LAS SPECIFICATION

VERSION 1.4 – R12

10 June 2012

Approved:
November 2011

Published by:
The American Society for Photogrammetry & Remote Sensing
5410 Grosvenor Lane, Suite 210
Bethesda, Maryland 20814-2160
Voice: 301-493-0290
Fax: 301-493-0208
Web: www.asprs.org

REVISION HISTORY

R11 - Approved Version (Nov 2011)
R12 - Errata (June 2012) - Typographical Corrections

- Corrected Public Header size in descriptive paragraph to 375 bytes
- Corrected two instances of Scan Angle Rank from "Unsigned Char" to "Char"

LAS FORMAT VERSION 1.4:

1 Purpose, scope, and applicability

The LAS file is intended to contain LIDAR (or other) point cloud data records. The data will generally be put into this format from software (e.g. provided by LIDAR hardware vendors) which combines GPS, IMU, and laser pulse range data to produce X, Y, and Z point data. The intention of the data format is to provide an open format that allows different LIDAR hardware and software tools to output data in a common format.

This document reflects the fourth revision of the LAS format specification since its initial version 1.0 release.

THE ADDITIONS OF LAS 1.4 INCLUDE:

- Backward compatibility with LAS 1.1 – LAS 1.3 when payloads consist of only legacy content
- LAS 1.4 Mode which supports
  - Extension of offsets and field sizes to support full 64 bit
  - Support for up to 15 returns per outgoing pulse
  - Extension of the Point Class field to support 256 classes
  - Definition of several new ASPRS standard 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
- Other minor changes

2 Conformance

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

3 Authority

The American Society for Photogrammetry & Remote Sensing (ASPRS) is the owner of the LAS Specification. The standard is maintained by committees within the organization as directed by the ASPRS Board of Directors. Questions related to this standard can be directed to ASPRS at 301-493-0290, by email at asprs@asprs.org, or by mail at 5410 Grosvenor Lane, Suite 210, Bethesda, Maryland 20814-2160.

4 Requirements

LAS FORMAT DEFINITION:

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). All data are in little-endian format. The public header block contains generic data such as point numbers and point data bounds. We refer to the data content of the file as the "payload."

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 data payload of 65,535 bytes.

The Extended Variable Length Records (EVRLs) allow a higher payload than 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.

Table 1: LAS 1.4 Format Definition

PUBLIC HEADER BLOCK
VARIABLE LENGTH RECORDS (VLR)
POINT DATA RECORDS
EXTENDED VARIABLE LENGTH RECORDS (EVLR)

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").

LEGACY COMPATIBILITY (LAS 1.1 – LAS 1.3):

LAS 1.4 moves the file specification from a 32 bit file structure (maximum value of $2^{32} - 1$ == 4,294,967,295 == UINT32_MAX) to a 64 bit file structure ($2^{64} - 1$).

To maintain the ability to place a LAS 1.1 through LAS 1.3 payload (point record types 0-5, GeoTIFF coordinate reference system, referred to as "Legacy" payloads) in a LAS 1.4 file structure, it was necessary to duplicate some of the fields within the LAS 1.4 file structure. These duplicate fields are named "Legacy xxx" where "xxx" denotes the meaning of the field.

A LAS 1.4 file writer who wishes to maintain backward compatibility must maintain both the legacy fields and the equivalent non-legacy fields in synchronization. Of course, this is not possible if the number of points exceeds UINT32_MAX, in which case the legacy fields must be set to zero. If a file writer is not maintaining backward compatibility, the legacy fields must always be set to zero.

If there is a discrepancy between a non-zero legacy field and the equivalent LAS 1.4 field, the LAS 1.4 reader should use the legacy value to maintain the same behavior as a LAS 1.1 through LAS 1.3 reader. Best practice is to also throw an informative error so that the file can be repaired.

COORDINATE REFERENCE SYSTEM (CRS) REPRESENTATION:

GeoTIFF is being replaced by Well Known Text (WKT) as the required Coordinate Reference System (CRS) representation for the new point types (6-10) introduced by LAS 1.4.

GeoTIFF is maintained for legacy reasons for point types 0-5.

A “WKT” bit has been added to the Global Encoding flag in the Public Header Block. If this bit is set, the CRS for the file will be located in the WKT (Extended) Variable Length Records (EVLR, VLR).

A file writer who desires to maintain backward compatibility with legacy LAS for point types 0-5 must add a GeoTIFF VLR to represent the CRS for the file and ensure that the WKT bit is false.

The CRS representation is summarized in Table 2

Table 2: Coordinate Reference System Representation

Point Type	WKT bit == False	WKT bit == True
0-5	GeoTIFF	WKT
6-10	Error	WKT

It is considered a file error to have more than one GeoTIFF (E)VLR or more than one WKT (E)VLR in the file. A writer can append a new CRS EVLR to a file by “superseding” the existing CRS (E)VLR. Superseding is performed by changing the LAS_Spec ID of the record to “Superseded”, a new LASF_Spec defined in this release.

