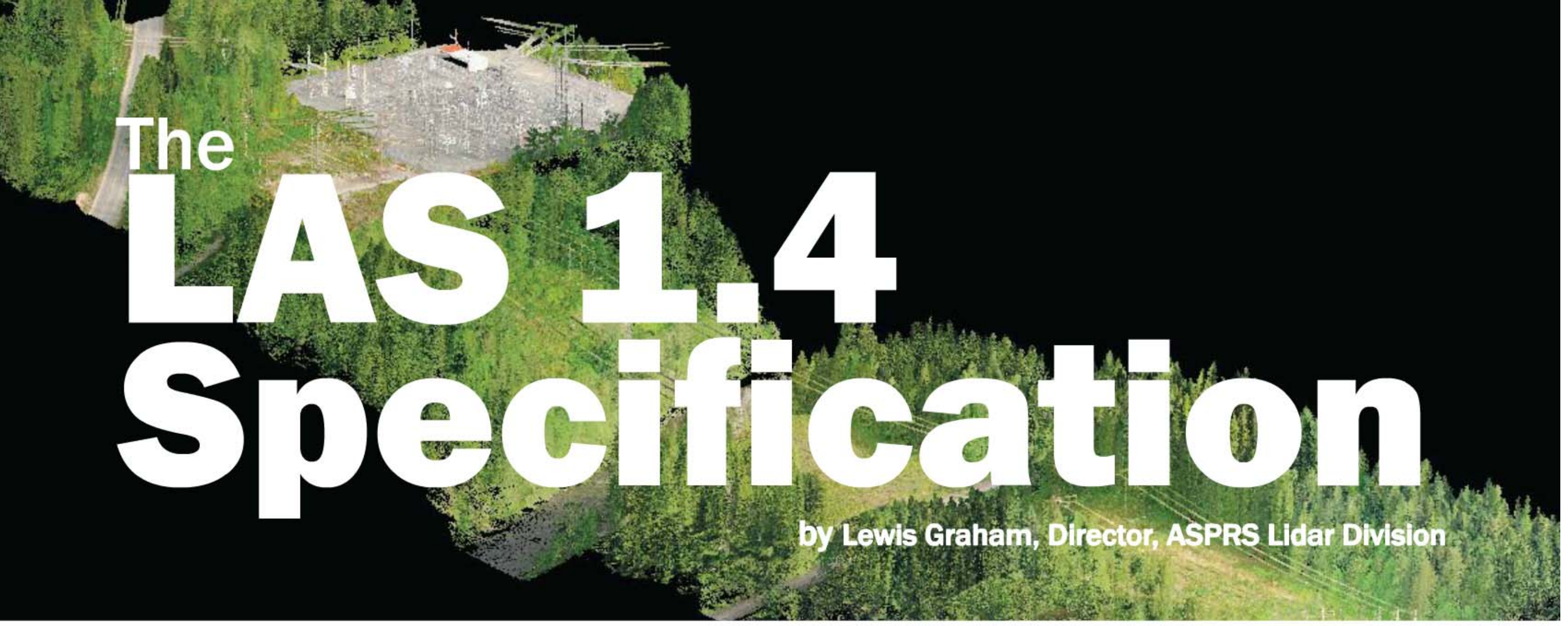


The LAS 1.4 Specification

by Lewis Graham, Director, ASPRS Lidar Division

Introduction

LAS is a specification for a *point cloud* file format. It is primarily used for transmitting laser point cloud data (Lidar) but can be used for any general 2D or 3D point oriented encoding.

This article provides an overview of the latest version of the specification, LAS 1.4, as well as a history of the LAS format. This article is an update of a previous *PE&RS* article concerning the LAS Lidar data specification and I have not changed the prior article in areas not related to the new version. Generally I will use the pronoun "I" when something is primarily my personal opinion and "we" when the work was consensus work of the LAS Working Group (LWG). The LWG is a working group within the ASPRS Lidar Division. It comprises a group of representatives of lidar hardware and software manufacturers, as well as several representatives of end-use agencies.

The introduction of a specification file format (LAS 1.0) for kinematic lidar data in 2002 has been very successful as evidenced by the fact that all major software vendors have adopted the specification and customers are routinely requiring that lidar data be delivered according to this specification. As with any specification, usage has made it apparent that changes need to be made both due to omissions in the original design as well as continuing maturity of the lidar industry. These updates have been occurring on an approximate two-year cycle.

At the ASPRS conference in Herndon, Virginia in November 2011, the ASPRS Board of Directors approved an update to the lidar file specification, officially making LAS 1.4 the new specification. I thought it would be worthwhile to explain this evolution of the LAS specification both from a practical point of view (what does one do to ensure adherence to the specification) and from a philosophical point of view (why was a particular change made and what are the underlying assumptions about data processing that may have necessitated a change).

Highlights of LAS 1.4

The update to LAS that resulted in the creation of version 1.4 was primarily motivated by the need to extend the number of possible classes from the 32 available in LAS 1.3 to 256 (ironically, LAS 1.0 did support 256 classes but this was changed in the update to LAS 1.1).

Of course, any time one starts updating data formats, other features have to be added! The summary of major additions to LAS 1.4 is:

- Extension of offsets and field sizes to support full 64 bit addressing
- Support for up to 15 returns per outgoing pulse
- Extension of the Point Class field to support 256 classes
- Definition of several new ASPRS defined classes
- Extension of the Scan Angle field to 2 bytes to support finer angle resolution
- Addition of a Sensor Channel bit field to support mobile mapping systems
- Addition of Well Known Text (WKT) definitions for Coordinate Reference Systems
- Addition of an Overlap bit to allow indicating pulses in the overlap region while maintaining the class definition
- Addition of an (optional) Extra Byte Variable Length Record to describe "extra bytes" stored with each point

"The update to LAS that resulted in the creation of version 1.4 was primarily motivated by the need to extend the number of possible classes from the 32 available in LAS 1.3 to 256."

Specifying LAS 1.4

Throughout this paper I will go in to some detail concerning the major points of the LAS 1.4 specification. This section provides information for those folks who just want to know what they should be specifying when contracting for lidar data.

The most critical features of LAS 1.4 with which most users should be concerned are support for 256 classes, an Overlap region indicator bit and specification of Coordinate Reference Systems (CRS) by the Open Geospatial Consortium method of Well Known Text (WKT). Since "class" and "Overlap" are per-point attributes, it is necessary to use a new Point Data Record Format (PDRF) to specify these

continued on page 94

continued from page 93

Table 1. The New Point Data Record Formats (PDRF) of LAS 1.4

PDRF (LAS 1.4 Only)	Major Features
6	Base type
7	Red, Green, Blue (3 channel) colorization support
8	Red, Green, Blue, NIR[1] (4 channel) colorization support
9	Waveform Packet Support
10	Red, Green, Blue, NIR (4 channel) colorization and Waveform Packet support

new features. LAS 1.4 adds five new formats (PDRF 6 through PDRF 10) to the specification that all include the new CRS encoding, extended classes and designation bits. The major differences between these five new formats are delineated in Table 1.

