What's New in MrSID Decode SDK 10.0.1.5246

Library	9.5.5 Version	10.0.1 Version	Description
zlib	1.2.11 (statically linked)	1.3.2 (statically linked)	General-purpose lossless data compression library used internally by the SDK for PNG writing, TIFF compression (deflate), and other compressed data streams. Upgrade includes security fixes (CVE-2022-37434 heap buffer overflow in inflate), performance improvements, and modernized build system. Statically linked — no separate DLL shipped.

What's New in MrSID Decode SDK 10.0.1.5245

Release Date: 2026 | **Build:** 10.0.1.5245 | **Previous Version:** 9.5.5 (2023)

1. Overview

The MrSID Decode SDK (DSDK) is a set of C++ libraries that provides a framework for creating image pipelines, enabling developers to efficiently read, decode, and manipulate geospatial imagery in a variety of formats including MrSID (MG2, MG3, MG4), JPEG 2000 (JP2/J2K), NITF, TIFF/GeoTIFF, and more.

Version 10.0.1 is a major release that delivers:

- Modernized platform support: New compiler toolchains for Windows, Linux, macOS, iOS, and Android, while retiring legacy platforms.
- Upgraded third-party geospatial libraries: PROJ and GDAL are major upgrades; LASzip is upgraded and consolidated.
- New image format capabilities: New PNG and JPEG output writers; ECW and camera RAW reading continue to be supported.
- Security and stability fixes: Continued fuzz-testing hardening and pixel-shift fixes in composite mosaics.

This release ships as a unified package containing two SDKs:

Component	Version	Purpose
Raster DSDK	10.0.1	Decode and manipulate raster imagery (MrSID, JP2, NITF, TIFF, ECW, etc.)
LiDAR DSDK	1.1.5	Decode and manipulate LiDAR point cloud data in MrSID MG4 and LAS formats

2. New and Updated Platform Support

This section describes which development environments and target operating systems are supported for building applications with SDK 10.0.1.

2.1 Added Platforms

The following platforms are newly supported in 10.0.1:

Platform	Compiler / Toolchain	What This Means
Windows (64 & 32-bit)	Visual Studio 2026 - VC++ 18.0 (VC18.0)	Build apps using the latest VS 2026 IDE. Pre-built libraries with VC18.0 toolset, providing newest MSVC runtime and C++ standard library compatibility.
Windows (64 & 32-bit)	Visual Studio 2022 - VC++ 17.0 (VC17.0)	Fully supported build environment. Apps target Windows 10, 11, Server 2019, Server 2022.
Linux	GCC 13.2.1 on RHEL 9	Latest GCC via Red Hat Developer Toolset on RHEL 9. Also runs on CentOS Stream 9, Rocky Linux 9, Ubuntu 24.04.
Macintosh	Clang 16.0 (Xcode 16) on macOS 15 Sequoia	Native macOS 15 (Sequoia) support. Universal binary - Intel (x86-64) and Apple Silicon (ARM64).
iOS	Clang 16.0 (Xcode 16) - iOS 16+	Build apps for iPhones/iPads on iOS 16+. ARM64 + x86-64 simulators. Static libraries only.
Android	NDK Clang 17.0.2 - API Level 24+	Updated Android NDK targeting API 24+ (Android 7.0). armeabi-v7a, arm64-v8a, x86, x86_64.

2.2 Removed Platforms (No Longer Supported)

The following platforms have been removed from SDK 10.0.1. If you are using any of these, you must upgrade your development environment before migrating.

Platform	Compiler / Toolchain	Reason for Removal
Windows	Visual Studio 2017 (VC14.0 Update 3+)	Microsoft ended mainstream support; runtime no longer maintained.
Windows	Visual Studio 2015 (VC14.0)	End-of-life. No security patches.
Windows	Visual Studio 2013 (VC12.0)	EOL since April 2024. Incompatible with modern C++ standards.
Linux	GCC 5.3.1 on RHEL 6.8	RHEL 6 EOL November 2020. Toolset v4.1 no longer maintained.
Linux	GCC 4.8.2 on RHEL 6.8	RHEL 6 and default compiler no longer receiving updates.
Macintosh	Clang 8.0 (Xcode 8.2) on macOS 10.12	macOS Sierra unsupported. Xcode 8.2 cannot

		build for modern targets.
Macintosh	Clang 7.0 (Xcode 7.3) on OS X 10.11	OS X El Capitan EOL. Xcode 7.3 lacks ARM64 support.
iOS	Clang 8.0 (Xcode 8.2)	iOS 8+ no longer relevant. App Store now requires iOS 16+.
iOS	Clang 7.0 (Xcode 7.3)	ARMv7/ARMv7s deprecated by Apple.
Android	NDK GCC 4.9 (API Level 12)	Google deprecated GCC; API Level 12 (Android 3.1) obsolete.
Android	NDK 13b on Ubuntu Desktop 12.04	Ubuntu 12.04 EOL 2017. NDK 13b no longer maintained.

