and requires no extra buffering. However, it produces a pause of unpredictable duration. If lower-speed media becomes prevalent post-ship, consumer perception of the title suffers. A more robust approach is to double-buffer the pertinent game state data, write one copy asynchronously to persistent storage, and allow the player to continue to play using the second copy.

Whatever approach you employ, titles can adopt several measures to ensure that saves complete quickly and to avoid unnecessary delays.

1. Pre-size the file. That is, give the destination file the correct size before writing begins. Files that grow incrementally incur substantial performance overhead. If the save file is written in a single operation, no action is necessary. Otherwise, the title can use code like the following:

   ```
   hFile = CreateFile(FileName, GENERIC_READ | GENERIC_WRITE, 0,
   NULL, CREATE_ALWAYS,
     FILE_ATTRIBUTE_NORMAL, NULL);
   SetFilePointer(hFile, dwFinalFileSize, NULL, FILE_BEGIN);
   SetEndOfFile(hFile);
   ```

```
        SetFilePointer(hFile, 0, NULL, FILE_BEGIN);
```

2. Write in large power-of-2 sizes, such as 64 KB. All devices operate more efficiently at native block sizes, as can be observed from the performance tables given earlier.
3. Write consecutively. Most file systems and devices have built-in optimizations for sequential access.
4. Write only what changes. If you know that only a small portion of the overall file has changed, consider overwriting only that portion, but padded to a reasonable block size. This decision comes with a tradeoff: If you modify only portions of a file, the file will become fragmented, but if you always overwrite the entire file, the operation will take longer. Although USB devices are essentially random access devices at the hardware level, fragmentation still reduces efficiency of on-board caching. Nevertheless, in many situations, minimizing write size should be prioritized over avoiding fragmentation.
5. Use a large cache size in **XContentCreateEx**. If the cache size is as large as the saved data, the double-buffering scheme recommended earlier, essentially can become transparent to the title.
6. Avoid hidden blocking operations. Some system I/O calls must wait until other pending operations have completed. It's not always obvious which combinations of calls will block. For instance, a **CreateFile** call following a **WriteFile** call will block until the write completes. A good method of handling such situations is to offload all I/O calls to a separate title thread—preferably one thread per storage device—and always poll that thread asynchronously from the game loop.

Title Installed to USB Storage, No HDD Present

As noted previously, USB storage devices can serve as alternate installation locations for Play from Hard Drive. That means that consumers can reap the benefits of offloading titles from ODD—quieter running, lower power consumption, and faster seeks—even on consoles without a hard drive. As in the past, SVOD requires that the original game disc remain present in the console drive tray, for verification purposes.

In this situation, streaming speed is different from when the title runs directly from ODD. Sustained data rates could be faster, or slower, depending on the media model and on the area of the ODD where the data resided. Seek penalties are very likely to be lower, but could conceivably be higher. Disc layer switch penalties are absent. Titles that depend heavily on the exact performance characteristics of the ODD may experience degradation, or, in some cases, improvement. At a minimum, all titles must have a fallback to remain playable from arbitrary media, under all performance conditions.

As mentioned previously, the GDFX file system on DVD employs a 32 KB buffer between the disc and the title, which mitigates the effects of many small reads. When the same reads are made from a USB mass storage device, the behavior may be very different—and also very device-dependent. Titles that make many small and/or unaligned reads and writes may observe drastic changes in performance among different devices. To avoid this possibility, try not to rely on OS optimizations to smooth out your access patterns. Whenever possible, align your data requests to a large natural granularity—32 KB is an easy choice because it has native support in the Game Disc Layout Tool (see "Game Disc Layout Editor Main Menu Commands" in the XDK documentation). And as always, try to follow general best practices outlined in the references at the end of this paper.

Title Installed to USB Storage, HDD Present

The 2-GB title utility partition, often used by games as a short-term file cache, is unaffected by the new USB support. Regardless of whether USB mass storage devices are present, the title utility partition will be created only on the internal hard drive, if the console has one. Performance characteristics of the title utility partition remain the same as in past flash versions.

However, titles that use the utility partition still should make one code change to account for USB storage. This change involves the determination of whether the title has been launched from HDD, or from another storage type. If the title was launched from HDD, it should refrain from using the utility partition, which is also on HDD. If the title was launched from ODD, or from USB mass storage, it should feel free to make normal use of the title utility partition—for example, as a secondary streaming source to improve access times.