As is evident from Table 1, most new deliveries will be in PDRF 6.

LAS 1.4 *requires* that the Coordinate Reference System (CRS) encoding for PDRF 6-10 be WKT as opposed to the old technique of GeoTIFF packets. Thus, you are assured by specifying record types 6-10 that your data will have proper CRS encoding. I feel that this is critical for archived data since GeoTIFF (the old encoding standard) has limited support for describing the vertical components of CRS and limited support for newly emerging CRS definitions.

I believe that it is highly likely that the United States Geological Survey (USGS) will require point data record formats that support the designation of overlap via the new Overlap bit supported in PDRFs 6-10 and that this requirement will accelerate the move from prior versions of LAS to LAS 1.4.

Emulating Prior LAS Versions

During the specification process for LAS 1.4, there was a strong desire from several developers of open source software to create fields in the LAS 1.4 specification that might make it possible for a prior version of a LAS reader to ingest a LAS 1.4 file, if that file met certain conditions. These conditions are:

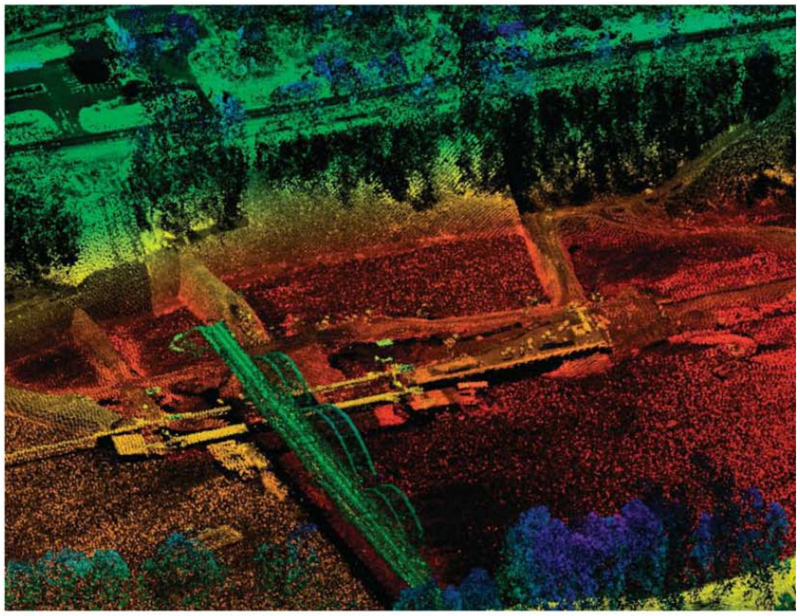
- The offset to data pointers must remain at or below 32 bit limitations
- Point Data Record Format must be one of the "legacy" formats (0 - 5)
- The Coordinate Reference System (CRS) encoding must be GeoTIFF (as opposed to the new method of WKT)
- The CRS must be encoded in a Variable Length Record as opposed to an Extended Variable Length Record
- The legacy reader software must ignore the LAS version number

I was personally opposed to this emulation mode because it introduces (if used) some unreliability to the LAS file and relies on a software defect (namely ignoring version number) to work at all. Unfortunately, the details (outside of the LAS Working Group) became obscure with some LAS consumers thinking that if this emulation mode were supported, their existing LAS software would be enabled for the new LAS 1.4 features. This, of course, is not possible since the major new features are encoded in the new point types (6-10) that simply were not present prior to LAS 1.4.

The publicity behind this emulation mode became (at least to me) surprisingly political and, at that point, a technical debate became impossible. In the interest of maintaining harmony amongst the very large group of LAS users, we (the LWG) added this emulation capability to the format.

I think this emulation feature should be used only for experimental purposes where the exact implementation is known, *a priori*. My recommendation is that you *never* specify a data delivery of LAS 1.0 - LAS 1.3 data within the context of a LAS 1.4 file. If you need LAS 1.0 to 1.3, simply require your vendor to deliver in the proper format.

General rule: Always specify Point Data Record Types of 6 and above and legacy fields set to zero.



The Original Specification

The original lidar binary file specification grew out of work done by EnerQuest Systems (whose software development work is now owned by Virtual Geomatics) in the late 1990s. EnerQuest had developed a proprietary point cloud binary format for laser scanned data (LASer) to make dealing with the generally unorganized data associated with lidar more efficient. Don Wicks, then president of EnerQuest, generously agreed to contribute the EnerQuest LAS work to the public domain. Leica Geosystems, Optech and Z/I Imaging (I was then with Z/I) joined the effort and the LAS 1.0 specification was born (the bulk of the implementation work was done by John Nipper, a software developer then at EnerQuest). About a year after this industry group defined the initial specification, the ASPRS agreed to take over ownership.

The LAS specification is a relatively compact binary encoding of point

[1]NIR - Near Infrared

location and point *attribute* data. Rather than store attributes in *referenced* records, the light-weight attribute data of LAS is stored in the same record as the point data. At the time of the definition of LAS 1.0, we did not try to anticipate how users would want to augment the format.

The Philosophy of Lidar Processing

In order to fully appreciate the design of the LAS format, it is necessary to understand the thought process that the original designers of LAS 1.0 had with respect to lidar data processing.

A time-of-flight lidar system emits a laser pulse and zero or more pulse returns[2] ("echoes[3]") are detected in the receiver. These return pulses may be reflected from nearly anything at all (e.g., birds, planes, tree leaves, buildings, ground, etc.). It is the task of the lidar data editor to *classify* these data with respect to their impinged surface. Thus the original LAS specification supported the idea of assigning a *classification code* to an echo. This classification attribute could be set to the surface type such as Building or Ground. Indeed, much of the software challenge in lidar processing has been a drive toward developing various algorithms that can automate this processing (so-called lidar *filtering* algorithms, an unfortunate misnomer).

The original philosophy behind lidar processing was that points can never be added to or deleted from a lidar data set after acquisition. Additionally, points should never be *moved* except under rigorously modeled adjustments such as *bore sight* calibration and project geometric correction (for errors introduced by inaccuracies in the positioning system). We (the original LAS design group) felt very strongly that simply *deleting* or moving data that do not fit a desired model is tantamount to fabricating data. To support the editing concept, certain classes were reserved to indicate data adjustments beyond reflectance classification. Thus if, for example, a high point (perhaps reflected from a bird) were to be omitted from the data set, it remained in the set but its classification code was set to WITHHELD.

