August 24th 2011

An alternate proposal for the LAS 1.4 specification

This is not an official LAS document! Do not implement! Yes, it would be easy. (-;

This alternate proposal for the LAS 1.4 specification is result of numerous discussion via personal emails and messages in the CLICK bulletin board, from the ASPRS LAS committee mailing list, and in "the LAS room". It was prepared by Martin Isenburg - the author of LAStools and LASzip compression - to have a tangible document that describes an alternate LAS 1.4 specification that achieves all goals of LAS 1.4, is easy to implement, and harmonically blends industry needs with the desire of the LIDAR community for maintaining compatibility in the LAS 1.x family.

On many previous occasions [1,2,3,4,5], myself - and many others - have outlined that a LAS 1.4 standard that *unnecessarily* breaks the compatibility of the LAS 1.x family does a disservice to the LIDAR community, endangers the success of the LAS format, and undermines the credibility of the ASPRS as a governing standards body. This sentiment is shared by the community at large.

Contrary to claims by Lewis Graham [6], the backwards / forwards compatibility described in this alternate proposal for the LAS 1.4 specification is **very easy to implement** and will **not cause software to "crash"**. His worries that "The problematic scenario is a LAS 1.3 reader that simply crashes when reading a LAS 1.4 file with new point record types." [6] are unfounded. The same could be said about any older LAS reader opening a LAS 1.3 file with newer point types 4 or 5.

His dismissal of our request for compatibility as being "akin to expecting PowerPoint 2003 to read a PowerPoint 2007 file" [6] does not convince. Are we to look at Microsoft as a role model? Not to mention, both Word and Powerpoint 2003 provide on-the-fly converters that can read 2007. A better mentor may be the Adobe Acrobat 6.0 reader that happily reads 7.0 content (without the new features). We can achieve this by rearranging a few fields in the LAS header.

Finally, I have yet to hear from any other person besides Lewis Graham an argument in favor of discontinuing compatibility with LAS 1.4. Who agrees with Lewis? Hello? Anyone? I vehemently object that "The LWG does not intend to revisit this issue during the comment period" [6].

[1] https://lidarbb.cr.usgs.gov/index.php?showtopic=6385
[2] https://lidarbb.cr.usgs.gov/index.php?showtopic=11725
[3] https://lidarbb.cr.usgs.gov/index.php?showtopic=13592
[4] http://www.cs.unc.edu/~isenburg/lastools/download/open_letter_broken_LAS_1_4_specification.pdf
[5] http://groups.google.com/group/lasroom
[6] http://www.asprs.org/Press-Releases/LAS-1-4-Draft-Specification-Released-by-ASPRS.html

This alternate specification is not only backward but also forward compatible. That means that any older LAS reader that was designed to be forward-compatible (e.g. libLAS, LASlib, readers derived from open source projects, ...) will be able to read LAS 1.4 files that do not contain **exclusive LAS 1.4 content**. We define **exclusive LAS 1.4 content** as files that either contain the new point types 6 - 10 or more than 4,294,967,295 points of the old point types 0 - 5.

The main features of this alternate specification in comparison to the official proposal [6] are

- **Contains all the functionalities** of the proposal submitted by Lewis Graham
    - 64 bit counters for more than 4,294,967,295 points
    - Variable Length Records that allow higher payloads
    - Up to 15 returns per pulse
    - Up to 256 different classifications
    - Higher precision scan angles
    - ...
- **Is fully backward and forward compatible**
- Is a super-specification for content of the entire LAS 1.x family
- Still supports point types 0 to 5
- Keeps the LAS Header compatible with LAS 1.0 - 1.3 (i.e. first 235 bytes do not change)
- Leaves Variable Length Records compatible (i.e. VRL headers don't change by 6 bytes)
- Introduces new Appended Variable Length Records (AVLR) that not only allow higher payloads but also make it possible to add information (e.g. projection, spatial indexing, pyramiding, ...) to the end of existing LAS files **without having to rewrite the entire file**
- Allows defining scan angle precision via a scaling factor for the new point formats 6 - 10
- Has a new, optional LASF_Spec VLR to describe the "extra bytes" of a point record.

LAS FORMAT DEFINITION:

The format contains binary data consisting of a header block, (optional) Variable Length Records (VLRs), the Point Data Records, and also (optional) any number of Appended Variable Length Records (AVLRs).

Table 0 – LAS Format Definition

PUBLIC HEADER BLOCK
VARIABLE LENGTH RECORDS (VLR)
POINT DATA RECORDS
APPENDED VARIABLE LENGTH RECORDS (AVLR)

A LAS file that contains point record types 4 or 5 could potentially contain one block of waveform data packets that immediately follows the Point Data Records that is stored in an Extended Variable Length Record (EVLR). While still supported for compatibility reasons this practice is deprecated with LAS 1.4. Instead the waveform data packets should be stored in an external *.wdp file as described in the LAS 1.3 specification.

Table 1 – LAS Format Definition When Containing Waveform Data (deprecated)

PUBLIC HEADER BLOCK
VARIABLE LENGTH RECORDS (VLR)
POINT DATA RECORDS
one EVLR with WAVEFORM DATA PACKETS
APPENDED VARIABLE LENGTH RECORD (AVLR)

All data are in little-endian format. The header block consists of a public block followed by optional Variable Length Records (VLRs). The public block contains generic data such as point numbers and coordinate bounds. The Variable Length Records contain variable types of data including projection information, metadata, waveform packet information, and user application data. They are limited to a data payload of 65535 bytes. One Extended Variable Length Record (EVRL) may follow the Point Data Records. This is used to store waveform data packets inside a LAS file - a practice that was introduced with LAS 1.3 but is deprecated with LAS 1.4. A new concept for LAS 1.4 are the optional Appended Variable Length Records (AVLRs). They are at the end of a LAS file and are organized into a linked list. They have two advantages: (1) they allow a higher payload than a VLR and (2) they can be appended to the end of a LAS file. This is a huge plus over the old VLRs as it allows, for example, to add projection or spatial indexing information to an existing LAS file without requiring a full re-write of the entire file.

DATA TYPES:

The following data types are used in the LAS format definition. Note that these data types are conformant to the 1999 ANSI C Language Specification (ANSI/ISO/IEC 9899:1999 ("C99").

