THE IMAGING & GEOSPATIAL INFORMATION SOCIETY

LAS Specification 1.4 - R16 DRAFT

Release Information:

Version Approved – November 2011
Revision date – 09 July 2019

PDF build date – 19 March 2022
GitHub commit – 29a5b5628099314021cab44173710a637b6c1f88
GitHub repo – https://github.com/ASPRSorg/LAS

Published by:

The American Society for Photogrammetry & Remote Sensing
425 Barlow Place, Suite 210
Bethesda, Maryland 20814-2160
Voice: 301-493-0290
Fax: 225-408-4422
Web: www.asprs.org

CONTENTS:

1 Introduction

1.1 Purpose, Scope, and Applicability

The LASer (LAS) file is intended to contain point cloud data records, including those derived from lidar or other sources. The data will commonly be put into this format from software (e.g., provided by hardware vendors), which combines GPS, IMU, and laser pulse range data to produce points with X, Y, and Z coordinates. The purpose of LAS is to provide an open format that allows different hardware and software tools to exchange point cloud data in a common format.

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

1.1.1 LAS 1.4 Revision History

Summary of LAS 1.4 revisions (GitHub Issue numbers included when applicable):

- 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".
- R13 - Added Domain Profile Section (July 2013).
- R14 - Multiple updates (March 2019):
  - Aesthetic changes from migration to GitHub.
  - Multiple capitalization & typo corrections.
  - Updated ASPRS contact info. (I-30[5])
  - Additional standard classifications 19-22 for PDRFs 6-10: (I-11[6], I-26[7])
    - Class 19 – Overhead Structure in PDRFs 6-10.
    - Class 20 – Ignored Ground.
    - Class 21 – Snow.
    - Class 22 – Temporal Exclusion.
  - Added OGC endorsement. (I-31[8])
  - Added minimum PDRF sizes to attribute tables. (I-47[9])
  - Section reorganization: (I-57[10])
    - Addition of Table of Contents with section numbers. (c.f. I-27[11], I-49[12])
    - Divided Defined Variable Length Records section into Coordinate Reference System VLRs section (s3) and Other Specification Defined VLRs (s4).

[5] https://github.com/ASPRSorg/LAS/issues/30
[6] https://github.com/ASPRSorg/LAS/issues/11
[7] https://github.com/ASPRSorg/LAS/issues/26
[8] https://github.com/ASPRSorg/LAS/issues/31
[9] https://github.com/ASPRSorg/LAS/issues/47
[10] https://github.com/ASPRSorg/LAS/issues/57
[11] https://github.com/ASPRSorg/LAS/issues/27
[12] https://github.com/ASPRSorg/LAS/issues/49

        * Expanded EVLR discussion in Legacy Compatibility section (s2.1) and moved Legacy Compatibility section to EVLR definition (now s2.7.1).
        * Swapped order of LAS 1.4 Revision History (now s1.1.1) and LAS 1.4 Additions (now s1.1.2).
        * Rearranged paragraphs in Extra Bytes VLR description.
    - Deprecated "tuple" and "triple" extra byte data types. (I-1[13])
        * Added explanation and example of implicit arrays from descriptor names.
    - Clarified that ExtraByte min/max should be an untransformed value. (I-4[14])
    - Clarified that Legacy Point Counts should be set to zero if using non-legacy PDRFs. (I-12[15])
    - Clarified Full Waveform descriptions and added wiki link. (I-9[16])
    - Renamed X(t), Y(t), and Z(t) from waveform packets to Parametric dx/dy/dz.
    - PDRF9 now correctly requires Scanner Channel like other PDRFs. (I-29[17])
    - Clarified origin date/time for Adjusted Standard GPS Time. (I-40[18])
    - Clarified null-termination of fixed-length `char` arrays, especially VLR Description. (I-46[19])
    - Clarified relationship between FileSourceID and PointSourceID. (I-59[20])
    - Added language to support technologies other than conventional linear-mode lidar scanners. (I-35[21])
        * Clarified and renamed Synthetic Return Numbers Global Encoding bit.
        * Clarified Synthetic point classification flag.
        * Clarified validity of zero-value PointSourceID.
        * Unified Return Number and Number of Returns descriptions between legacy and non-legacy PDRFs.
        * Clarified Scan Direction and Edge of Flight Line Flags for non-rotational systems.
    - Added wiki link for Project ID examples. (I-38[22])
- R15 - Errata and typo corrections for R14 (July 2019):
    - Minor editorial punctuation corrections. (I-78[23])
    - Fixed unintended reordering of Min/Max XYZ in R14 LAS header. (I-79[24])
    - Added missing Scanner Channel field in note about PDRF6-10 bit field. (I-80[25])
- R16 - DRAFT
    - Clarified uniformity of GPS Time for all returns of a pulse. (I-81[26])