It was soon evident that we needed to retain the original classification of points that were being withheld from the data set (e.g., we needed to designate a point as both "Ground" and "Withheld") and there was a need to be able to *synthetically* insert points into an existing lidar data set (for example, digitizing a synthetic lidar point to represent the attachment point of a transmission wire to an insulator). Thus, starting with LAS 1.1, the idea of modifier bits was added to the class designation.

Today the philosophy remains that once lidar data have been geocoded in the proprietary software supplied by the hardware vendor, they should only be geometrically moved in a system calibration and/or geometric correction process and that points should never be deleted.

Extending the Lidar Edit Philosophy

The LAS 1.4 specification incorporates features that allow more robust point marking, adhering to the philosophy that points should never be physically deleted or moved during editing. For example, if points are found to be reflecting from saw grass, they should not be collected up as a group in a point editor and literally *moved* to the ground. We argue that what has really happened is that a collection of points have been marked as *withheld* and a new collection of points *added* based (one would hope!) on an accurate editing surface such as ancillary stereo imagery. This type of point marking has been available in LAS since version 1.1 by supplying appropriate attribute fields in the point records. LAS 1.4 supports four attributes related to point editing (described later).

The LAS specification is being used quite frequently as a general format for point cloud data, regardless of its source. For example, I know of at least one company that generates LAS format data from the output of a stereo image correlator and then applies lidar processing tools to the task of editing these data. We very strongly support this use of the LAS specification and have, in fact, added some fields in the 1.4 release to expand support for facilitating these non-traditional uses (primarily the addition of a Near-Infrared channel for supporting 5 channel 3D image data derived from line imagers).

"The LAS 1.4 specification incorporates features that allow more robust point marking, adhering to the philosophy that points should never be physically deleted or moved during editing."

The General LAS Layout

The format contains binary data consisting of a public header block, any number of (optional) Variable Length Records (VLRs), the Point Data Records, and any number of (optional) Extended Variable Length Records (EVLRs). This general layout is depicted in Table 2.

Table 2. LAS 1.4 Format Definition

Public Header Block
Variable Length Records (VLR)
Point Data Records
Extended Variable Length Records (EVLR)

The public header block contains generic data such as point counts and point data bounds. Interestingly, it does not contain the definition of the Coordinate Reference System (CRS).

The Variable Length Records (VLRs) contain variable types of data including projection information, metadata, waveform packet information, and user application data. They are limited to a maximum data payload size of 65,535 bytes. The continued existence of VLRs in LAS 1.4 is one of the compromise results of supporting emulation mode (discussed earlier). Software writers of LAS 1.4 files where emulation is not needed (the recommended approach) can encode all variable information into EVLRs.

The Extended Variable Length Records (EVRLs) allow a larger payload than do the VLRs and have the advantage that they can be appended to the end of a LAS file. This allows adding, for example, projection information to a LAS file without having to rewrite the entire file.

A LAS file that contains point record types 4, 5, 9, or 10 could potentially contain one block of waveform data packets that is stored as the payload of any Extended Variable Length Record (EVLR). Unlike other EVLRs, the Waveform Data Packets (if stored internally to the file) have the offset to the storage header contained within the Public Header Block ("Start of Waveform Data Packet Record").

[2] and, effective with LAS 1.3, possibly wave packets
[3] I use echo and return interchangeably

continued on page 96

continued from page 95

Projects

In LAS 1.1, we defined the idea of a *Project*. A Project can be a lidar job as defined by a number of flights/drives over a project area or it could simply be a number of elevation files being used in a processing job not necessarily even related to lidar. A Project is uniquely identified by a Globally Unique Identifier (GUID). A GUID (sometimes alternatively referred to as a Universally Unique Identifier, UUID, or as a Uniform Resource Indicator, URI) is generated by a "GUID Generator" algorithm available within the Microsoft development systems or from open sources. A GUID is a 16 byte identifier that is unique across the domain of all possible GUIDs. The Project GUID existed in LAS 1.0 but we clarified its intended use in LAS 1.1.

It is a good idea to assign a Project ID at the time of creation of the project and to ensure that this same project ID is used for all files that are associated with the project.

File Source ID

The File Source ID field is used to uniquely identify *this* file within a Project. If the source of the file is a lidar flight/take[4] line, then this field is used as the flight line number. Thus inclusion of the File Source ID solves a major problem that existed in LAS 1.0; the restriction of flight line information to 256 lines. The File Source ID field supports 65,536 individual files within a project.

It is important to note that the LAS format can embody any sort of point cloud data in addition to lidar data. This means that a project could consist of a heterogeneous collection of a variety of elevation files from any number of sources. For example, we might construct an elevation project with some files originating from lidar flights, others from USGS elevation files, still others from photogrammetrically correlated data and so forth. Thus the concept of *Project* becomes generalized.

System ID

In keeping with the notion of generalizing the LAS specification to apply to any sort of point cloud data, System ID is used to provide information about the manner in which the LAS file was generated. For a lidar system, this field should be encoded with a system identifier such as "ALTM 1210." Similarly, this field is used to indicate the method by which a file was generated when the source is not a lidar sensor. The specification provides several predefined values such as "MERGE" and "EXTRACTION" and allows the user to provide any definition (within the limits of a 32 byte description). A common example within lidar processing would be the generation of a processing tile by merging the content of several flight lines. In this case the Project ID is set to the overall Project ID assigned to the project, a new and unique File Source ID is created and the System ID field is set to MERGE.

Point Data Records

LAS files are homogeneous with respect to point records. The record *type* is indicated by the Point Data Record Format (PDRF) in the Public Header Block. All points within a single file must be the same with the format being specified by the PDRF.

It is important to note that the LAS specification does not impose a point ordering scheme on a LAS file. Thus the points can (and very often do) appear in any order. For example, if point n is marked as "return 2 of 4" and point n+1 is marked as "return 3 of 4", there is no guarantee that these two points are associated with the same outgoing point (you would have to use GPS time to sort the returns, as noted later). I have seen a lot of software that erroneously assumes that LAS points will be sorted by spatial proximity, by return number or other attributes. This results in, for example, poor display performance when the data are, in fact, not in the assumed order.

LAS 1.4 added five new PDRFs (6-10) to the specification. The salient new features of these types are discussed in a later section. While LAS 1.4 continues to support PDRFs 0-5, I strongly recommend that you always specify the new point types 6-10. The primary reason for this recommendation is that merging point types 0-5 into files using point types 6-10 will require a type 6-10 PDRF (due to expanded fields). A second reason for this recommendation is that LAS *requires* Well Known Text (WKT) encoding for the Coordinate Reference System (CRS) of PDRF 6-10. WKT is a much more reliable CRS representation than GeoTIFF (in fact, there are a number of CRSs that simply cannot be represented in GeoTIFF). Remember, if you have a LAS file and you cannot determine its CRS, the data are useless for absolutely referenced data exploitation.