- char (1 byte)
- unsigned char (1 byte)
- short (2 bytes)
- unsigned short (2 bytes)
- long (4 bytes)
- unsigned long (4 bytes)
- long long (8 bytes)
- unsigned long long (8 bytes)
- double (8 byte IEEE floating point format)

PUBLIC HEADER BLOCK:

Table 2 – Public Header Block

Item	Format	Size	Required
File Signature (“LASF”)	char[4]	4 bytes	*
File Source ID	unsigned short	2 bytes	*
Global Encoding	unsigned short	2 bytes	*
Project ID - GUID data 1	unsigned long	4 bytes	
Project ID - GUID data 2	unsigned short	2 byte	
Project ID - GUID data 3	unsigned short	2 byte	
Project ID - GUID data 4	unsigned char[8]	8 bytes	
Version Major	unsigned char	1 byte	*
Version Minor	unsigned char	1 byte	*
System Identifier	char[32]	32 bytes	*
Generating Software	char[32]	32 bytes	*
File Creation Day of Year	unsigned short	2 bytes	*
File Creation Year	unsigned short	2 bytes	*
Header Size	unsigned short	2 bytes	*
Offset to point data	unsigned long	4 bytes	*
Number of Variable Length Records	unsigned long	4 bytes	*
Point Data Format ID (0-99 for spec)	unsigned char	1 byte	*
Point Data Record Length	unsigned short	2 bytes	*
Number of point records	unsigned long	4 bytes	*

Number of points by return	unsigned long [5]	20 bytes	*
X scale factor	double	8 bytes	*
Y scale factor	double	8 bytes	*
Z scale factor	double	8 bytes	*
X offset	double	8 bytes	*
Y offset	double	8 bytes	*
Z offset	double	8 bytes	*
Max X	double	8 bytes	*
Min X	double	8 bytes	*
Max Y	double	8 bytes	*
Min Y	double	8 bytes	*
Max Z	double	8 bytes	*
Min Z	double	8 bytes	*
Start of Waveform Data Packet Record	unsigned long long	8 bytes	*
Extended Number of point records	unsigned long long	8 bytes	*
Extended Number of points by return	unsigned long long [15]	120 bytes	*
First Appended Variable Length Record	unsigned long long	8 bytes	*
Scan angle scale factor	double	8 bytes	*

Any field in the Public Header Block that is not required and is not used must be zero filled.

File Signature: The file signature must contain the four characters “LASF”, and it is required by the LAS specification. These four characters can be checked by user software as a quick look initial determination of file type.

File Source ID: This field should be set to a value ranging from 1 to 65,535. If this file was derived from an original flight line, this is often the flight line number. A value of zero (0) is interpreted to mean that an ID has not been assigned. In this case, processing software is free to assign a number. Note that this scheme allows a LIDAR project to contain up to 65,535 unique sources. A source can be an original flight line or it can be result of merge and/or extract operations.

Global Encoding: This is a bit field used to indicate certain global properties about the file. In LAS 1.2 (the version in which this field was introduced), only the low bit is defined (this is the bit, that if set, would have the unsigned integer yield a value of 1). This bit field is defined as:

Table 3 - Global Encoding - Bit Field Encoding

Bits	*Field Name*	*Description*
0	GPS Time Type	The meaning of GPS Time in the point records. If this bit is not set, the GPS time in the point record fields is GPS Week Time (the same as versions 1.0 through 1.2 of LAS). Otherwise, if this bit is set, the GPS Time is standard GPS Time (satellite GPS Time) minus 1 x 10^9 (Adjusted Standard GPS Time). The offset moves the time back to near zero to improve floating point resolution.
1	Waveform Data Packets Internal	If this bit is set, the waveform data packets are located within this file (note that this bit is mutually exclusive with bit 2). This is deprecated now.
2	Waveform Data Packets External	If this bit is set, the waveform data packets are located externally in an auxiliary file with the same base name as this file but the extension *.wdp.

		(note that this bit is mutually exclusive with bit 1)
3	Return numbers have been synthetically generated	If this bit is set, the point return numbers in the point pata records have been synthetically generated. This could be the case, for example, when a composite file is created by combining a First Return File and a Last Return File. In this case, first return data will be labeled “1 of 2” and second return data will be labeled “2 of 2”
4:15	Reserved	Must be set to zero

Project ID (GUID data): The four fields that comprise a complete Globally Unique Identifier (GUID) are now reserved for use as a Project Identifier (Project ID). The field remains optional. The time of assignment of the Project ID is at the discretion of processing software. The Project ID should be the same for all files that are associated with a unique project. By assigning a Project ID and using a File Source ID (defined above) every file within a project and every point within a file can be uniquely identified, globally.

Version Number: The version number consists of a major and minor field. The major and minor fields combine to form the number that indicates the format number of the current specification itself. For example, specification number 1.4 would contain 1 in the major field and 4 in the minor field.

System Identifier: The version 1.0 specification assumed that LAS files are exclusively generated as a result of collection by a hardware sensor. Subsequent versions recognize that files often result from extraction, merging or modifying existing data files. Thus System ID becomes:

Table 4 – System Identifier

Generating Agent	**System ID**
Hardware system	String identifying hardware (e.g. “ALTM 1210” or “ALS50”
Merge of one or more files	“MERGE”
Modification of a single file	“MODIFICATION”
Extraction from one or more files	“EXTRACTION”
Reprojection, rescaling, warping, etc.	“TRANSFORMATION”
Some other operation	“OTHER” or a string up to 32 characters identifying the operation

Generating Software: This information is ASCII data describing the generating software itself. This field provides a mechanism for specifying which generating software package and version was used during LAS file creation (e.g. “TerraScan V-10.8”, “REALM V-4.2” and etc.). If the character data is less than 32 characters, the remaining data must be null.

File Creation Day of Year: Day, expressed as an unsigned short, on which this file was created. Day is computed as the Greenwich Mean Time (GMT) day. January 1 is considered day 1.

File Creation Year: The year, expressed as a four digit number, in which the file was created.

Header Size: The size, in bytes, of the Public Header Block itself. In the event that the header is extended by a new revision of the LAS specification through the addition of data at the end of the header, the Header Size field will be updated with the new header size. The Public Header Block may not be extended by users.

Offset to point data: The actual number of bytes from the beginning of the file to the first field of the first point record data field. This data offset must be updated if any software adds/removes data to/from the Variable Length Records.

Number of Variable Length Records: This field contains the current number of VLRs that are stored in the file preceding the Point Data Records. This number must be updated if the number of VLRs changes.

Point Data Format ID: The point data format ID corresponds to the Point Data Record format type. LAS 1.4 defines types 0 through 10.

Point Data Record Length: The size, in bytes, of the Point Data Record. All Point Data Records within a single LAS file must be the same type and hence the same length. If the specified size is larger than implied by the point format type (e.g. 32 bytes instead of 28 bytes for type 1) the remaining bytes are user-specific "extra bytes". This has been supported since LAS 1.0. What is new in LAS 1.4 is that the format and meaning of such "extra bytes" can (optionally) be described with an Extra Bytes VLR (see Table 23 and Table 24) in order to make it useful to others as well.

Number of point records: This field contains the total number of point records within the file. If this field is zero then the number stored in the 64-bit **Extended number of point records** field is used instead (see below). This will always have to be the case for any LAS file containing more than 4,294,967,295 points.

