XGD3 Disc Format and Content Integrity Verification Overview

By Zsolt Mathe, Development Lead
Advanced Technology Group

Published: February 7, 2011

Over the lifetime of the Xbox 360, games have become more complex, requiring larger assets. The new disc format, also known as XGD3, extends the capacity of the game disc from 6.8 binary GB to 7.8 binary GB, precisely by 1,040,777,216 bytes. XGD3 formatted discs are compatible with all Xbox 360 consoles. There are no changes in title submission or the replication process. Additionally, XGD3 performance and reliability is similar to the XGD2 disc format, but with subtle differences in read bandwidth and seek times. With the additional space, titles can use higher quality or higher resolution assets to take full advantage of XGD3.

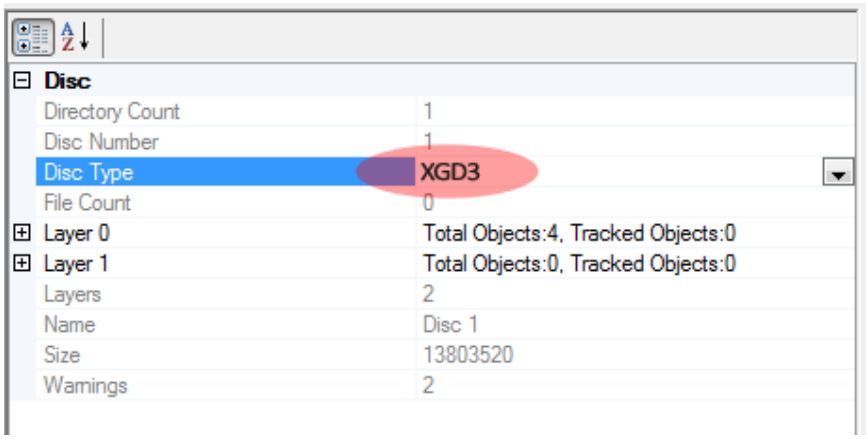
With the introduction of XGD3, there now is more anti-piracy and content integrity verification (CIV) support. There also is support for validating XGD3 media by using the new anti-piracy APIs. The size of XGD3 discs provides another layer of protection because they are too large to be burned onto a standard dual-layer DVD.

Given the benefits of both XGD3 and CIV, it is recommended that all titles consider using these features. Titles wishing to use XGD3 and CIV need to certify with the upcoming spring 2011 system update. The APIs are available in the February 2011 XDK.

Creating XGD3 Media

Starting with the February 2011 XDK, the Game Disc Layout Editor and Game Disc Emulator tools have preview support for XGD3 media creation and emulation. To create an XGD3 disc layout, open the Game Disc Layout Editor, and then select the **XGD3** option in the **Disc Type** parameter field. An existing XGD2 layout also can be switched to XGD3 in this manner.

Figure 1. Selecting the XGD3 option in Game Disc Layout Editor



After the disc type is set to XGD3, note that the available space—as indicated by the tool—is increased. At this point content can be added to the layout.

When using the Game Disc Emulator with a XGD3 disc layout, the emulator automatically uses XGD3 specific timings. As noted previously, the performance of XGD3 is similar to the original XGD2 format. There are minor differences, however, in read bandwidth, seek times, and layer switch times because the tracks are packed more densely on the disc. With XGD3, read bandwidth improved and seek times regressed; both increased by a few percent. As such, a well laid-out disc can yield faster streaming times with XGD3. Conversely, a poor layout with many seeks can yield worse performance with XGD3.

There are no changes to the title submission and certification process for XGD3. All replicators support it, and there is no additional per disc cost impact.

Content Integrity Verification

The content integrity verification system provides a title-controlled process to verify the integrity of the content. CIV is a security mechanism designed to detect pirated or modified media. The CIV algorithm consists of verifying a series of signed hashes of the game assets and of the auto-generated pseudo random filler data. All existing anti-piracy measures also are incorporated.