3. Upgraded Third-Party Libraries

The SDK relies on several well-known open-source geospatial libraries. Below are the exact versions shipped in SDK 10.0.1 (verified from binary version metadata and the source tree), compared with 9.5.5.

Library	9.5.5 Version	10.0.1 Version	Change	What It Does
PROJ	4.9.2 (statically linked)	9.7.1 (proj_9.dll)	Major upgrade	Industry-standard library for cartographic projections and coordinate transformations. Jump from 4.9.2 to 9.7.1 brings newer EPSG codes, time-dependent transformations, grid-based datum shifts, vertical CRS, redesigned API. Requires proj.db (~9.6 MB) plus grid files in 3rd-party/share/proj/. Shipped in BOTH Raster and LiDAR.
GDAL	2.2.3 (gdal202.dll in LiDAR; static in Raster)	3.6.4 (gdal.dll in Lidar; static in Raster)	Major upgrade	Geospatial Data Abstraction Library for raster/vector format I/O. Upgrade brings COG support, multi-threaded I/O, PROJ 9 integration, improved NetCDF/HDF5/Zarr drivers. Verified via DLL metadata: ProductVersion=3.6.4, Product=GDAL/OGR. NOTE: 10.0.1 ships gdal.dll only with Raster SDK; LiDAR no longer bundles its own copy.
GEOS	3.5.0	3.5.0	Unchanged	Geometry Engine - Open Source. Computational geometry (intersections, unions, buffering). Used internally by GDAL. Shipped as geos.dll (C++) and geos_c.dll (C API) in BOTH SDKs.
Intel TBB	4.1 (March 2013)	4.1 (March 2013)	Unchanged	Threading Building Blocks for multi-threaded decoding. Source: tbb41_20130314oss. Verified via DLL metadata: ProductVersion=4,1,0,0, Product=Intel(R) Threading Building Blocks for Windows. Controlled via MrSIDImageReader::setMaxWorkerThreads().
LibRaw	Included (libraw.dll)	Included (libraw.dll)	Already in 9.5.5	Reads camera RAW from virtually all digital cameras (Canon CR2/CR3, Nikon NEF, Sony ARW, Adobe DNG, etc.). Already shipped in 9.5.5; continues in 10.0.1 (Raster SDK only).
ECW SDK (NCSEcw)	Included (NCSEcw.dll)	v5.4.0.1417 (NCSEcw.dll)	Already in 9.5.5	Hexagon ECWJP2 SDK for native ECW reading. Verified via DLL metadata: ProductVersion=5,4,0,1417, Product=ECWJP2 SDK. Raster SDK only.
LASlib (LAStools)	Included	Included	Unchanged	High-performance ASPRS LAS file library. Supports LAS 1.0-1.4. Shipped as laslib.dll (LiDAR SDK only).
LASzip	2.2.0 (LASZip.dll	3.4.4 (integrated	Upgraded + consolidated	Lossless compression for LAS (.laz). Upgrade 2.2.0 -> 3.4.4 brings improved compression and full LAS

	separate)	into laslib.dll)		1.4 compatibility. Now linked into laslib.dll - separate LASZip.dll no longer shipped.
PDAL	1.2.0	1.2.0	Unchanged	Point Data Abstraction Library. Pipeline framework for translating/manipulating point clouds. Shipped as pdalcpp.dll and pdal_util.dll (LiDAR SDK only).

3.1 Library Distribution Summary

Which third-party DLLs ship with each SDK:

DLL	Raster SDK	LiDAR SDK
gdal.dll (3.6.4)	Yes	No
proj_9.dll (9.7.1)	Yes	Yes
geos.dll (3.5.0)	Yes	Yes
geos_c.dll (3.5.0)	Yes	Yes
tbb.dll (4.1)	Yes	Yes
libraw.dll	Yes	No
NCSEcw.dll (5.4.0.1417)	Yes	No
laslib.dll (with LASzip 3.4.4)	No	Yes
pdalcpp.dll (PDAL 1.2.0)	No	Yes
pdal_util.dll (PDAL 1.2.0)	No	Yes

4. Raster DSDK Changes (v10.0.1)

The Raster DSDK is the core component of the MrSID SDK. It provides C++ classes for reading, decoding, filtering, and writing raster imagery using a pipeline architecture (reader -> filter -> writer).