The following function, **XContentQueryVolumeDeviceType**, can be used to query this information.

```
DWORD XContentQueryVolumeDeviceType(
     LPCSTR szRootName,
     XCONTENTDEVICETYPE *pDeviceType,
     PXOVERLAPPED *pOverlapped
);
```

With the value GAME for *szRootName*, the possible output values written to *pDeviceType* are shown in Table 9.

Table 9. Possible output values written to *pDeviceType*.

Value	Description
XCONTENTDEVICETYPE_HDD	Hard disc drive
XCONTENTDEVICETYPE_MU	Memory unit
XCONTENTDEVICETYPE_ODD	Optical disc drive

As noted previously, memory unit can refer to a standard MU, an IMU, or any USB mass storage device. The following code shows you how to query what type of device was used to launch a running title.

```
DWORD dwLicenseMask;
BOOL bIsPlayedFromHardDrive;

if ( XContentGetLicenseMask( &dwLicenseMask, NULL ) != ERROR_SUCCESS )
{
  printf( "Title was launched as a standalone DVD, or using DVD emulation, or from the
development area of the HDD.\n" );
  bIsPlayedFromHardDrive = FALSE;
}
```

```
else
{
  DWORD dwDeviceType;
  XContentQueryVolumeDeviceType( "GAME", &dwDeviceType, NULL );

  switch ( dwDeviceType )
  {
  case XCONTENTDEVICETYPE_HDD:
    bIsPlayedFromHardDrive = TRUE;
    printf( "Title was launched from a content package on HDD.\n" );
    break;

  case XCONTENTDEVICETYPE_MU:
    bIsPlayedFromHardDrive = FALSE;
    printf( "Title was launched from a content package on USB or MU.\n" );
    break;

  case XCONTENTDEVICETYPE_ODD:
    bIsPlayedFromHardDrive = FALSE;
    printf( "Title was launched from a content package on ODD.\n" );
    break;

  default:
    bIsPlayedFromHardDrive = FALSE;
    printf( "Title was launched from a content package on unknown device type.\n" );
    break;
  }
}
```

Hard-Coded Assumptions

Developers should avoid making specific assumptions about supported storage device types, available device capacities, or device performance characteristics. Consider, for example, a hypothetical title with a DLC pack 1 GB in size. At the time the title was developed, that DLC could only have been installed to HDD. But post-ship, customers could also install the DLC to USB mass storage. Suppose the title developer failed to check for the removal of storage device types other than the HDD. In that case, a player who removed a USB stick containing the DLC could crash the title.

Be careful of invalid assumptions—for example:

- Do not assume that a device labeled as type MU can have any fixed maximum size.
- Do not assume a large capacity storage device must be an HDD.
- Do not assume that a device labeled as type MU has fixed performance characteristics (beyond the minimum specifications cited earlier).
- Do not assume that a device with fast read access must be an HDD.
- Do not assume that ODD, HDD, and MU are the only device types (in particular, don’t have a **switch** statement on device type without a **default**: case).
- Do not assume that your game is launched from a particular device type, or even from any of the device types supported currently.
- When matchmaking over the network, do not assume, or require, that all candidate players have similar access speeds from storage.
- Do not assume that the maximum load or save time you can measure during development is the maximum load or save time for the consumer.

Data Compression

Now that your title can end up installed to a broad array of target media, data compression becomes, if anything, more important than ever. For example, a loosely compressed title may fit comfortably on a DVD at 4 GB. However, if that title could be reduced to 3 GB, it could also be installed onto a 4-GB USB stick. Even on an 8-GB USB stick, this size reduction could shave minutes off of the installation process—a savings for which your customers will be grateful. Current standards allow for a sustained write throughput as low as 2.5 MB/sec. At that rate, each additional GB of content takes 400 seconds, or nearly 7 minutes, to copy. We encourage all developers to take a careful look at the available compression techniques and recommendations in the XDK documentation, and in past publications.

Conclusion

As the lifespan of the Xbox 360 console continues, we are committed to adding value for customers through the addition of new features. USB mass storage support represents a part of that commitment. The storage ecosystem is always expanding. We encourage developers to think through the ramifications of this growth.

References

Streamlining Game Loading, Xbox 360 white paper

Developing with Xbox 360 Unified Storage, Xbox 360 white paper

Effective Use of Game Disc File Caching, Xbox 360 white paper

Squeezing It All In, Game Developers Conference 2009

The One Disc Game: Compression, Streaming, and Caching Effectively, Gamefest 2008