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)
- float (4 byte IEEE floating point format)
- double (8 byte IEEE floating point format)
- string (a variable series of 1 byte characters, ASCII encoded, null terminated)

PUBLIC HEADER BLOCK:

Table 3: 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 Record Format	unsigned char	1 byte	*
Point Data Record Length	unsigned short	2 bytes	*
Legacy Number of point records	unsigned long	4 bytes	*
Legacy 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	*
Start of first Extended Variable Length Record	unsigned long long	8 bytes	*
Number of Extended Variable Length Records	unsigned long	4 bytes	*
Number of point records	unsigned long long	8 bytes	*
Number of points by return	unsigned long long [15]	120 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 4: 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 data 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	WKT	If set, the Coordinate Reference System (CRS) is WKT. If not set, the CRS is GeoTIFF. It should not be set if the file writer wishes to ensure legacy compatibility (which means the CRS must be GeoTIFF)
5:15	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. It should be noted that the LAS Working Group does not associate any particular meaning to major or minor version number.

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 5: System Identifier

Generating Agent	**System ID**
Hardware system	String identifying hardware (e.g. “ALTM 1210”, “ALS50”, “LMS-Q680i” etc.
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” 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. For LAS 1.4 this size is 375 bytes. 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 Record Format: The point data record indicates the type of point data records contains in the file. LAS 1.4 defines types 0 through 10. These types are defined in the Point Data Record Format section of this specification.

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. The format and meaning of such “extra bytes” can (optionally) be described with an Extra Bytes VLR (see Table 24 and Table 25).

Legacy Number of point records: This field contains the total number of point records within the file if the file is maintaining legacy compatibility and the number of points is no greater than UINT32_MAX. It must be zero otherwise.

Legacy Number of points by return: These fields contain an array of the total point records per return if the file is maintaining legacy compatibility and the number of points is no greater than

UINT32_MAX. 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 the file is not maintaining legacy compatibility, each member of the array must be set to zero.

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 digits, 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 Waveform Data Package Record. Note that this will be the first byte of the Waveform Data Packet header. If no waveform records are contained within the file, this value must be zero. It should be noted that LAS 1.4 allows multiple Extended Variable Length Records (EVLR) and that the Waveform Data Packet Record is not necessarily the first EVLR in the file.

Start of First Extended Variable Length Record: This value provides the offset, in bytes, from the beginning of the LAS file to the first byte of the first EVLR.

Number of Extended Variable Length Records: This field contains the current number of EVLRs (including, if present, the Waveform Data Packet Record) that are stored in the file after the Point Data Records. This number must be updated if the number of EVLRs changes. If there are no EVLRs this value is zero.

Number of point records: This field contains the total number of point records in the file. Note that this field must always be correctly populated, regardless of legacy mode intent.

Number of points by return: These fields contain 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 fifteen returns. Note that these fields must always be correctly populated, regardless of legacy mode intent.

VARIABLE LENGTH RECORDS (VLRs):

The Public Header Block can be followed by any number of Variable Length Records (VLRs) so long as the total size does not make the start of the Point Record data inaccessible by an unsigned long (“Offset to Point Data” in the Public Header Block). The number of VLRs 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 6: 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	

Reserved: This value must be set to zero

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.

Record ID: The Record ID is dependent upon the User ID. There can be 0 to 65,535 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 65,535 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.

POINT DATA RECORDS

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 (i.e. Point Data Record Format). 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 ASCII, 0 for integers).

Point Data Record Format 6-10 have improved several aspects of the core information in the point data records, particularly support for 256 classes and the definition of a specific “Overlap” bit. While all point record formats (0 – 10) are supported in LAS 1.4, the preferred formats are 6-10.

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 7: 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 listed in Table 8 and the classification values in Table 9.

Table 8: 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 for legacy point formats (see Table 8)
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 would be a *ground* point that had been *Synthetically* collected and marked as a model *key-point*.

Table 9: ASPRS Standard LIDAR Point Classes (Point Data Record Format 0-5)

Classification Value (bits 0:4)	***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)
9	Water
10	*Reserved for ASPRS Definition*
11	*Reserved for ASPRS Definition*
12	Overlap Points[2]
13-31	*Reserved for ASPRS Definition*

[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

[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.

[2] Overlap Points are those points that were immediately culled during the merging of overlapping flight lines. In general, the *Withheld* bit should be set since these points are not subsequently classified.