Following is a discussion about how to properly integrate the CIV APIs.

Content Integrity Verification APIs

To take advantage of the content verification system, the title must call the CIV APIs. It is the decision of the title to use these APIs; use of the XGD3 disc format does not require use of CIV.

The APIs consist of simple methods to check if media was altered. The full disc is verified, regardless of how much storage the title data consumes. When a title is played from another storage device, such as a hard drive or flash, CIV verifies the content on the game disc, not the content installed on the storage device.

Because these APIs read data from the disc, they should be used judiciously when the disc is idle. Calling them more often and throughout the gameplay session adds more protection against compromised media. How frequently to call depends on the title, the amount of content being streamed, and the general I/O bandwidth required by the title. Each time a content verification query is requested, the API uses 3 percent of a CPU single hardware thread during execution, and 75 KB of title memory. Typical call execution time can be a few seconds.

The CIV system is enabled by calling **XSecurityCreateProcess**, and is disabled by calling **XSecurityCloseProcess**. The main API that handles the verification is **XSecurityVerify**, which never blocks, even when null is passed in for the **OVERLAPPED** structure. The **OVERLAPPED** structure is provided for compatibility for minimal integration impact to titles that already use OVERLAPPED I/O. Without an **OVERLAPPED** structure, it is impossible to determine when the API finished executing. The execution time can be capped to a maximum number of milliseconds in order to limit the amount of disc I/O activity. Once **XSecurityVerify** is called, detailed content integrity information can be retrieved by calling **XSecurityGetFailureInfo**. Note that an error code returned from any of these APIs does not indicate a CIV

threat. **XSecurityGetFailureInfo** can return ERROR_LOCK_VIOLATION, which indicates that the media verification is still in progress. In this case, call **XSecurityGetFailureInfo** at a later time when the drive becomes free. If a failure is detected, it is likely the media is comprised, which presents the title with several options:

- Call **XShowAntiPiracyUI** to show a system-supplied message box.
- Disable save games, achievements, an XLSP connection, and any other LIVE APIs.
- Exit the title to the Dashboard.
- Delay giving any indication or feedback until a later time. This can be a helpful technique to deter pinpointing where the content verification happens in the title.

There also are more creative options. Ultimately, it is up to the title to decide how it handles a CIV failure.

The following code snippet demonstrates how to initialize and use the CIV system:

```
// Initialize CIV at title startup
DWORD dwResult = XSecurityCreateProcess( 4 ); // Use hardware thread 4
if( dwResult != ERROR_SUCCESS )
	return dwResult;


// Call XSecurityVerify periodically, when title is not reading from the disc
dwResult = XSecurityVerify( 500,  // Cap the execution time to 500 ms
                            NULL,
                            NULL ); // No completion routine
if( dwResult != ERROR_SUCCESS )
	return dwResult;  // This is not considered a CIV failure


XSECURITY_FAILURE_INFORMATION FailureInformation = {0};
FailureInformation.dwSize = sizeof( XSECURITY_FAILURE_INFORMATION );
dwResult = XSecurityGetFailureInfo( &FailureInformation );
if( dwResult != ERROR_SUCCESS )
	return dwResult;  // This is not considered a CIV failure. Possibilities
include I/O contention, out of memory or locking.
if( FailureInformation.FailedHashes > 0 ) // We have an integrity failure
{
	XShowAntiPiracyUI( TRUE ); // Show message box and terminate the title
}
```

During development, injecting failures is a must in order to exercise the failure code path in the title. To inject media failures, use **XSecurityInject**, and specify the number of hash and read errors. Because it is difficult to predict how a disc may be compromised, it is best to inject random failures at random times and to ensure the title behaves correctly by detecting a CIV failure.

Further Reading

This paper outlined the basics of the XGD3 disc format and the content integrity verification system, touching on both asset management and security. The following white papers discuss these topics in detail, and are recommended reading.

Streamlining Game Loading

Xbox 360 Security: Best Practices

Effective Use of Game Disc File Caching