[13] https://github.com/ASPRSorg/LAS/issues/1
[14] https://github.com/ASPRSorg/LAS/issues/4
[15] https://github.com/ASPRSorg/LAS/issues/12
[16] https://github.com/ASPRSorg/LAS/issues/9
[17] https://github.com/ASPRSorg/LAS/issues/29
[18] https://github.com/ASPRSorg/LAS/issues/40
[19] https://github.com/ASPRSorg/LAS/issues/46
[20] https://github.com/ASPRSorg/LAS/issues/59
[21] https://github.com/ASPRSorg/LAS/issues/35
[22] https://github.com/ASPRSorg/LAS/issues/38
[23] https://github.com/ASPRSorg/LAS/issues/78
[24] https://github.com/ASPRSorg/LAS/issues/79
[25] https://github.com/ASPRSorg/LAS/issues/80
[26] https://github.com/ASPRSorg/LAS/issues/81

  - Added example usage of Overlap bit. (I-5[27])
  - Reworded mandatory zero Classification for Synthetic PDRF0-5. (I-86[28])
  - Added Byte Offset columns to Header, VLR, EVLR, and PDRF tables. (I-55[29])
  - Clarified how and when certain header fields should be zero-filled. (I-101[30])
  - More closely aligned data types with C99 definitions. (I-115[31])

For detailed information on changes in revisions R14 and newer, review the inline differencing provided on the GitHub page[32].

1.1.2 Comparison of LAS 1.4 to Previous Versions

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 indicate points in the overlap region while maintaining the class definition.
  - Addition of an (optional) *Extra Bytes* Variable Length Record to describe "extra bytes" stored with each point.
- Other minor changes:
  - Added definitions for "LAS Domain Profile" and "LAS Domain Profile Description".
  - Added links to official LAS wiki: https://github.com/ASPRSorg/LAS/wiki

[27] https://github.com/ASPRSorg/LAS/issues/5
[28] https://github.com/ASPRSorg/LAS/issues/86
[29] https://github.com/ASPRSorg/LAS/issues/55
[30] https://github.com/ASPRSorg/LAS/issues/101
[31] https://github.com/ASPRSorg/LAS/issues/115
[32] https://github.com/ASPRSorg/LAS

1.2 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")).

1.3 Authority

1.3.1 ASPRS

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:

- Online at https://github.com/ASPRSorg/LAS
- By phone at 301-493-0290
- By email at las@asprs.org or asprs@asprs.org
- By mail at 8550 United Plaza Blvd, Suite 1001 Baton Rouge, LA 70809

1.3.2 OGC

LAS has been recognized by the Open Geospatial Consortium (OGC[33]) in 2018 as an OGC Community Standard. The OGC version of the document with forward material about standards that LAS references and its status within the standard body can be found at https://portal.opengeospatial.org/files/17-030r1.

Future recognition and activity on OGC referencing activities of LAS can be followed at http://www.opengeospatial.org/standards/community.

[33] http://www.opengeospatial.org

2 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 (EVLRs) allow a higher payload than VLRs and have the advantage that they can be appended to the end of a LAS file. This allows, for example, adding 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 (VLRs)
Point Data Records
Extended Variable Length Records (EVLRs)

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

2.1 Legacy Compatibility (LAS 1.1 - LAS 1.3)

LAS 1.4 moves the file specification from a 32 bit file structure (maximum value of $2^{32} - 1 \equiv 4,294,967,295 \equiv$ 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 is 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. However, 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.

LAS 1.4 introduced the option to define Variable Length Records (VLRs) as Extended Variable Length Records (EVLRs) instead. A LAS 1.4 file writer wishing to maintain backward compatibility must use only VLRs. See the *Legacy Compatibility for EVLRs* section for more information.

2.2 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 below:

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.

2.3 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”)), as defined in the standard header <`stdint.h`>.