"It is critically important for programmers to understand that a point data record in LAS can contain fields beyond those defined by the PDRF."

It is critically important for programmers to understand that a point data record in LAS can contain fields beyond those defined by the PDRF. For example, PDRF 0 defines 20 bytes of specified information. If desired, a software implementer can add additional bytes to the end of this point type (for example, perhaps to encode an accuracy estimator for Z). Thus, one must always refer to the Point Data Record Length field in the Public Header Block to determine the correct number of bytes for each point record.

LAS 1.4 (thanks to a contribution from Dr. Martin Isenburg) adds a way to formally define extensions of the standard PDRFs via a mechanism called "Extra Bytes." While nothing in the LAS specification prevents a developer from adding their own PDRF (say PDRF=21), no other software would be able to read the point data. By using the Extra Bytes facility, software that does not implement Extra Bytes will still continue to correctly read the "base data" of the point record. Those who participated in the review of the canceled LAS 2.0 effort will recognize that this was the total representation method of points in that format.

GPS Time

Most Point Data Record Formats (PDRF) support a field for the time of the pulse. This time is encoded as either GPS Week-Second time or Standard GPS time as indicated by the Global Encoding field in the Public Header Block. We recommend that all procurers of lidar data *require* that the GPS time be encoded as Standard time. This is because GPS Week Time does not allow one to determine the absolute time that a pulse was transmitted.

[4] In Mobile Mapping, a single span of data collection is often referred to as a data "take.

Although not clearly described in the LAS specification (an oversight that I just recognized while writing this article), the GPS time is the time when the pulse was *emitted* by the lidar sensor. Thus all returns (see next section) of a multiple return pulse will have the same GPS time.

When we first defined the original LAS specification, we felt that GPS time, on a per pulse basis, would be used only for certain "close to the sensor" operations. However, many algorithms for data exploitation, particularly in the area of mobile mapping, heavily rely on GPS time. Additionally, the GPS time is currently the only mechanism for correctly correlating a set of multiple returns. For this reason, GPS time was included as a field in all of the new PDRFs of LAS 1.4.

Returns

LAS is designed for discrete point data (that can optionally support wave form packets). While it can be used for any 2D or 3D point (or N dimensional with the Extra Bytes facility), the primary emphasis is on kinematic, discrete return, time of flight systems lidar. All modern lidar systems of this type are capable of multiple returns per outgoing pulse. A diagram of a potential multiple return scenario is illustrated for a five return scenario in Figure 1.

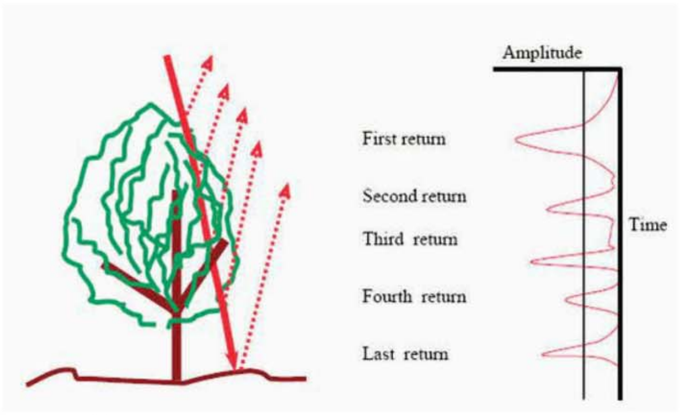


Figure 1. A five return system.

LAS 1.4 supports up to 15 discrete returns per outgoing pulse. The count of the number of returns per pulse is contained in the field Number of Points by Return in the Public Header Block (not the Legacy Number of Points by Return - this is used for encoding a LAS 1.1 - 1.3 payload within a LAS 1.4 file body - a legacy implementation that can support only five returns). Note that this field is an array that is a histogram of the number of returns for this file. It does not represent the physical capabilities of the system(s) that was used to acquire the data.

The return position of a point within a multiple return source pulse is such an important factor in data analysis that this position is encoded for all Point Data Record Types. PDRF 0-5 (the legacy types from prior LAS versions) support up to five returns, whereas the new types (6-10) support up to 15.

All returns from a multiple return outgoing pulse will have the same GPS time stamp. This fact is used for correlating the returns of a multiple return pulse. It is entirely possible that returns from the same outgoing pulse will span more than one LAS file. This generally occurs when data are segregated into tiling schemes and different returns of the same outgoing pulse span a tile boundary.

Classification

The LAS 1.0 specification devoted an 8 bit field to CLASS. The purpose of the field was to allow a point to be tagged as to its reflectance surface (e.g., Ground, Low Vegetation and etc.). A second use for CLASS was to mark points that were special in some sense, such as WITHHELD or MODEL KEYPOINT. Other than reserve space in each point record for the field, the value stored in this field was entirely up to the user.

When we initially devised the LAS 1.0 specification, we did not really view the specification as a format that would be conveyed to the ultimate end-users of lidar data. It was intended to be a transport protocol between makers of lidar sensors and developers of lidar processing tools. However (and this is a very good thing), LAS rapidly became the de facto specification for all phases of lidar data processing and delivery. This holistic use necessitates specification of the values for the CLASS field. In other words, we all have to agree on which numeric value will signify the various classes.

A second shortcoming of the original CLASS field in the LAS 1.0 specification was that one could not simultaneously maintain a class attribute and a treatment attribute. For example, the field would not support the idea of a point being both Ground and a Model Key Point. Encoding a point as a Key Point destroyed its reflectance categorization.

To solve these two problems, the 8 bit CLASS field of LAS 1.0 was split into two parts; a 5 bit CLASS section and three bits used as Flags (see Figure 2). The classes ranged from zero to 31.

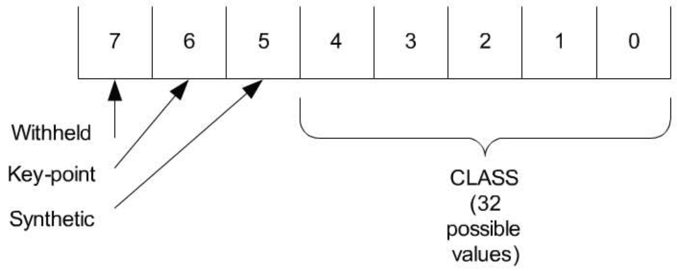


Figure 2. The Classification Field of LAS 1.1 - 1.3.

When we made this transition to LAS 1.1, there were typically only a handful of classes defined (usually less than ten). We therefore thought that 32 classes (0-31) would provide plenty of room for any classification scheme. Unfortunately, with the recent widespread use of lidar for mobile mapping applications and transmission line coding, the need for additional room for class definitions has become critical.

For this reason, LAS 1.4, for the new PDRFs 6-10, separates CLASS back in to a dedicated 8 byte field. Class IDs 0 through 63 are reserved for ASPRS use, leaving classes 64 through 255 available for definition by users.