4.1 DLL Naming Convention

Starting with SDK 8.5, the Windows DLL includes the version number in its filename, allowing applications to link against multiple SDK versions simultaneously without conflicts.

Component	9.5.5 Filename	10.0.1 Filename	Notes
C++ SDK DLL	lti_dsdk_9.5.dll	lti_dsdk_10.0.dll	Main SDK dynamic library. Ship with your application.
C API DLL	lti_dsdk_cdll_9.5.dll	lti_dsdk_cdll_10.0.dll	C wrapper for use from C, Python, etc.
C++ stub library	lti_dsdk.lib	lti_dsdk.lib	Unchanged. Link against this; loads versioned DLL.
C API stub library	lti_dsdk_cdll.lib	lti_dsdk_cdll.lib	Unchanged.

Important: Ship lti_dsdk_10.0.dll with all dependencies (tbb.dll, gdal.dll, geos.dll, geos_c.dll, proj_9.dll, and optionally libraw.dll, NCSEcw.dll).

4.2 New Features

4.2.1 PNG Image Writer

A new PNGImageWriter class (header: PNGImageWriter.h) allows writing decoded imagery to PNG. Useful for:

- Web-based map tile generation
- Lossless output with alpha (transparency) support
- Quick preview generation from large MrSID/JP2 files

4.2.2 JPEG Image Writer

A new JpegImageWriter class (header: JpegImageWriter.h) writes JPEG output directly. Useful for:

- Generating thumbnails and previews
- Web service endpoints serving imagery as JPEG
- Reducing output size where lossy compression is acceptable

4.2.3 ECW Image Reading (continued from 9.5.5)

NCSEcw.dll v5.4.0.1417 continues to be shipped, enabling native reading of ECW (Enhanced Compressed Wavelet) format - widely used for aerial photography and ortho-mosaics.

- ECW files opened via standard MrSIDImageReader/pipeline framework
- Supports ECW v2 and v3 file formats
- Mixed collections of MrSID, JP2, and ECW imagery using single SDK

4.2.4 Camera RAW Reading (continued from 9.5.5)

libraw.dll continues to be shipped, supporting 600+ camera RAW formats including:

- Canon (CR2, CR3), Nikon (NEF, NRW), Sony (ARW, SRF), Adobe (DNG)
- Phase One (IIQ), Fujifilm (RAF), Olympus (ORF), Panasonic (RW2)

Useful for photogrammetry/remote sensing workflows where imagery comes from drone or aerial cameras.

4.3 Bug Fixes

4.3.1 Fuzz Testing Fixes

Several bugs discovered through automated fuzz testing have been fixed. Fuzz testing feeds malformed or randomly generated input to uncover crashes or memory corruption. Fixes improve robustness for:

- Corrupted or truncated MrSID files
- Malformed JPEG 2000 codestreams
- Invalid metadata records
- Unexpected file header values

Critical for server-side applications (e.g., ExpressServer) where untrusted input may be encountered.

4.3.2 Composite Mosaic Pixel Shift Fix (LT-1990)

Fixed a bug where certain tiles in mixed-resolution MrSID composite mosaics exhibited a fractional pixel shift, causing visible seam artifacts at tile boundaries when decoding at certain zoom levels. Originally patched in 9.5.4.4709; now part of 10.0.1.

4.4 API Version

SDK version macros in lti_version.h:

```
#define LTI_SDK_VERSION  0x0A01    // Combined version identifier
#define LTI_SDK_MAJOR    10        // Major version
#define LTI_SDK_MINOR    0         // Minor version
#define LTI_SDK_REV      1         // Revision
```

Runtime version is also available via the mrsidinfo utility:

```
> mrsidinfo -version
Version 10.0.1.5245
Built Apr 22 2026 17:53:42 (release, 64 bits)
```