Number of points by return: These fields contains an array of the total point records per return. The first value will be the total number of records from the first return, the second contains the total number for return two, and so on up to five returns. If a field is zero then the number stored in the corresponding 64-bit **Extended number of points by return** field is used instead (see below). This will always be the case to accommodate numbers larger than 4,294,967,295.

X, Y, and Z scale factors: The scale factor fields contain a double floating point value that is used to scale the corresponding X, Y, and Z long values within the point records. The corresponding X, Y, and Z scale factor must be multiplied by the X, Y, or Z point record value to get the actual X, Y, or Z coordinate. For example, if the X, Y, and Z coordinates are intended to have two decimal point values, then each scale factor will contain the number 0.01.

X, Y, and Z offset: The offset fields should be used to set the overall offset for the point records. In general these numbers will be zero, but for certain cases the resolution of the point data may not be large enough for a given projection system. However, it should always be assumed that these numbers are used. So to scale a given X from the point record, take the point record X multiplied by the X scale factor, and then add the X offset.
$X_{coordinate} = (X_{record} * X_{scale}) + X_{offset}$
$Y_{coordinate} = (Y_{record} * Y_{scale}) + Y_{offset}$
$Z_{coordinate} = (Z_{record} * Z_{scale}) + Z_{offset}$

Max and Min X, Y, Z: The max and min data fields are the actual unscaled extents of the LAS point file data, specified in the coordinate system of the LAS data.

Start of Waveform Data Packet Record: This value provides the offset, in bytes, from the beginning of the LAS file to the first byte of the EVLR that contains the Waveform Data Packets as its payload and that is located after the Point Data Records. If the Waveform Data Packets are stored in an external *.wpd file - which is the preferred practice - then this value is zero.

Extended Number of point records: If the “Number of point records” field is zero, this field contains the actual number of points records in the file. Ideally this field is only used for LAS files that contain exclusive LAS 1.4 content in order to maintain forward compatibility. If the “Number of point records” field is not zero it overrides whatever number is stored here.

Extended Number of points by return: These fields contains an array of the total point records per return up to 15 returns. If the corresponding “Number of point records” fields are non-zero they override whatever number is stored here.

First Appended Variable Length Record: This value provides the offset, in bytes, from the beginning of the LAS file to the first byte of the first Appended Variable Length Record of a single-linked list of AVLRs. If there are no AVLRs then this value is zero.

Scan angle scale factor: This field contains a double floating point value that is used to scale the Scale angle rank short values of each point record. For example, if the scan angles have 0.05 degree increments, then the scale factor can be set to 0.05, if the scan angles have 1 degree increments, it can be set to 1, and if the scan angles have 0.001 degree increments, it can be set to 0.001.

The official proposal to fix the scan angle scale factor at 0.006 is bound to cause headaches.

VARIABLE LENGTH RECORDS (VLRs):

The Public Header Block is followed by one or more Variable Length Records. (There is one mandatory Variable Length Record, **GeoKeyDirectoryTag.**) The number of Variable Length Records is specified in the “Number of Variable Length Records” field in the Public Header Block. The Variable Length Records must be accessed sequentially since the size of each variable length record is contained in the Variable Length Record Header. Each Variable Length Record Header is 54 bytes in length.

Table 5 – Variable Length Record Header

Item	**Format**	**Size**	**Required**
Reserved	unsigned short	2 bytes	
User ID	char[16]	16 bytes	*
Record ID	unsigned short	2 bytes	*
Record Length After Header	unsigned short	2 bytes	*
Description	char[32]	32 bytes	

User ID: The User ID field is ASCII character data that identifies the user which created the variable length record. It is possible to have many Variable Length Records from different sources with different User IDs. If the character data is less than 16 characters, the remaining data must be null. The User ID must be registered with the LAS specification managing body. The management of these User IDs ensures that no two individuals accidentally use the same User ID. Two User IDs are reserved: one for globally specified records (LASF_Spec) and another for projection types (LASF_Projection). Keys may be requested at http://www.asprs.org/lasform/keyform.html.

Record ID: The Record ID is dependent upon the User ID. There can be 0 to 65535 Record IDs for every User ID. The LAS specification manages its own Record IDs (User IDs owned by the specification), otherwise Record IDs will be managed by the owner of the given User ID. Thus each User ID is allowed to assign 0 to 65535 Record IDs in any manner they desire. Publicizing the meaning of a given Record ID is left to the owner of the given User ID. Unknown User ID/Record ID combinations should be ignored.

Record Length after Header: The record length is the number of bytes for the record after the end of the standard part of the header. Thus the entire record length is 54 bytes (the header size of the VLR) plus the number of bytes in the variable length portion of the record.

Description: Optional, null terminated text description of the data. Any remaining characters not used must be null.

EXTENDED VARIABLE LENGTH RECORD (EVLR)

For backwards compatibility there can also be a single Extended Variable Length Record (EVLR) that follows the Point Data Records and contains the Waveform Data Packets. This is the case whenever bit number 1 of the Global Encoding field in the Public Header Block is set and the Start of Waveform Data Packet Record in the Public Header Block is non-zero. Otherwise this EVLR is stored in an external *.wpd file. The latter is recommended practise for LAS 1.4.

Table 6 – Extended Variable Length Record Header

Item	Format	Size	Required
Reserved	unsigned short	2 bytes	
User ID	char[16]	16 bytes	*
Record ID	unsigned short	2 bytes	*
Record Length After Header	unsigned long long	8 bytes	*
Description	char[32]	32 bytes	

The EVLR is in spirit identical to a VLR but can carry a larger payload as the "Record Length After Header" field is 8 bytes instead of 2 bytes. The Extended Variable Length Record Header is 60 bytes in length. It always uses the User ID "LASF_Spec" and the Record ID 65,535.

Waveform Data Packets: (required when using Point formats 4, 5, 9 or 10)

User ID:	LASF_Spec
Record ID:	65,535

The packet of Raw Waveform Amplitude values for all records immediately follow this variable length header. Note that when using a bit resolution that is not an even increment of 8, the last byte of each waveform packet must be padded such that the next waveform record will start on an even byte boundary.

APPENDED VARIABLE LENGTH RECORDS (AVLRs)

At the end of the file there can be any number of Appended Variable Length Records. They are in spirit similar to VLRs but their number is not stored in the Public Header Block; instead they are a organized into a single-linked list. The position of the first AVLRs is specified in the "Start of Appended Variable Length Records" field in the Public Header Block. Since they are orgnized in a linked-list the AVLRs must be accessed sequentially. The Appended Variable Length Record Header is 68 bytes in length. The appending functionality is particularly useful for adding projection, pyramiding, or spatial indexing information to a LAS file without re-writing the entire file.