Using ASPRS defined classes allows any software application to correctly display the meaning of the class without the software vendors needing to communicate this information. Now one could argue that this could be more elegantly communicated by simply encoding the class definition table in a Variable Length Record (in fact, this is what we did in the original version of LAS) but this creates a mapping problem when merging files.

The class definitions for PDRFs 6-10 are listed in Table 3.

Two very important changes have occurred in this classification table; the removal of Model Key Point (class 8) and Overlap Point (class 12) as classifications. To avoid confusion with the classification of the old

continued on page 98

continued from page 97

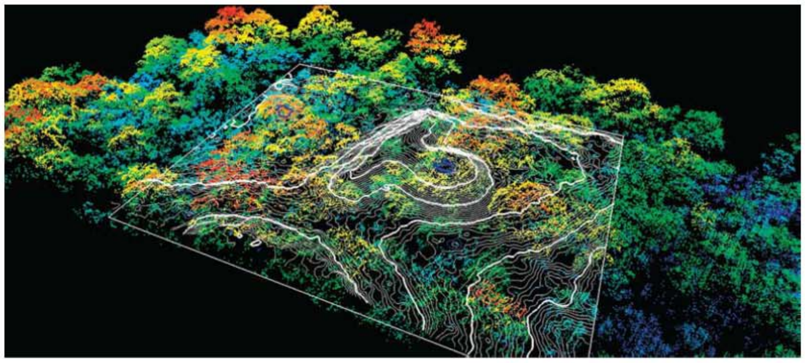
Table 3. ASPRS Standard Lidar Point Classes (Point Data Record Formats 6-10)

Classification Value	Meaning
0	Created, never classified
1	Unclassified[5]
2	Ground
3	Low Vegetation
4	Medium Vegetation
5	High Vegetation
6	Building
7	Low Point (noise)
8	Reserved
9	Water
10	Rail
11	Road Surface
12	Reserved
13	Wire – Guard (Shield)
14	Wire – Conductor (Phase)
15	Transmission Tower
16	Wire-structure Connector (e.g., Insulator)
17	Bridge Deck
18	High Noise
19-63	Reserved
64-255	User definable

point data record formats 0-5, we have set these two classes to Reserved. Indicating Key Points and Overlap with PDRF 6-10 is performed by attribute bits (see next section).

Note that we have added a few new ASPRS standard class codes such as Rail and Wire types. We anticipate that the full 0-63 class slots reserved for ASPRS will be defined over time as user needs mature.

Obviously another good reason to only accept lidar data in formats 6-10 is to ensure that the classification encoding is homogeneous across all of your data.



[5] We are using both 0 and 1 as Unclassified to maintain compatibility with current popular classification software such as TerraScan. We extend the idea of classification value 1 to include cases in which data have been subjected to a classification algorithm but emerged in an undefined state. For example, data with class 0 is sent through an algorithm to detect man-made structures – points that emerge without having been assigned as belonging to structures could be remapped from class 0 to class 1.

Point Attributes

Point attribute bits were first expanded to include classification flags in LAS version 1.1. In LAS 1.4 (for PDRF 6-10) an additional flag was added to indicate points in the overlap region. The Point Attributes of PDRF 6-10 are listed in Table 4. The first 4 bits are devoted to classification flags.

Table 4. Point Attributes for PDRF 6-10

Classification Flags	4 bits (bits 0 - 3)	4 bits
Scanner Channel	2 bits (bits 4 - 5)	2 bits
Scan Direction Flag	1 bit (bit 6)	1 bit
Edge of Flight Line	1 bit (bit 7)	1 bit

The first flag bit is the *Synthetic* flag. If this flag is set (value 1), it indicates that the associated point was not collected by means of the method indicated in the System ID field for the file but was added via some other mechanism. The most common use of this field is moving a lidar point. We advocate a philosophy that once “bore sighting” and project geometric correction (the equivalent of photogrammetric block bundle adjustment) have been performed, individual lidar points cannot have their X, Y or Z values adjusted. Thus if one were to move a collection of lidar points observed in an editing session to be reflecting from within low vegetation to the ground, the operation would actually proceed as a deletion of the point within the low vegetation (the deletion is actually setting the WITHHELD flag bit, not removing the point from the file) and the addition of new ground points. These newly added ground points would have their CLASS value set to GROUND and their SYNTHETIC attribute flag bit set to True (a value of 1). The Synthetic flag should be used for any *additive* operation such as digitizing points in lidargrammetry, adding a point to indicate a power line connection and so forth.

The second flag is KEY POINT. This allows the marking of a point as one that is to be kept for elevation file formation (perhaps as a contributor to a contour generation routine) while maintaining its class attribute. Thus the point could maintain a class of Water and also be flagged as a Key Point. Key point algorithms thin data to the minimum points necessary to maintain a specified accuracy criteria. A typical approach is to create a Triangulated Irregular Network (TIN) using all of the points (of the desired class such as ground) and then iteratively remove points that do not cause the regenerated TIN to violate a specified error tolerance.

The third flag is the WITHHELD flag. Points should never be deleted from an LAS file but rather flagged as deleted. Thus, for example, in the removal of water points, a point could be classified as Water (CLASS = 9) and then have the WITHHELD bit set to 1 to indicate that the point should be omitted from further modeling.

New to LAS 1.4 is the fourth flag, OVERLAP. It is frequently desired to thin data in the overlap region of lidar flight lines (or, in the case of mobile, "takes"). This operation is used to either improve the statistical accuracy of the lidar data or to simply thin the multiple density data in these regions. Using a Class to indicate overlap (the technique prior to LAS 1.4) caused problems in that one loses the actual classification of the point (e.g., Ground). This leads to problems in analyzing accuracy in the overlap region. What exactly comprises points that should be designated as Overlap will be determined (as it is now) by the procurer of the lidar data. For example, USGS has an extensive description of how points should be designated as Overlap.

Also new to LAS 1.4 is the ability to designate a Scanner Channel. The intended use is for mobile mapping systems with multiple laser scanners. Two bits are used for this indicator and thus one of four scanners can be designated. For a single scanner system, the channel must be zero.

The final two bits of the attribute field are Scan Direction and Edge of Flight Line. Unfortunately, even though these are mandatory fields in the LAS point data, I often see them omitted. Scan Direction is defined for rotating mirror scanners and is defined to be positive for counter-clockwise rotation if the flight direction were coming out of your eye (looking at the back of the aircraft as it moves along its track). I personally do not know how this parameter might be used in data processing. The last flag, edge of flight line, is far more important for non-360° scanners. It can be used to quickly compute a graphic for the edge of a lidar flight (see Figure 3).