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 – 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	*

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	*

Item	Format	Size	Required
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	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 and indicates the User Defined Record that describes 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 in an auxiliary WPD 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	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 returns, 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 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	

Item	**Format**	**Size**	**Required**
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	*
User Data	unsigned char	1 byte	
Scan Angle	short	2 bytes	*
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 (unless, of course, it is mandated by a particular delivery specification) 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 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 or the mirror facet changes. Note that this field has no meaning for 360° Field of View scanners (such as Mobile LIDAR scanners) and should not be set.

Classification: Classification must adhere to the following standard:

Table 17: ASPRS Standard LIDAR Point Classes (Point Data Record Formats 6-10)

Classification Value	***Meaning***
0	Created, never classified
1	Unclassified[3]
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

Scan Angle: The Scan Angle 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

[3] 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.

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.

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	*
User Data	unsigned char	1 byte	
Scan Angle	short	2 bytes	*
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	*

Item	Format	Size	Required
User Data	unsigned char	1 byte	
Scan Angle	short	2 bytes	*
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	*
User Data	unsigned char	1 byte	
Scan Angle	short	2 bytes	*
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	*
User Data	unsigned char	1 byte	
Scan Angle	short	2 bytes	*
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.

EXTENDED VARIABLE LENGTH RECORDS (EVLRs)

The Point Record Data can be followed by any number of EVLRs.

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 number of EVLRs is specified in the “Number of Extended Variable Length Records” field in the Public Header Block. The start of the first EVLR is at the file offset indicated by the “Start of first Extended Variable Length Record” in the Public Header Block. The Extended Variable Length Records must be accessed sequentially since the size of each variable length record is contained in the Extended Variable Length Record Header. Each Extended Variable Length Record Header is 60 bytes in length.

Table 22: 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	*

Item	Format	Size	Required
Record Length After Header	unsigned long long	8 bytes	*
Description	char[32]	32 bytes	

Note: As with the VRL, the Reserved field must be set to zero

DEFINED VARIABLE LENGTH RECORDS:

LAS 1.4 defines Variable Length Records (VLR) and Extended Variable Length Records (EVLR). A writer who wishes to maintain legacy compatibility must use only VLRs (except for internally stored waveform data). A writer who is not concerned with a legacy LAS reader having access to a VLR can elect to use an EVLR, even for predefined VLRs such as Coordinate Reference System (CRS) information. This ability is useful, for example, when a user wishes to update information normally contained within a VLR without the need of rewriting the point data.

A new LASF_Spec “Superseded” (value 7) has been defined to allow a writer to indicate that a VLR should no longer be used. For example, if a user appends a new WKT EVLR to a file, the existing WKT VLR should have its LASF Spec number changed to Superseded to indicate that it is no longer in use.

Coordinate Reference System Information

The Coordinate Reference System (CRS) information for the point data is required for all data. The CRS information will be placed in Variable Length Records or Extended Variable Length Records (note that if the writer wishes to maintain legacy compatibility, then GeoTIFF in VLRs must be used). The CRS is represented by either GeoTIFF or Well Know Text as indicated by the WKT Global Encoding bit. Point Record Formats 0-5 can use either GeoTIFF or WKT (but not both simultaneously). Point Record Formats 6-10 must use WKT.

Georeferencing Information using 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). This specification may be found at www.opengeospatial.org/standards/ct. As there are a few dialects of WKT, please note that 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 record 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.

Georeferencing Information using GeoTIFF

The GeoTIFF specification is defined by http://www.remotesensing.org/geotiff/geotiff.html.

GeoTIFF georeferencing for the LAS formats uses the same 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 23: 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.