Table 7 – Appended Variable Length Record Header

Item	Format	Size	Required
Reserved	unsigned short	2 bytes	
User ID	char[16]	16 bytes	*

Record ID	unsigned short	2 bytes	*
Next Appended Variable Length Record	unsigned long long	8 bytes	*
Record Length After Header	unsigned long long	8 bytes	*
Description	char[32]	32 bytes	

Next Appended Variable Length Record: This value provides the offset, in bytes, from the beginning of the LAS file to the first byte of the next Appended Variable Length Record of a single-linked list of AVLRs. If there is no next AVLRs then this value is zero.

POINT DATA RECORDS

Note that the Point Data Start Signature was removed in LAS Version 1.1. LAS file I/O software must use the "Offset to point data" field in the Public Header Block to locate the starting position of the first Point Data Record. Note that all Point Data Records must be the same type. Point data items that are not 'Required' must be set to the equivalent of zero for the data type (e.g. 0.0 for floating types, null for ACSII, 0 for integers).

POINT DATA RECORD FORMAT 0:

Point Data Record Format 0 contains the core 20 bytes that are shared by Point Data Record Formats 0 to 5.

Table 8 – Point Data Record Format 0

Item	Format	Size	Required
X	long	4 bytes	*
Y	long	4 bytes	*
Z	long	4 bytes	*
Intensity	unsigned short	2 bytes	
Return Number	3 bits (bits 0 – 2)	3 bits	*
Number of Returns (given pulse)	3 bits (bits 3 – 5)	3 bits	*
Scan Direction Flag	1 bit (bit 6)	1 bit	*
Edge of Flight Line	1 bit (bit 7)	1 bit	*
Classification	unsigned char	1 byte	*
Scan Angle Rank (-90 to +90) – Left side	char	1 byte	*
User Data	unsigned char	1 byte	
Point Source ID	unsigned short	2 bytes	*

X, Y, and Z: The X, Y, and Z values are stored as long integers. The X, Y, and Z values are used in conjunction with the scale values and the offset values to determine the coordinate for each point as described in the Public Header Block section.

Intensity: 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. Intensity, 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). If intensity is not included, this value must be set to zero. This normalization is required to ensure that data from different sensors can be correctly merged.

Please note that the following four fields (Return Number, Number of Returns, Scan Direction Flag and Edge of Flight Line) are bit fields within a single byte.

Return Number: The Return Number is the pulse return number for a given output pulse. A given output laser pulse can have many returns, and they must be marked in sequence of return. The first return will have a Return Number of one, the second a Return Number of two, and so on up to five returns.

Number of Returns (given pulse): The Number of Returns is the total number of returns for a given pulse. For example, a laser data point may be return two (Return Number) within a total number of five returns.

Scan Direction Flag: The Scan Direction Flag denotes the direction at which the scanner mirror was traveling at the time of the output pulse. A bit value of 1 is a positive scan direction, and a bit value of 0 is a negative scan direction (where positive scan direction is a scan moving from the left side of the in-track direction to the right side and negative the opposite).

Edge of Flight Line: The Edge of Flight Line data bit has a value of 1 only when the point is at the
end of a scan. It is the last point on a given scan line before it changes direction.

Classification: This field represents the “class” attributes of a point. If a point has never been classified, this byte must be set to zero. The format for classification is a bit encoded field with the lower five bits used for the class and the three high bits used for flags. The bit definitions are

Table 9 - Classification Bit Field Encoding for Point Record types 0 to 5

Bit	*Field Name*	*Description*
0:4	Classification	Standard ASPRS classification from 0 - 31 as defined in the classification table further below (see Table 17).
5	Synthetic	If set then this point was created by a technique other than LIDAR collection such as digitized from a photogrammetric stereo model or by traversing a waveform.
6	Key-point	If set, this point is considered to be a model key-point and thus generally should not be withheld in a thinning algorithm.
7	Withheld	If set, this point should not be included in processing (synonymous with Deleted).

Note that bits 5, 6 and 7 are treated as flags and can be set or clear in any combination. For example, a point with bits 5 and 6 both set to one and the lower five bits set to 2 (see table below) would be a ground point that had been Synthetically collected and marked as a model key-point.

[A note on Bit Fields – The LAS storage format is “Little Endian.” This means that multi-byte data fields are stored in memory from least significant byte at the low address to most significant byte at the high address. Bit fields are always interpreted as bit 0 set to 1 equals 1, bit 1 set to 1 equals 2, bit 2 set to 1 equals 4 and so forth.]

Scan Angle Rank: The Scan Angle Rank is a signed one-byte number with a valid range from -90 to +90. The Scan Angle Rank is the angle (rounded to the nearest integer in the absolute value sense) at which the laser point was output from the laser system including the roll of the aircraft. The scan angle is within 1 degree of accuracy from +90 to –90 degrees. The scan angle is an angle based on 0 degrees being nadir, and –90 degrees to the left side of the aircraft in the direction of flight.

User Data: This field may be used at the user’s discretion.

Point Source ID: This value indicates the file from which this point originated. Valid values for this field are 1 to 65,535 inclusive with zero being used for a special case discussed below. The numerical value corresponds to the File Source ID from which this point originated. Zero is reserved as a convenience to system implementers. A Point Source ID of zero implies that this point originated in this file. This implies that processing software should set the Point Source ID equal to the File Source ID of the file containing this point at some time during processing.

POINT DATA RECORD FORMAT 1:

Point Data Record Format 1 is the same as Point Data Record Format 0 with the addition of GPS Time.

Table 10 – Point Data Record Format 1

Item	**Format**	**Size**	**Required**
X	long	4 bytes	*
Y	long	4 bytes	*
Z	long	4 bytes	*
Intensity	unsigned short	2 bytes	
Return Number	3 bits (bits 0 – 2)	3 bits	*
Number of Returns (given pulse)	3 bits (bits 3 – 4)	3 bits	*
Scan Direction Flag	1 bit (bit 6)	1 bit	*
Edge of Flight Line	1 bit (bit 7)	1 bit	*
Classification	unsigned char	1 byte	*
Scan Angle Rank (-90 to +90) – Left side	char	1 byte	*
User Data	unsigned char	1 byte	
Point Source ID	unsigned short	2 bytes	*
GPS Time	double	8 bytes	*

GPS Time: The GPS Time is the double floating point time tag value at which the point was acquired. It is GPS Week Time if the Global Encoding low bit is clear and Adjusted Standard GPS Time if the Global Encoding low bit is set (see Global Encoding in the Public Header Block description).

POINT DATA RECORD FORMAT 2:

Point Data Record Format 2 is the same as Point Data Record Format 0 with the addition of three color channels. These fields are used when “colorizing” a LIDAR point using ancillary data, typically from a camera.