4.5 Supported Raster Formats

Format	Read	Write	Notes
MrSID MG2	Yes	No	Legacy MrSID format
MrSID MG3	Yes	No	Wavelet-based with lossless support
MrSID MG4	Yes	No	Latest MrSID with band selection, floating point
JPEG 2000 (JP2/J2K)	Yes	No	ISO/IEC 15444, including GMLJP2
NITF	Yes	No	National Imagery Transmission Format
TIFF / GeoTIFF	Yes	Yes	Via GeoTIFFImageWriter
ECW	Yes	No	Via NCSEcw.dll
Camera RAW	Yes	No	Via libraw.dll
JPEG	No	Yes	Via JpegImageWriter
PNG	No	Yes	NEW in 10.0.1 - via PNGImageWriter
BMP	No	Yes	Via BMPImageWriter (supports alpha band)
Raw / BBB	Yes	Yes	Band-sequential raw pixel data

5. LiDAR DSDK Changes (v1.1.5)

The LiDAR DSDK is a separate set of C++ libraries for working with point cloud data stored in the MrSID MG4 format. LiDAR data consists of millions of 3D points captured by airborne or terrestrial laser scanners, used for terrain modeling, vegetation analysis, urban planning, etc.

5.1 DLL Naming

Component	Filename	Notes
LiDAR SDK DLL	lti_lidar_dsdk_1.1.dll	Main LiDAR decode library
Stub library	lti_lidar_dsdk.lib	Link against this; loads DLL at runtime
TBB	tbb.dll	Required for multi-threaded point cloud decoding

5.2 Command-Line Utilities

Two CLI utilities ship in Lidar_DSDK/bin/:

5.2.1 lidarinfo

Displays LiDAR file information: spatial extent, point count, record format, CRS, channel definitions.

```
> lidarinfo -version
Version 1.1.5.5245
Built Apr 22 2026 17:53:42 (release, 64 bits)
```

Supported input: MrSID MG4, LAS (1.0-1.4), delimited text (TXT/CSV).

5.2.2 lidardecode

Transcodes MG4 LiDAR files to LAS or text. For extracting point data from compressed format.

5.3 Supported LiDAR Features

5.3.1 MG4 Format with Lossless Floating-Point Compression

MrSID MG4 for LiDAR uses wavelet-based compression with lossless floating-point support. Point coordinates (X,Y,Z) and attributes are compressed without precision loss. Compression ratios of 5:1 to 20:1 are typical, dramatically reducing storage.

5.3.2 LAS 1.4 Channel Support

Channel	Description
ScannerChannel	Identifies which scanner head captured a given point (multi-scanner systems)
ClassFlags	Bitfield flags for point classification (Synthetic, KeyPoint, Withheld, Overlap)
NearInfrared	Near-infrared intensity values, useful for vegetation analysis

5.3.3 GPS Time Channels

Channel	Description
GPSTime_Week	Seconds since midnight Sunday (GPS week time). Range: 0 to 604,800 seconds.
GPSTime_Adjusted	Satellite GPS time minus 1*10^9. Continuous reference, does not reset weekly.

5.4 Community Contributions

Lidar_DSDK/contributions/ contains additional community-contributed code:

- GeoTIFF - Example code for extracting GeoTIFF keys from LiDAR metadata using libgeotiff
- liblas - Example PointReader/PointWriter implementations using liblas
- SWIG bindings - Interface files for Python, Ruby, and C# bindings

6. Examples and Sample Code

The SDK ships with comprehensive examples in Raster_DSDK/examples/src/. A Visual Studio solution (examples.sln) is provided.

6.1 Decoding Examples

Example	Description
DecodeJP2ToBBB.cpp	Reads JPEG 2000 and writes BBB (Band-By-Band) raw. Basic pipeline.
DecodeJP2ToJPG.cpp	Reads JPEG 2000 and writes as JPEG. Demonstrates JPEG writer.
DecodeJP2ToMemory.cpp	Decodes JPEG 2000 region directly into in-memory buffer.
DecodeMrSIDBandSelection.cpp	Reads multi-band MrSID MG4 and decodes only selected bands.
DecodeMrSIDToMemory.cpp	Reads MrSID and decodes a region into memory. Demonstrates scene selection.
DecodeMrSIDToRaw.cpp	Reads MrSID and writes raw pixel data.
DecodeMrSIDToTIFF.cpp	Reads MrSID and writes as GeoTIFF. Most common use case.
DecodeMrSIDLidar.cpp	Reads LiDAR point cloud data from MrSID MG4 using LiDAR DSDK API.
DecodeNITFToBBB.cpp	Reads NITF and writes BBB raw. Defense/intelligence applications.