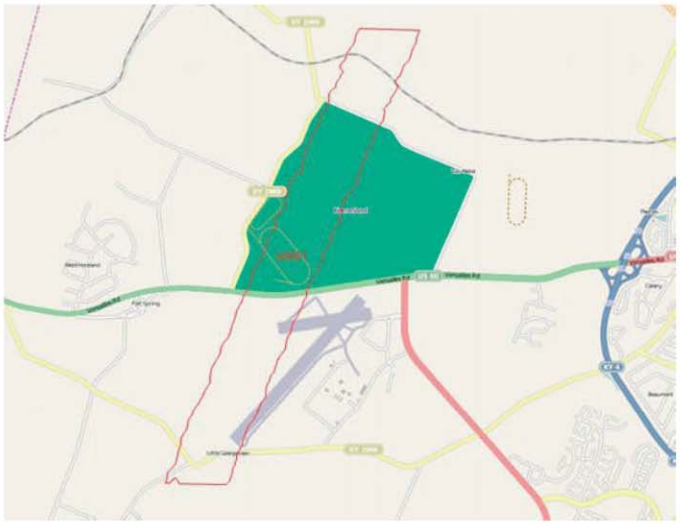


Figure 3. Edge of Lidar Flight Line.

User Data

Each Point Data Record Format (PDRF) supports 1 byte of User Data. This is a data field that can be used for any purpose by processing software. There is no guarantee that data stored in this field will be preserved from one application software package to another. In fact, many software packages use this field for a scratch pad area during processing. Clients ordering lidar data may specify that this field be set to zero prior to data delivery.

Scan Angle

The Scan Angle, for PDRF 6-10, has been expanded to two bytes for better scan angle resolution. It is a signed short that represents the rotational position of the emitted laser pulse with respect to the vertical of the coordinate system of the data. Down in the data coordinate system is the 0.0 position. Each increment represents 0.006 degrees. Counter- clockwise rotation, as viewed from the rear of the sensor, facing in the along-track (positive trajectory) direction, is positive. The maximum value in the positive sense is 30,000 (180 degrees which is up in the coordinate system of the data). The maximum value in the negative direction is -30,000 which is also directly up. Note that scan angle can now, under this new definition, be used for 360° rotational sensors such as those used on mobile mapping systems.

Intensity, RGB, NIR Channels

The intensity value is the integer representation of the pulse return magnitude. This value is optional and system specific. However, it should always be included if available.

Red, Green, Blue (RGB) and near infrared (NIR) can also be stored with certain point formats. These channels are intended for storing values from colorization of the points by one or more auxiliary optical sensors (cameras). For example, colorizing lidar data with RGB reflectance data from a 360° camera is a common operation in mobile mapping.

This data, when included, is always normalized to a 16 bit, unsigned value by multiplying the value by 65,536/(intensity dynamic range of the sensor). For example, if the dynamic range of the sensor is 10 bits, the scaling value would be (65,536/1,024). This normalization is required to ensure that data from different sensors can be correctly merged. If intensity and/or color data are not included, these values must be set to zero

LAS is more and more frequently being used to represent point clouds from non-laser derived data such as 3D points from image correlators. By using Point Data Record Format 8, up to five channels of information (using the available I, R, G, B, NIR fields) can be stored.

"It is a good idea to assign a Project ID at the time of creation of the project and to ensure that this same project ID is used for all files that are associated with the project."

Point Source ID

The Point Source ID field (2 bytes) is used to identify the source file in which a point was originally created within a Project data set. For example, if a project contained a source file of lidar points that originated from a flight line and that flight line had been assigned number 77, all points within that original file would keep their Point Source ID of 77 throughout all processing regardless of how many files through which that point might have propagated. This feature allows mapping back to the original source of a point even when an LAS file has resulted from composite operations such as merge and extract. The most common example is that of a processing tile formed by merging numerous flight lines into the tile. By using the Point Source ID, each point can be uniquely identified as to its original source. Thus, this is the field that is used in software when performing operations such as "color by flight line".

In general, once a Point Source ID has been assigned to a point, it should not change. It should also be noted that Point Source ID is not meaningful outside the context of a single project (e.g., if points from two or more projects are blended, they cannot be segregated using Point Source ID). However, if you have control over all projects that will be merged, you can pre-plan flight lines such that back-tracking and unique flight lines are maintained.

You will note that for a pure lidar project (all data sources originated from lidar flight lines), the Point Source IDs must all map back to original flight line numbers except for SYNTHETIC points. This is illustrated in Figure 4. In this simple example, we have two flight lines ("strips") that have been labeled with File Source IDs of 1 and 2. The points from these two

continued on page 100

continued from page 99

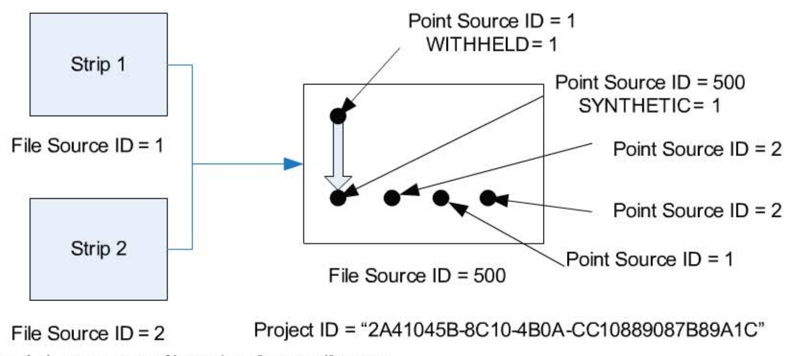


Figure 4. A composite file with a "moved" point.

flight lines have been MERGED into a file that has been assigned a File Source ID of 500. Within this merged file, the left-most point has been "moved" down. This is accomplished by setting the WITHHELD bit of the original point to 1 (TRUE) and then creating a new point in the desired location. Since the new point was created within Source File ID 500, it is assigned a Point Source ID of 500. In addition, this new point has its SYNTHETIC flag set to 1 (TRUE). Notice that in the merged file, each individual point can be traced back to its origin.

Waveform Data

LAS 1.3 introduced the storage of "waveform" data to the specification. This capability continues, unchanged in LAS 1.4. A waveform is essentially a digitized representation of the return intensity of a laser pulse with respect to time. The waveform nature of the return was indicated in Figure 1.

Primarily for convenience and to reduce data storage requirements, earlier lidar systems analyzed waveform return data in front-end electronics and declared the potential location of an impact with an object space object. As digitization hardware improved and software algorithms became more sophisticated, there was a drive from both hardware vendors and data processors (particularly the academic community) to make waveform data available to processing software. Hence, this capability was introduced in LAS 1.3.

Rather than store the entire waveform, waveform packets spanning "interesting" events are stored. An interesting event could be anything the hardware vendor decides but is typically amplitude above the noise level within the region of interest range (e.g., if the nominal flying height above ground is 2,000 meters, events 200 meters from the aircraft would not normally be collected).