Table 11 – Point Data Record Format 2

Item	**Format**	**Size**	**Required**
X	long	4 bytes	*
Y	long	4 bytes	*
Z	long	4 bytes	*
Intensity	unsigned short	2 bytes	
Return Number	3 bits (bits 0, 1, 2)	3 bits	*
Number of Returns (given pulse)	3 bits (bits 3, 4, 5)	3 bits	*
Scan Direction Flag	1 bit (bit 6)	1 bit	*
Edge of Flight Line	1 bit (bit 7)	1 bit	*
Classification	unsigned char	1 byte	*

Scan Angle Rank (-90 to +90) – Left side	char	1 byte	*
User Data	unsigned char	1 byte	
Point Source ID	unsigned short	2 bytes	*
Red	unsigned short	2 bytes	*
Green	unsigned short	2 bytes	*
Blue	unsigned short	2 bytes	*

Red: The Red image channel value associated with this point.

Green: The Green image channel value associated with this point.

Blue: The Blue image channel value associated with this point.

The Red, Green, Blue values should always be normalized to 16 bit values. For example, when encoding an 8 bit per channel pixel, multiply each channel value by 256 prior to storage in these fields. This normalization allows color values from different camera bit depths to be accurately merged.

POINT DATA RECORD FORMAT 3:

Point Data Record Format 3 is the same as Point Data Record Format 2 with the addition of GPS Time.

Table 12 – Point Data Record Format 3

Item	Format	Size	Required
X	long	4 bytes	*
Y	long	4 bytes	*
Z	long	4 bytes	*
Intensity	unsigned short	2 bytes	
Return Number	3 bits (bits 0, 1, 2)	3 bits	*
Number of Returns (given pulse)	3 bits (bits 3, 4, 5)	3 bits	*
Scan Direction Flag	1 bit (bit 6)	1 bit	*
Edge of Flight Line	1 bit (bit 7)	1 bit	*
Classification	unsigned char	1 byte	*
Scan Angle Rank (-90 to +90) – Left side	char	1 byte	*
User Data	unsigned char	1 byte	
Point Source ID	unsigned short	2 bytes	*
GPS Time	double	8 bytes	*
Red	unsigned short	2 bytes	*
Green	unsigned short	2 bytes	*
Blue	unsigned short	2 bytes	*

POINT DATA RECORD FORMAT 4:

Point Data Record Format 4 adds Wave Packets to Point Data Record Format 1.

Table 13 – Point Data Record Format 4

Item	Format	Size	Required
X	long	4 bytes	*
Y	long	4 bytes	*

Z	long	4 bytes	*
Intensity	unsigned short	2 bytes	
Return Number	3 bits (bits 0 – 2)	3 bits	*
Number of Returns (given pulse)	3 bits (bits 3 – 5)	3 bits	*
Scan Direction Flag	1 bit (bit 6)	1 bit	*
Edge of Flight Line	1 bit (bit 7)	1 bit	*
Classification	unsigned char	1 byte	*
Scan Angle Rank (-90 to +90) – Left side	unsigned char	1 byte	*
User Data	unsigned char	1 byte	
Point Source ID	unsigned short	2 bytes	*
GPS Time	double	8 bytes	*
Wave Packet Descriptor Index	unsigned char	1 byte	*
Byte offset to waveform data	unsigned long long	8 bytes	*
Waveform packet size in bytes	unsigned long	4 bytes	*
Return Point Waveform Location	float	4 bytes	*
X(t)	float	4 bytes	*
Y(t)	float	4 bytes	*
Z(t)	float	4 bytes	*

Wave Packet Descriptor Index: This value plus **99** is the Record ID of the Waveform Packet Descriptor indicates the User Defined Record that is used to describe the waveform packet associated with this LIDAR point. Up to 255 different User Defined Records which describe the waveform packet are supported. A value of zero indicates that there is no waveform data associated with this LIDAR point record.

Byte offset to Waveform Packet Data: The waveform packet data are stored in the LAS file in an Extended Variable Length Record (or, optionally, in an auxiliary file). The Byte Offset represents the location of the start of this LIDAR points' waveform packet within the waveform data variable length record (or external file) relative to the beginning of the Waveform Packet Data header. The absolute location of the beginning of this waveform packet relative to the beginning of the file is given by:

Start of Waveform Data Packet Record + Byte offset to Waveform Packet Data

for waveform packets stored within the LAS file and

Byte offset to Waveform Packet Data

for data stored in an auxiliary file

Waveform packet size in bytes: The size, in bytes, of the waveform packet associated with this return. Note that each waveform can be of a different size (even those with the same Waveform Packet Descriptor index) due to packet compression. Also note that waveform packets can be located only via the Byte offset to Waveform Packet Data value since there is no requirement that records be stored sequentially.

Return Point location: The offset in picoseconds (10^{-12}) from the first digitized value to the location within the waveform packet that the associated return pulse was detected.

X(t), Y(t), Z(t): These parameters define a parametric line equation for extrapolating points along the associated waveform. The position along the wave is given by:

$$X = X_0 + X(t)$$
$$Y = Y_0 + Y(t)$$
$$Z = Z_0 + Z(t)$$

where X, Y and Z are the spatial position of the derived point, X_0, Y_0, Z_0 are the position of the "anchor" point (the X, Y, Z locations from this point's data record) and t is the time, in picoseconds, relative to the anchor point (i.e. t = zero at the anchor point). The units of X, Y and Z are the units of the coordinate systems of the LAS data. If the coordinate system is geographic, the horizontal units are decimal degrees and the vertical units are meters.

POINT DATA RECORD FORMAT 5:

Point Data Record Format 5 adds Wave Packets to Point Data Record Format 3.

Table 14 – Point Data Record Format 5

Item	Format	Size	Required
X	long	4 bytes	*
Y	long	4 bytes	*
Z	long	4 bytes	*
Intensity	unsigned short	2 bytes	
Return Number	3 bits (bit 0 – 2)	3 bits	*
Number of Returns (given pulse)	3 bits (bit 3 – 5)	3 bits	*
Scan Direction Flag	1 bit (bit 6)	1 bit	*
Edge of Flight Line	1 bit (bit 7)	1 bit	*
Classification	unsigned char	1 byte	*
Scan Angle Rank (-90 to +90) – Left side	unsigned char	1 byte	*
User Data	unsigned char	1 byte	
Point Source ID	unsigned short	2 bytes	*
GPS Time	double	8 bytes	*
Red	unsigned short	2 bytes	*
Green	unsigned short	2 bytes	*
Blue	unsigned short	2 bytes	*
Wave Packet Descriptor Index	unsigned char	1 byte	*
Byte offset to waveform data	unsigned long long	8 bytes	*
Waveform packet size in bytes	unsigned long	4 bytes	*
Return Point Waveform Location	float	4 bytes	*
X(t)	float	4 bytes	*
Y(t)	float	4 bytes	*
Z(t)	float	4 bytes	*

POINT DATA RECORD FORMAT 6:

Point Data Record Format 6 contains the core 30 bytes that are shared by Point Data Record Formats 6 to 10. The difference to the core 20 bytes of Point Data Record Formats 0 to 5 is that there are more bits for return numbers in order to support up to 15 return, there are more bits for point classifications to support up to 256 classes, there is a higher precision scan angle (16 bits instead of 8), and that the GPS time is mandatory.