6.2 Pipeline and Architecture Examples

Example	Description
DerivedImageFilter.cpp	Custom image filter by subclassing LTIImageFilter.
DerivedImageReader.cpp	Custom image reader by subclassing LTIImageReader.
DerivedImageWriter.cpp	Custom image writer by subclassing LTIImageWriter.
DerivedStream.cpp	Custom I/O stream by subclassing LTIOStreamInf (HTTP, S3, etc.).
Pipeline.cpp	Complete pipeline construction: reader -> filter -> writer.
Mosaic.cpp	LTIMosaicFilter combining multiple inputs into single seamless output.

6.3 Utility Examples

Example	Description
ErrorHandling.cpp	Proper error handling using LT_STATUS codes.
GeoScene.cpp	LTIGeoCoord usage. Pixel <-> geographic coordinate conversion.
ImageInfo.cpp	Query image properties: dimensions, bands, datatype, colorspace, CRS, metadata.
InterruptDelegate.cpp	Cancel long-running decode using LTIInterruptDelegate.
MetadataDump.cpp	Reads and prints all metadata records (XMP, EXIF, GeoTIFF keys).
ProgressDelegate.cpp	Progress callbacks via LTIProgressDelegate.
SceneBuffer.cpp	LTISceneBuffer for BSQ data. Convert to BIP/BIL.

6.4 C and Cross-Language Examples

Example	Description
UsingCInterface.c	C API (ltic_api.h) for C, Python (ctypes/cffi), other non-C++ languages.
UsingCStream.c	Custom I/O streams using the C API.

UsingStreams.cpp	Advanced C++ streams: buffered, memory, sub-streams.

6.5 WPF Viewer Application

A complete WPF viewer is included in Raster_DSDK/examples/WPFViewer/. C# .NET application demonstrating:

- Loading and displaying MrSID, JP2, and other imagery
- Pan, zoom, and navigation
- Interop between C# and the native C++ SDK via C++/CLI bridge
- Targets .NET Framework 4.8

Solution: WPFViewer.sln. Project: PipelineViewer.csproj.

7. SDK Directory Structure

Complete directory layout of the SDK distribution:

```
MrSID_DSDK-10.0.1.5245-win64-vc18/
+-- README.txt                              # Top-level readme
+-- LICENSE.pdf                             # End User License Agreement
+-- examples/                               # Shared examples
|   +-- src/                                # Example source + VS solution
|   +-- data/                               # Sample MrSID, JP2 test files
+-- Raster_DSDK/                            # Raster Decode SDK
|   +-- bin/                                # CLI tools + runtime DLLs
|   |   +-- mrsidinfo.exe, mrsiddecode.exe
|   |   +-- gdal.dll, geos.dll, geos_c.dll, proj_9.dll, tbb.dll
|   |   +-- libraw.dll                      # Camera RAW support
|   |   +-- NCSEcw.dll                      # ECW reading support
|   +-- include/                            # 88 C++ and C API headers
|   |   +-- MrSIDImageReader.h, J2KImageReader.h, NITFReaderManager.h
|   |   +-- PNGImageWriter.h                # NEW: PNG writer
|   |   +-- JpegImageWriter.h               # NEW: JPEG writer
|   |   +-- ltic_api.h, lti_version.h
|   +-- lib/                                # Link libraries + DLLs
|   |   +-- lti_dsdk.lib, lti_dsdk_10.0.dll
|   |   +-- lti_dsdk_cdll.lib, lti_dsdk_cdll_10.0.dll
|   +-- doc/                                # Documentation
|   |   +-- UserManual/, ReferenceManual/
|   |   +-- CHANGES.txt, ISSUES.txt
|   +-- examples/                           # Raster-specific examples + WPFViewer
|   +-- 3rd-party/share/                    # GDAL data, PROJ database
|   +-- LICENSE.pdf, Copyrights_*.pdf
+-- Lidar_DSDK/                             # LiDAR Decode SDK
    +-- bin/                                # lidarinfo.exe, lidardecode.exe
    |   +-- laslib.dll                      # LAS file support (with LASzip 3.4.4)
    |   +-- pdalcpp.dll, pdal_util.dll      # PDAL libraries
    |   +-- proj_9.dll, geos.dll, geos_c.dll, tbb.dll
    +-- include/lidar/                      # LiDAR C++ headers
```

```
+-- lib/                              # lti_lidar_dsdk.lib, lti_lidar_dsdk_1.1.dll
+-- doc/                              # User Manual, Reference Manual, CHANGES
+-- examples/                         # LiDAR examples
+-- contributions/                    # Community code (GeoTIFF, liblas, SWIG)
+-- LICENSE.pdf, Copyrights_*.pdf
```

8. Migration Notes (from 9.5.5 to 10.0.1)

Step-by-step guide for upgrading existing applications:

8.1 Recompile Required

All applications must be recompiled against the new SDK 10.0.1 headers and link libraries. Internal data structures and class layouts have changed. Update include paths to Raster_DSDK/include/ and link against new lti_dsdk.lib.