To allow dual use of the data (e.g., point mode and waveform mode), each waveform packet is indexed by at least one point (see Figure 5). This allows software based on discrete returns to continue to function. There is a misconception that this encoding scheme does not allow "returnless" waveform packets to be encoded. However, this is easily achieved by the hardware vendor (in post-processing software) tagging desired "returnless" waveform packets with a synthetic return.

An inspection of the encoding of the waveform packet scheme reveals that we constructed a hybrid approach to storing waveform information. A return pulse can:

- not point to a waveform packet
- be the sole pointer into a location within a waveform
- Point into a waveform packet pointed to by one or more other returns (first example of Figure 5).

A single LAS file can encode up to 255 waveform packets schemes. These schemes are encoded within Wave Packet Descriptors (stored in VLRs). A waveform packet descriptor contains the fields listed in Table 5. As of version 1.4, compression of the waveform data is not yet supported. The temporal sample spacing is in picoseconds, providing a maximum resolution of 0.03 cm.

Table 5. Waveform Packet Descriptor User Defined Record

Item	Format	Size
Bits per sample	unsigned char	1 byte
Waveform compression type	unsigned char	1 byte
Number of samples	unsigned long	4 bytes
Temporal Sample Spacing	unsigned long	4 bytes
Digitizer Gain	double	8 bytes
Digitizer Offset	double	8 bytes

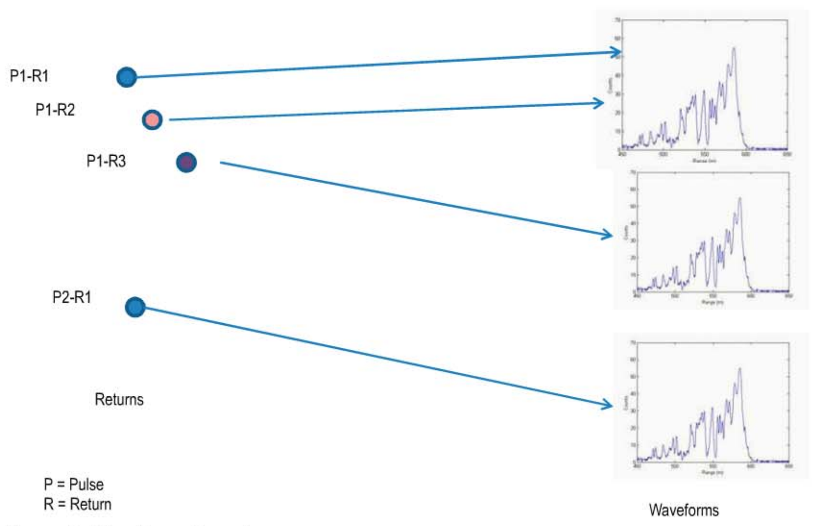


Figure 5. Waveform Encoding.

Each point indexing a waveform packet includes first order coefficients for "walking the wave." This allows you to shift the location of a declared point using a first order approximation. The common operation here would be to analyze a wave packet for the location where you would like to declare a return by shifting the default return of a sensor biased toward high returns (for example, in high grass) to the first rise in amplitude. You would compute the X, Y and Z values for your new return by starting at the closest sensor declared return on the wave packet and using the first order X(t), Y(t), Z(t) coefficients to position the new pulse. Finally, this new pulse would be marked as *synthetic*.

The exploitation of waveform data in a meaningful way is just starting to appear in algorithms of commercial software (e.g., TerraScan from Terrasolid). As time goes on, we expect this to become more prevalent.

There has been some criticism of the waveform storage of LAS 1.3/LAS 1.4. These criticisms range from too much data being stored to too little data being represented. However, this is a bit of the cart before the horse. As commercial software vendors add meaningful processing algorithms, we will get a better handle on what actually needs to be exchanged. Additionally, a full understanding of the nature of the problem is necessary prior to making changes. For example, today we store wave walking coefficients on a per return basis. Would a single set of coefficients suffice? A detailed analysis of the curvature of the pulse (caused primarily by atmospheric refraction) would be required prior to making this change.

(Extended) Variable Length Records

The LAS format has always provided a facility for users to store metadata that is "private" in nature (meaning that other software will simply ignore these records). With the exception of waveform data, prior to LAS 1.4, these Variable Length Records (VLR) have occurred (in the file) prior to the Point Data Records. A VLR can hold a payload of up to 65,535 bytes (because the data packet size is limited to 16 bit offsets). To legitimately use a VLR, a user must request a VLR "Key" from the LAS Working Group (www.lasformat.org). This Key, along with a Record ID number, allows one to uniquely identify a VLR. Note that a registration key is only needed when one intends to add a custom metadata packet to a LAS file (which, for most users, would be quite rare).

For a number of uses, the length of the VLRs proved to be a serious limitation (for example, when using this facility for storing spatial index information). Thus in LAS 1.4, we made the Extended Variable Length Record available for general use (prior to LAS 1.4, an EVLR was used only for Waveform Packet information). EVLRs are intended to be placed after the point data records. An EVLR payload size is represented by a *long long* (64 bits) so this should definitely solve size limitations!

Unfortunately, VLRs/EVLRs are designed to be sequentially placed in a LAS file (rather than daisy chained). Thus all VLRs should be sequentially placed ahead of the point data records and all EVLRs should be sequentially placed after the point data records.

There are several VLR/EVLRs reserved for LAS Specification use. An example is the encoding of coordinate reference system (CRS) information. For the legacy Point Data Record Formats (PDRF) 0-5, this CRS encoding can be either GeoTIFF or (preferred) Well Known Text (WKT). For PDRF 6-10, the CRS encoding must be WKT. An important note to software developers is that the LAS defined records can be either stored as VLRs or as EVLRs. Thus, when searching a file for the CRS records, it is necessary to search the EVLRs if a CRS record was not found in the VLRs.

Extra Bytes

A new LAS Variable Length Record (VLR[6]) has been defined called "Extra Bytes." This VLR allows one to parse a Point Data Record that has been augmented with extra fields. As noted earlier in this article, a standard LAS Point Data Record can have data tacked on to the end of each point data record. Prior to LAS 1.4, a writing software would have to communicate to a reading software the content of this extra data using some previously agreed private scheme. The Extra Bytes metadata provides the ability to specify the followiong for each field:

- Field Name
- Description
- Data Type (all standard type as well a several array types)
- A No Data signature
- Minimum permissible value
- Maximum permissible value
- Scale
- Offset

Flags are included to indicate if various attributes apply to a specific field (such as minimum and maximum values.

This facility will prove quite useful for those who wish to make use of "off the shelf" lidar software that ingests LAS format data but need to augment that data for other uses. For example, Extra Bytes will provide a canonical method of added "Sigma Z" to a point data record.