Table 15 – Point Data Record Format 6

Item	Format	Size	Required
X	long	4 bytes	*
Y	long	4 bytes	*
Z	long	4 bytes	*
Intensity	unsigned short	2 bytes	
Return Number	4 bits (bits 0 - 3)	4 bits	*

Number of Returns (given pulse)	4 bits (bits 4 - 7)	4 bits	*
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	*
Classification	unsigned char	1 byte	*
Scan Angle	short	2 bytes	*
User Data	unsigned char	1 byte	
Point Source ID	unsigned short	2 bytes	*
GPS Time	double	8 bytes	*

Note that the following five fields (Return Number, Number of Returns, Classification Flags, Scan Direction Flag and Edge of Flight Line) are bit fields,encoded into two bytes.

Return Number: The Return Number is the pulse return number for a given output pulse. A given output laser pulse can have many returns, and they must be marked in sequence of return. The first return will have a Return Number of one, the second a Return Number of two, and so on up to fifteen returns. The Return Number must be between 1 and the Number of Returns, inclusive.

Number of Returns (given pulse): The Number of Returns is the total number of returns for a given pulse. For example, a laser data point may be return two (Return Number) within a total number of up to fifteen returns.

Classification Flags: Classification flags are used to indicate special characteristics associated with the point. The bit definitions are:

Table 16 - Classification Bit Field Encoding for Point Record types 6 to 10

Bit	***Field Name***	***Description***
0	Synthetic	If set then this point was created by a technique other than LIDAR collection such as digitized from a photogrammetric stereo model or by traversing a waveform.
1	Key-point	If set, this point is considered to be a model key-point and thus generally should not be withheld in a thinning algorithm.
2	Withheld	If set, this point should not be included in processing (synonymous with Deleted).
3	Overlap	If set, this point is within the overlap region of two or more swaths or takes. Setting this bit is not mandatory but allows Classification of overlap points to be preserved.

Note that these bits are treated as flags and can be set or cleared in any combination. For example, a point with bits 0 and 1 both set to one and the Classification field set to 2 (see table below) would be a *ground* point that had been *synthetically* collected and marked as a *model key-point*.

Scanner Channel: Scanner Channel is used to indicate the channel (scanner head) of a multi-channel system. Channel 0 is used for single scanner systems. Up to four channels are supported (0-3).

Scan Direction Flag: The Scan Direction Flag denotes the direction at which the scanner mirror was traveling at the time of the output pulse. A bit value of 1 is a positive scan direction, and a bit value of 0 is a negative scan direction (where positive scan direction is a scan moving from the left side of the in-track direction to the right side and negative the opposite).

Edge of Flight Line: The Edge of Flight Line data bit has a value of 1 only when the point is at the end of a scan. It is the last point on a given scan line before it changes direction. Note that this field has no meaning for 360° Field of View scanners (such as Mobile LIDAR scanners) and should not be set.

Classification must adhere to the following standard:

Table 17 - ASPRS Standard LIDAR Point Classes

Classification Value	***Meaning***
0	Created, never classified
1	Unclassified[1]
2	Ground
3	Low Vegetation
4	Medium Vegetation
5	High Vegetation
6	Building
7	Low Point (noise)
8	Model Key-point (mass point) – deprecated. The Model Key-point bit should be instead used. In the future, this value will be used for High Noise.
9	Water
10	Rail
11	Road Surface
12	Reserved (or “Overlap Points” in older content)
13	Wire - Guard
14	Wire – Conductor (Phase)
15	Transmission Tower
16	Wire-structure Connector (e.g. Insulator)
17	Reserved
18-63	Reserved
64-255	User definable

Scan Angle: The Scan Angle is a signed short that - multiplied with the “Scan angle scale factor” from the Public Header Block is the angle (rounded to the nearest precision increment in the absolute value sense) 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. Counter-Clockwise rotation, as viewed from the rear of the sensor, facing in the along-

[1] 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.

track (positive trajectory) direction (e.g. negative degrees are to the left side of the aircraft in the direction of flight.

The official proposal to fix the scan angle scale factor at 0.006 is bound to cause headaches as most scan angle increments used by scanner hardware (e.g. 1.0, 0.25, 0.05, 0.01, 0.005, 0.001) cannot be divided by 0.006. Enforcing a one-fits-all scan angle scale factor would – unnecessarily - lead to unnecessary quantization error. LAS already uses a user-defined scale factor for the x, y, and z coordinates. I see no reason not to extend this practice to the scan angle.

POINT DATA RECORD FORMAT 7:

Point Data Record Format 7 is the same as Point Data Record Format 6 with the addition of three RGB color channels. These fields are used when "colorizing" a LIDAR point using ancillary data, typically from a camera.

Table 18 – Point Data Record Format 7

Item	**Format**	**Size**	**Required**
X	long	4 bytes	*
Y	long	4 bytes	*
Z	long	4 bytes	*
Intensity	unsigned short	2 bytes	
Return Number	4 bits (bits 0, 1, 2, 3)	4 bits	*
Number of Returns (given pulse)	4 bits (bits 4, 5, 6, 7)	4 bits	*
Classification Flags	4 bits (bits 0 – 3)	4 bits	
Scanner Channel	2 bits (4-5)	2 bits	*
Scan Direction Flag	1 bit (bit 6)	1 bit	*
Edge of Flight Line	1 bit (bit 7)	1 bit	*
Classification	unsigned char	1 byte	*
Scan Angle	short	2 bytes	*
User Data	unsigned char	1 byte	
Point Source ID	unsigned short	2 bytes	*
GPS Time	double	8 bytes	*
Red	unsigned short	2 bytes	*
Green	unsigned short	2 bytes	*
Blue	unsigned short	2 bytes	*

POINT DATA RECORD FORMAT 8:

Point Data Record Format 8 is the same as Point Data Record Format 7 with the addition of a NIR (near infrared) channel.

Table 19 – Point Data Record Format 8

Item	**Format**	**Size**	**Required**
X	long	4 bytes	*
Y	long	4 bytes	*
Z	long	4 bytes	*
Intensity	unsigned short	2 bytes	
Return Number	4 bits (bits 0, 1, 2, 3)	4 bits	*

Number of Returns (given pulse)	4 bits (bits 4, 5, 6, 7)	4 bits	*
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	*
Classification	unsigned char	1 byte	*
Scan Angle	short	2 bytes	*
User Data	unsigned char	1 byte	
Point Source ID	unsigned short	2 bytes	*
GPS Time	double	8 bytes	*
Red	unsigned short	2 bytes	*
Green	unsigned short	2 bytes	*
Blue	unsigned short	2 bytes	*
NIR	unsigned short	2 bytes	*

NIR: The NIR (near infrared) channel value associated with this point.