Other Specification Defined VLRs

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];
};
```

TEXT AREA DESCRIPTION: (optional)
User ID: LASF_Spec
Record ID: 3

This VLR/EVLR is used for providing a textual description of the content of the LAS file. It is a null terminated, free-form ASCII string.

EXTRA BYTES: (optional)

User ID: LASF_Spec
Record ID: 4
Record Length after Header: n records x 192 bytes

The Extra Bytes VLR provides a mechanism whereby additional information can be added to the end of a standard Point Record. This VLR has been added to LAS 1.4 to formalize a process that has been used in prior versions of LAS. It is envisioned that software that is not cognizant of the meaning of the extra bytes will simply copy these bytes when manipulating files.

This record is only needed for LAS files where points contain user-defined “extra bytes”. This happens when the point record size is set to a larger value than required by the point type. For example, when a LAS file that contains point type 1 has a point record size of 32 instead of 28 there are 4 “extra bytes”. The Extra Bytes VLR contains a simple description of the type and the meaning of these “extra bytes” so they can be accessed in a consistent manner across applications. The 4 “extra bytes” could, for example, be of data_type 9 - a floating point value - that specifies an "echo width" for each return. In this case there would be one EXTRA_BYTES struct in the payload of this VLR. An example with two EXTRA_BYTES struct in the payload are 14 "extra bytes" that have data type 29 - a floating point vector of length 3 - followed by data type 3 - an unsigned short - that specify the "laser pulse direction" and a "normalized reflectivity".

The "extra bytes" are made accessible via a unique name. The "name" field distinguishes the additional point attributes that a LIDAR software may add to the points in a LAS file so they can be accessed later in a consistent manner by another software. Descriptive names such as "normalized reflectivity", "echo width", or "shading normal" are encouraged. The use of generic names such as "variable1" or "temp1" is discouraged.

The bit mask in the options field 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 “extra bytes” are to be multiplied and translated to compute their actual value, and whether there is a special value that should be interpreted as NO_DATA. By default all bits are zero which means that the values in the corresponding fields are to be disregarded.

If the selected data_type is less than 8 bytes, the no_data, min, and max field should be upcast into 8-byte storage. For any float these 8 bytes would be upcast to a double, for any unsigned char, unsigned short, or unsigned long they would be upcast to an unsigned long long and for any char, short, or long, they would be upcast to 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[32];            // 32 bytes
        unsigned char        unused[4];           // 4 bytes
        anytype              no_data[3];          // 24 = 3*8 bytes
        anytype              min[3];              // 24 = 3*8 bytes
        anytype              max[3];              // 24 = 3*8 bytes
        double               scale[3];            // 24 = 3*8 bytes
        double               offset[3];           // 24 = 3*8 bytes
        char                 description[32];     // 32 bytes
};                                                // total of 192 bytes
```

The “reserved” field and the "unused" field must be set to zero. Any unused "no_data", "min", "max", "scale", or "offset" fields must be set to zero. Any unused characters in the "name" or "description" fields must be set to zero.

Table 24: Values for “data_type” Field

Value	***Meaning***	***Size***
0	undocumented extra bytes	specified by value in “options” field
1	unsigned char	1 byte
2	Char	1 byte
3	unsigned short	2 bytes
4	Short	2 bytes
5	unsigned long	4 bytes
6	Long	4 bytes
7	unsigned long long	8 bytes
8	long long	8 bytes
9	Float	4 bytes
10	Double	8 bytes
11	unsigned char[2]	2 byte
12	char[2]	2 byte
13	unsigned short[2]	4 bytes
14	short[2]	4 bytes
15	unsigned long[2]	8 bytes
16	long[2]	8 bytes
17	unsigned long long[2]	16 bytes
18	long long[2]	16 bytes
19	float[2]	8 bytes
20	double[2]	16 bytes
21	unsigned char[3]	3 byte

Value	*Meaning*	*Size*
22	char[3]	3 byte
23	unsigned short[3]	6 bytes
24	short[3]	6 bytes
25	unsigned long[3]	12 bytes
26	long[3]	12 bytes
27	unsigned long long[3]	24 bytes
28	long long[3]	24 bytes
29	float[3]	12 bytes
30	double[3]	24 bytes

Table 25: Encoding of "options" Bit Field

Bit	*Field Name*	*Description*
0	no_data_bit	If set the no_data value is relevant.
1	min_bit	If set the min value is relevant
2	max_bit	If set the max value is relevant
3	scale_bit	If set each value should be multiplied by the corresponding scale value (before applying the offset)
4	offset_bit	If set each value should be translated by the corresponding offset value (after applying the scaling)

A LAS file contains "undocumented extra bytes" when there are "extra bytes" but when there is no Extra Bytes VLR that describes them or when there are more "extra bytes" than described in an existing Extra Bytes VLR.

When adding an "Extra Bytes" VLR to a LAS file that contains "undocumented extra bytes" they must be "described" as data_type == 0 with the options bit field storing the number of undocumented bytes.

A LAS file has an "extra bytes mismatch" if the Extra Bytes VLR describes more "extra bytes" than each LAS point actually has. The occurrence of an "extra bytes mismatch" renders the Extra Bytes VLR invalid.

SUPERSEDED: (optional)

User ID:	LASF_Spec
Record ID:	7

This LASF Record ID is used to negate an existing VLR/EVLR when rewriting the file (to remove the undesired VLR/EVLR).. It is used, for example, when updating a record such as projection information where a new EVLR is appended to the end of the LAS file. The existing VLR which has been superseded must be marked with the SUPERSEDED Record ID.

WAVEFORM PACKET DESCRIPTOR: (REQUIRED WHEN USING POINT FORMATS 4, 5, 9 OR 10)

User ID:	LASF_Spec
Record ID:	n: where n > 99 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 26: 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 bytes	*
Digitizer Offset	double	8 bytes	*

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

EXTENDED VARIABLE LENGTH RECORDS:

WAVEFORM DATA PACKETS: (REQUIRED INTERNALLY/EXTERNALLY 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.