It is the responsibility of LAS software writers/readers to properly index point data records in LAS files. A software writer must *always* use the Point Data Record Length field in the Public Header Block to determine the size of the point records in a given LAS file. Simply counting bytes for a particular PDRF will result in an indexing error if extra bytes have been appended to a record. Software that is rewriting a LAS file containing Extra Bytes should simply copy these bytes to preserve the records.

It should be noted that while Extra Bytes is a tremendous improvement over no representation of augmented point data records, we do not anticipate that commercial software will attempt to build the semantic engines necessary to handle mixed mode Extra Bytes (that is, multiple files each with possibly different Extra Bytes metadata). Of course, this issue is certainly not unique to LAS!

"LAS 1.3 introduced the storage of "waveform" data to the specification. This capability continues, unchanged in LAS 1.4."

A Well Formed LAS 1.4 File

This section is primarily my opinion so it will all be written as first person. There is a bit of a difference of opinion among designers of LAS software as to how ambiguous conditions should be handled. Commercial software purveyors tend to focus on error handling, whereas the open source camp is more focused on attempting to continue to read the files. My observations, based on working the LAS format since inception and working with dozens of lidar processing companies, is that ambiguous conditions must be treated as fatal errors. This is primarily based on the observation that production managers usually do not want production technicians making decisions regarding potential error resolution. Nothing is quite as embarrassing as delivering a final lidar project in the wrong coordinate reference system! Thus, the following guidelines are recommended for software authors:

1. While LAS 1.4 will support the encoding of Point Data Record Formats (PDRF) 0-5 (the legacy formats) such that, under certain conditions (described in an earlier section) software that has not been updated to LAS 1.4 could read the file, I strongly recommend that this facility never be used for production-type scenarios. There is no need at all for using this facility since one can simply write out a file in one of the legacy formats, if compatibility with these prior formats is needed.

continued on page 102

[6] Note that I refer to these LAS Spec records as VLRs for convenience. They can actually be encoded as either VLRs or EVLRs.

continued from page 101

2. In light of 1, the following fields should always be set and maintained as zero:
   a. Legacy Number of Point Records
   b. Legacy Number of Points by Return.
3. Always use Well Known Text (WKT) encoding for the Coordinate Reference System (CRS). GeoTIFF has not been maintained in terms of new CRS and vertical representation is often not supported. Note that you will need to set the WKT bit of the Global Encoding field in the Public Header Block.
4. Add an Extra Bytes VLR when augmenting Point Data Records. This eliminates the problem of trying to figure out why your Point Data Record Format 0 is longer than 20 bytes!
5. Never store both a GeoTIFF and WKT VLR in the same file. There is very often not a one-to-one mapping from GeoTIFF to WKT and thus an intrinsic error exists.
6. Try to avoid using anything other than powers of 10 for the scale values in the Public Header Block. Floating point for X, Y, Z has been purposefully avoided in the LAS definition. Non-power of ten scaling makes this a pain to avoid when writing software.

"Clearly LAS has passed the "tipping point" and will, at least for the immediate future, remain the format of choice for time-dependent point cloud data."

Error Conditions

While we attempted to make the LAS 1.4 format unambiguous, we were not completely successful. Again, I will resort to first person in this section since what I say should be an error, others might say to take a different path. Remember, my focus is on rigorous error detection in lidar production environments. In any of the error conditions below, the software should, of course, provide an exact message as to the reason an error has been declared.

1. While I strongly recommend against using the *legacy* payload capability of LAS 1.4, one should treat a difference between a non-zero legacy field and its equivalent standard field as an error. For example, if Legacy Number of point records is not zero and is not the same value as Number of point records, an error should be declared.
2. Global Encoding bit WKT is set (non zero) but a valid WKT VLR does not exist (even if a GeoTIFF packet does exist).
3. PDRF is 6 or greater and the WKT bit is not set (this is explicitly pointed out as an invalid LAS file in the LAS 1.4 specification). This error must be declared even if a WKT VLR is found in the file.
4. Number of point records does not equal the sum over Number of points by return.
5. Return Number (in a Point Data Record) is zero.

In my opinion, any data inconsistencies within a LAS file should be treated as a "hard" error, meaning the reader will exit with an error rather than continue to process the file with a warning or, even worse, no message at all. After declaration of a hard error, the file should be fixed "off-line" and reintroduced to the production workflow only after correction. This opinion is based on my experience of seeing many embarrassing data deliveries caused by improperly formed LAS files (particularly in the area of coordinate reference system issues).

What is Next for LAS

When looking back at a prior article on LAS 1.1, I notice that we have now (at LAS 1.4) added most of the features that were in the "what's missing" section of that article (RGB, waveform, etc.).

I think now our (the LAS Working Group) focus needs to be on flexibility. The LAS format "hard codes" the various Point Data Record Formats (PDRF). While, with LAS 1.4, these formats can be extended with "Extra Bytes", this is still a mechanism that requires inside knowledge of the semantics of the data fields.

Several years ago, we proposed a more flexible LAS format during the design of LAS 2.0 (a version of the specification that was ultimately withdrawn). Here the approach was similar to the "Extra Bytes" facility of LAS 1.4 but applied to every field in the PDRF (similar to an XML encoding but binary). This type of encoding will eliminate the repetitive PDRF scheme and allow us to simply specify different formats based on a set of standard fields. It will also allow for different data types to be used for some of the standard fields, thus either expanding the fields (for example, allowing X, Y, Z to be specified as 64 bit ints) or contracting fields (allowing intensity to be encoded as an 8 bit field).

Perhaps the most needed addition at the moment is spatial indexing and pyramiding. Currently a number of venders (including my own companies) perform these performance enhancing additions via VLR encoding and/or auxiliary files. It will be very beneficial to standardize these facets so that a software package from vendor B can read an indexing scheme from vendor A. Of course, there are many ways to accomplish these additions so a consensus approach will be a challenge!

As it stands, LAS 1.4 is a fairly robust format for conveying point cloud data. Based on what I see on the hardware side, this version should give us a few years of breathing room before we need to do another update to the specification.

Summary

The LAS specification has made the transport and exploitation of lidar data considerably easier and more efficient by allowing all members of the community to communicate data in an unambiguous manner. While never intended as an *exploitation* format, a number of software providers have implemented LAS directly into the exploitation side of their software (as opposed to simply bridging at the input/output sections). This trend puts added responsibility on the LWG, particularly in the areas that impact Input/Output speed.

Clearly LAS has passed the "tipping point" and will, at least for the immediate future, remain the format of choice for time-dependent point cloud data.

Author

Lewis Graham, President/CTO
GeoCue Corporation
9668 Madison Blvd, Suite 202, Madison, AL 35758
lgraham@geocue.com