Note that Red, Green, Blue and NIR values should always be normalized to 16 bit values. For example, when encoding an 8 bit per channel pixel, multiply each channel value by 256 prior to storage in these fields. This normalization allows color values from different camera bit depths to be accurately merged.

POINT DATA RECORD FORMAT 9:

Point Data Record Format 9 adds Wave Packets to Point Data Record Format 6.

Table 20 – Point Data Record Format 9

Item	**Format**	**Size**	**Required**
X	long	4 bytes	*
Y	long	4 bytes	*
Z	long	4 bytes	*
Intensity	unsigned short	2 bytes	
Return Number	4 bits (bits 0 – 3)	4 bits	*
Number of Returns (given pulse)	4 bits (bits 4 – 7)	4 bits	*
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	*
Classification	unsigned char	1 byte	*
Scan Angle	short	2 bytes	*
User Data	unsigned char	1 byte	
Point Source ID	unsigned short	2 bytes	*
GPS Time	double	8 bytes	*
Wave Packet Descriptor Index	unsigned char	1 byte	*
Byte offset to waveform data	unsigned long long	8 bytes	*
Waveform packet size in bytes	unsigned long	4 bytes	*
Return Point Waveform Location	float	4 bytes	*
X(t)	float	4 bytes	*
Y(t)	float	4 bytes	*

Z(t)	float	4 bytes	*

POINT DATA RECORD FORMAT 10:

Point Data Record Format 10 adds Wave Packets to Point Data Record Format 7.

Table 21 – Point Data Record Format 10

Item	**Format**	**Size**	**Required**
X	long	4 bytes	*
Y	long	4 bytes	*
Z	long	4 bytes	*
Intensity	unsigned short	2 bytes	
Return Number	4 bits (bit 0 – 3)	4 bits	*
Number of Returns (given pulse)	4 bits (bit 4 – 7)	4 bits	*
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	*
Classification	unsigned char	1 byte	*
Scan Angle	short	2 bytes	*
User Data	unsigned char	1 byte	
Point Source ID	unsigned short	2 bytes	*
GPS Time	double	8 bytes	*
Red	unsigned short	2 bytes	*
Green	unsigned short	2 bytes	*
Blue	unsigned short	2 bytes	*
NIR	unsigned short	2 bytes	*
Wave Packet Descriptor Index	unsigned char	1 byte	*
Byte offset to waveform data	unsigned long long	8 bytes	*
Waveform packet size in bytes	unsigned long	4 bytes	*
Return Point Waveform Location	float	4 bytes	*
X(t)	float	4 bytes	*
Y(t)	float	4 bytes	*
Z(t)	float	4 bytes	*

Point Data Record Format 10 is the same as Point Data Record Format 9 with the addition of RGB and NIR values.

Defined Variable Length Records:
The projection information for the point data is required for all data. The projection information will be placed in Variable Length Records. Placing the projection information within the Variable Length Records allows for any projection to be defined including custom projections. The Well Known Text (WKT) specification is used for defining horizontal and vertical reference/projection system information for LAS 1.4 and beyond (www.opengeospatial.org/standards/ct). The GeoTIFF tags will continue to be used as an optional, secondary source of reference/coordinate system information (http://www.remotesensing.org/geotiff/geotiff.html).

Georeferencing Information

As of LAS 1.4, the preferred method for specifying georeferencing information for point data in LAS is the “Well-Known Text Representation of Spatial Reference Systems” (WKT). For definition of WKT we refer to Open Geospatial Consortium (OGC) specification “OpenGIS Coordinate Transformation Service Implementation Specification” revision 1.00 released 12 January 2001, section 7 (Coordinate Transformation Services Spec).

As there are a few dialects of WKT, please note LAS is *not* using the “ESRI WKT” dialect, which does not include TOWGS84 and authority nodes.

WKT georeferencing information can be specified in two optional Variable Length Records, the OGC Math Transform WKT Redord and the OGC Coordinate System WKT Record, as follows. Note that the Math Transform WKT record is added for completeness, and a Coordinate System WKT *may or may not* require a Math Transform WKT Record (a parameterized math transform definition).

OGC Math Transform WKT Record:

User ID:	LASF_Projection
Record ID:	2111

This record contains the textual data representing a Math Transform WKT as defined in section 7 of the Coordinate Transformation Services Spec, with the following notes:

- The OGC Math Transform WKT VLR data shall be a null-terminated string.
- The OGC Math Transform WKT VLR data shall be considered UTF-8.
- The OGC Math Transform WKT VLR data shall be considered C locale-based, and no localization of the numeric strings within the WKT should be performed.

OGC Coordinate System WKT:

User ID:	LASF_Projection
Record ID:	2112

This record contains the textual data representing a Coordinate System WKT as defined in section 7 of the Coordinate Transformation Services Spec, with the following notes:

- The OGC Math Transform WKT VLR data shall be a null-terminated string.
- The OGC Math Transform WKT VLR data shall be considered UTF-8.
- The OGC Math Transform WKT VLR data shall be considered C locale-based, and no localization of the numeric strings within the WKT should be performed.

As a means of transition, the GeoTIFF records as specified in LAS 1.3 may continue to be used in LAS 1.4. Software reading LAS 1.4 files must be able to consume the GeoTIFF VLR records with user ID “LASF_Projection” and Record IDs 34735 through 34737. It is acceptable to write only GeoTIFF VLR records into an LAS file, while software systems that choose to write OGC WKT VLRs must also write GeoTIFF VLRs for backward compatibility. If both WKT and GeoTIFF VLRs are present, then the WKT information takes precedence

The particulars of these GeoTIFF VLRs are repeated here for specification completeness (same as defined in LAS 1.3).

Georeferencing for the LAS format may use the same robust mechanism that was developed for the GeoTIFF standard. The variable length header records section may contain the same data that would be contained in the GeoTIFF key tags of a TIFF file. Since LAS is not a raster format and each point contains its own absolute location information, only 3 of the 6 GeoTIFF tags are necessary when using GeoTIFF records instead of WKT records. The ModelTiePointTag (33922), ModelPixelScaleTag (33550), and ModelTransformationTag (34264) records can be excluded. The GeoKeyDirectoryTag (34735), GeoDoubleParamsTag (34736), and GeoASCIIParamsTag (34737) records are used.

Only the GeoKeyDirectoryTag record is required when using GEOTIFF records instead of WKT records. The GeoDoubleParamsTag and GeoASCIIParamsTag records may or may not be present, depending on the content of the GeoKeyDirectoryTag record.