8.2 DLL Update

Replace lti_dsdk_9.5.dll with lti_dsdk_10.0.dll in your deployment. If using the C API, also replace lti_dsdk_cdll_9.5.dll with lti_dsdk_cdll_10.0.dll. Stub library names (.lib) do not change.

8.3 New Runtime Dependencies

DLL	Required?	Purpose
lti_dsdk_10.0.dll	Always	Core MrSID SDK
tbb.dll	Always	Multi-threaded decoding
gdal.dll	Always	Raster I/O operations
geos.dll	Always	Geometry operations (used by GDAL)
geos_c.dll	Always	C API for GEOS
proj_9.dll	Always	Coordinate projections
libraw.dll	If using RAW images	Camera RAW file decoding
NCSEcw.dll	If using ECW images	ECW file decoding

Also ensure 3rd-party/share/proj/ with PROJ database files is accessible (set PROJ_DATA env var if not in default location).

8.4 Compiler Upgrade

Minimum supported Windows compiler is now Visual Studio 2019 (VC16.0). VS 2013/2015/2017 users must upgrade. SDK ships pre-built libraries for VC16.0, VC17.0, and VC18.0.

8.5 Linux Upgrade

Minimum GCC is now 11.3.1 on RHEL 9. RHEL 6.x and 7.x with GCC 4.8.2 or 5.3.1 no longer supported. Migrate to RHEL 9 (or CentOS Stream 9, Rocky Linux 9, Ubuntu 22.04/24.04) with GCC 11.3.1 or 13.2.1.

8.6 macOS Upgrade

Minimum macOS development is now Clang 14.0 (Xcode 14) on macOS 13 (Ventura). macOS 10.x-11.x no longer supported. SDK builds as Universal binary supporting Intel and Apple Silicon.

9. Multi-Threading Support

The MrSID SDK uses Intel Threading Building Blocks (TBB) to automatically parallelize decode operations on multi-core processors.

How it works:

- By default, SDK creates worker threads equal to the number of CPU cores
- MrSID (MG3, MG4) and JPEG 2000 decoding operations are automatically parallelized
- tbb.dll must be distributed with your application

Controlling thread count:

```
MrSIDImageReader reader;
reader.initialize(fileSpec);
reader.setMaxWorkerThreads(4);  // Limit to 4 threads
```

Important threading limitation:

The MrSID SDK is neither reentrant nor thread-safe. Your application must ensure that methods of any given SDK object are never called simultaneously from multiple application threads. For parallel decoding, create separate reader instances per thread.

10. Known Issues

Known limitations and issues in SDK 10.0.1, with workarounds where applicable.

10.1 Thread Safety

The MrSID SDK for Raster is neither reentrant nor thread-safe. Use successfully in multithreaded apps only if methods of any given object are never called simultaneously from different threads.

10.2 JPEG 2000 Reader Limitations

J2KImageReader does not support all Part 1 JP2 encodings:

- Lookup tables (palettized images) - not supported
- Advanced colorspaces - some unusual JP2 colorspace definitions may not display correctly
- Mixed datatypes - all bands must have same data type
- Mixed sampling rates - all components must have same dimensions

10.3 JPEG 2000 Writer Optimization

The JPEG 2000 writer does not support rate-distortion optimization. May result in slightly larger file sizes than optimized encoders at the same quality level.

10.4 Windows DLL Heap Allocation

Known crashing condition when allocating/deallocating SDK objects on the heap using new/delete. Occurs because DLL and application may use different C++ runtime heaps.

Workaround:

Use SDK objects on the stack. If heap allocation is necessary, use the template wrapper pattern shown in ImageInfo.cpp and DerivedImageReader.cpp.

Reference: Microsoft KB122675

10.5 Foreign Language Filename Handling

CLI applications may handle filenames incorrectly on some foreign-language systems due to locale issues.

Workaround:

Set environment variable LT_NO_SETLOCALE=true. App will print "[setlocale() not used]" at startup. Affects only the demo CLI applications, not the SDK libraries.

10.6 Data Type Support

The SDK supports 8-bit and 16-bit samples (signed and unsigned) and 32-bit floating-point. Up to 512 bands per image, but all bands must be the same data type and dimensions.