GeoKeyDirectoryTag Record:

User ID:	LASF_Projection
Record ID:	34735

This record contains the key values that define the coordinate system. A complete description can be found in the GeoTIFF format specification. Here is a summary from a programmatic point of view for someone interested in implementation.

The GeoKeyDirectoryTag is defined as just an array of unsigned short values. But, programmatically, the data can be seen as something like this:

```
struct sGeoKeys
{
   unsigned short wKeyDirectoryVersion;
   unsigned short wKeyRevision;
   unsigned short wMinorRevision;
   unsigned short wNumberOfKeys;
   struct sKeyEntry
   {
      unsigned short wKeyID;
      unsigned short wTIFFTagLocation;
      unsigned short wCount;
      unsigned short wValue_Offset;
   } pKey[1];
};
```

```
Where:
wKeyDirectoryVersion = 1;        // Always
wKeyRevision = 1;                // Always
wMinorRevision = 0;              // Always
wNumberOfKeys          // Number of sets of 4 unsigned shorts to follow
```

Table 22 – GeoKey Four Unsigned Shorts

Name	**Definition**
wKeyID	Defined key ID for each piece of GeoTIFF data. IDs contained in the GeoTIFF specification.
wTIFFTagLocation	Indicates where the data for this key is located: 0 means data is in the wValue_Offset field as an unsigned short.

	34736 means the data is located at index wValue_Offset of the GeoDoubleParamsTag record. 34737 means the data is located at index wValue_Offset of the GeoAsciiParamsTag record.
wCount	Number of characters in string for values of GeoAsciiParamsTag , otherwise is 1
wValue_Offset	Contents vary depending on value for wTIFFTagLocation above

GeoDoubleParamsTag Record: (optional)

User ID: LASF_Projection
Record ID: 34736
This record is simply an array of doubles that contain values referenced by tag sets in the GeoKeyDirectoryTag record.

GeoAsciiParamsTag Record: (Optional)

User ID: LASF_Projection
Record ID: 34737
This record is simply an array of ASCII data. It contains many strings separated by null terminator characters which are referenced by position from data in the GeoKeyDirectoryTag record.

Classification lookup: (optional)

User ID: LASF_Spec
Record ID: 0
Record Length after Header: 256 recs X 16 byte struct len

```
struct CLASSIFICATION
{
        unsigned char ClassNumber;
        char Description[15];
};
```

Header lookup for flight-lines:
(Removed with Version 1.1 - Point Source ID in combination with Source ID provides the new scheme for directly encoding flight line number. Thus variable Record ID 1 now becomes reserved for future use.)

User ID: LASF_Spec
Record ID: 1

Histogram: (optional)

User ID: LASF_Spec
Record ID: 2

Text area description: (optional)

User ID: LASF_Spec
Record ID: 3

Extra Bytes: (optional)

User ID: LASF_Spec
Record ID: 4
Record Length after Header: n records x 92 bytes

This (optional) record is only needed for LAS files that contain user-defined "extra bytes" for every LAS point. This happens whenever the point record size is set to a larger value than required by the point type. For example, when a LAS file containing point type 1 has a point record size of 32 instead of 28 then there are 4 extra bytes per point. The "Extra Bytes" VLR contains a simple description of type and meaning of these extra bytes so they can be useful to other people by - optionally - exposing these "extra bytes" semantics via the LAS reader. The additional 4 bytes, for example, could be a floating point value that specifies the pulse width. In this case there would only be a single EXTRA_BYTES struct in the payload of this VLR.

The bit mask options specifies whether the min and max range of the value have been set (i.e. are meaningful), whether the scale and/or offset values are set with which the extrabytes are then to be multiplied and translated to reach the actual value, and whether there is a special value that should be interpreted as no_data. By default all bits are zero which is supposed to mean that the values in the corresponding fields are to be disregarded (and will probably be set to zero).
If the selected data_type is less than 8 bytes, the no_data_value, min, and max field should be upcast into 8-byte storage. For any float these 8 bytes would be a double, for any unsigned char, unsigned short, or unsigned long they would be an unsigned long long and for any char, short, or long, they would be a long long.

```
struct EXTRA_BYTES
{
        unsigned char       reserved[2];        // 2 bytes
        unsigned char       data_type;          // 1 byte
        unsigned char       options;            // 1 byte
        char                name[16];           // 16 bytes
        anytype             no_data_value;      // 8 bytes
        anytype             min;                // 8 bytes
        anytype             max;                // 8 bytes
        double              scale;              // 8 bytes
        double              offset;             // 8 bytes
        char                description[32];    // 32 bytes
};                                              // total of 92 bytes
```

Table 23 - Values for Extra Bytes Data Types

Value	***Meaning***	***Size***
0	unsigned char	1 byte
1	char	1 byte
2	unsigned short	2 bytes
3	short	2 bytes
4	unsigned long	4 bytes
5	long	4 bytes
6	unsigned long long	8 bytes
7	long long	8 bytes
8	float	4 bytes
9	double	8 bytes

Table 24 - Option Bit Field Encoding

Bit	Field Name	Description
0	no_data	If set the no_data value is relevant.
1	min	If set the min value is relevant
2	max	If set the max value is relevant
3	scale	If set each extra byte value should be multiplied by the scale value
4	offset	If set each extra byte value should be translated by the offset value

Waveform Packet Descriptor: (required when using Point format 4 or 5)

User ID: LASF_Spec
Record ID: n

Where n >=100 and n <355

These records contain information that describes the configuration of the waveform packets. Since systems may be configured differently at different times throughout a job, the LAS file supports 255 Waveform Packet Descriptors.

Table 24 – Waveform Packet Descriptor User Defined Record

Item	Format	Size	Required
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 bits	*
Digitizer Offset	double	8 bits	*

Bits per sample: 2 through 32 bits are supported.

Waveform Compression type: It is expected that in the future standard compression types will be adopted by the LAS committee. This field will indicate the compression algorithm used for the waveform packets associated with this descriptor. A value of 0 indicates no compression. Zero is the only value currently supported.

Number of Samples: The number of samples associated with this waveform packet type. This value always represents the fully decompressed waveform packet.

Temporal Sample Spacing: The temporal sample spacing in picoseconds. Example values might be 500, 1000, 2000 and so on, representing digitizer frequencies of 2 GHz, 1 GHz and 500 MHz respectively.

Digitizer Gain: The gain and offset are used to convert the raw digitized value to an absolute digitizer voltage using the formula: VOLTS = OFFSET + GAIN * Raw_Waveform_Amplitude

Digitizer Offset: The gain and voltage offset are used to convert the raw digitized value to a voltage using the formula: VOLTS = OFFSET + GAIN * Raw_Waveform_Amplitude