- int8_t (1 byte)
- uint8_t (1 byte)
- int16_t (2 bytes)
- uint16_t (2 bytes)
- int32_t (4 bytes)
- uint32_t (4 bytes)
- int64_t (8 bytes)
- uint64_t (8 bytes)
- float (4-byte `binary32` IEEE floating point format)
- double (8-byte `binary64` IEEE floating point format)
- string (a variable-length array of 1-byte characters, ASCII-encoded, null-terminated, contained in a fixed-length `char` array, where `char` must be equivalent to either `int8_t` or `uint8_t`)

Warning: Fixed-length `char` arrays will not be null-terminated if all bytes are utilized. Examples include the System Identifier and Generating Software in the LAS Header, the User ID or Description in the Variable Length Record, and the Name of an Extra Byte Descriptor.

2.4 Public Header Block

Table 3: Public Header Block

Item	Format	Byte Offset	Size	Required
File Signature (“LASF”)	char[4]	0	4 bytes	yes
File Source ID	uint16_t	4	2 bytes	yes
Global Encoding	uint16_t	6	2 bytes	yes
Project ID - GUID Data 1	uint32_t	8	4 bytes	
Project ID - GUID Data 2	uint16_t	12	2 bytes	
Project ID - GUID Data 3	uint16_t	14	2 bytes	
Project ID - GUID Data 4	uint8_t[8]	16	8 bytes	
Version Major	uint8_t	24	1 byte	yes
Version Minor	uint8_t	25	1 byte	yes
System Identifier	char[32]	26	32 bytes	yes
Generating Software	char[32]	58	32 bytes	yes
File Creation Day of Year	uint16_t	90	2 bytes	yes
File Creation Year	uint16_t	92	2 bytes	yes
Header Size	uint16_t	94	2 bytes	yes
Offset to Point Data	uint32_t	96	4 bytes	yes
Number of Variable Length Records	uint32_t	100	4 bytes	yes
Point Data Record Format	uint8_t	104	1 byte	yes
Point Data Record Length	uint16_t	105	2 bytes	yes
Legacy Number of Point Records	uint32_t	107	4 bytes	yes
Legacy Number of Point by Return	uint32_t[5]	111	20 bytes	yes
X Scale Factor	double	131	8 bytes	yes
Y Scale Factor	double	139	8 bytes	yes
Z Scale Factor	double	147	8 bytes	yes
X Offset	double	155	8 bytes	yes
Y Offset	double	163	8 bytes	yes
Z Offset	double	171	8 bytes	yes
Max X	double	179	8 bytes	yes
Min X	double	187	8 bytes	yes
Max Y	double	195	8 bytes	yes
Min Y	double	203	8 bytes	yes
Max Z	double	211	8 bytes	yes
Min Z	double	219	8 bytes	yes
Start of Waveform Data Packet Record	uint64_t	227	8 bytes	yes
Start of First Extended Variable Length Record	uint64_t	235	8 bytes	yes
Number of Extended Variable Length Records	uint32_t	243	4 bytes	yes

continues on next page

Table 3 – continued from previous page

Number of Point Records	uint64_t	247	8 bytes	yes
Number of Points by Return	uint64_t[15]	255	120 bytes	yes

Note: 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 from 0 to 65,535. A value of zero is interpreted to mean that an ID has not been assigned, which is the norm for a LAS file resulting from an aggregation of multiple independent sources (e.g., a tile merged from multiple swaths).

Note that this scheme allows a project to contain up to 65,535 unique sources. Example sources can include a data repository ID or an original collection of temporally consistent data such as a flight line or sortie number for airborne systems, a route number for mobile systems, or a setup identifier for static systems.

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. The origin of standard GPS Time is defined as midnight of the morning of January 6, 1980.
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	Synthetic Return Numbers	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, or when simulating return numbers for a system not directly supporting multiple returns.
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 (0).

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.

Note: Example implementations of representing the Project ID fields as a GUID can be found on the official LAS wiki: https://github.com/ASPRSorg/LAS/wiki

---

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 of up to 32 characters identifying the operation

If the character data is less than 32 characters, the remaining data must be null.

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 a uint16_t, 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. If any software adds/removes data to/from the Variable Length Records, then this offset value must be updated.

Number of Variable Length Records

This field contains the current number of VLRs that are stored in the file preceding the Point Data Records. If the number of VLRs changes, then this number must be updated.

This field is unrelated to the number of EVLRs appended to the file.

Point Data Record Format

The point data record indicates the type of point data records contained in the file. LAS 1.4 defines types 0 through 10. These types are defined in the *Point Data Records* 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.

Legacy Number of Point Records

This field contains the total number of point records within the file if the file is maintaining legacy compatibility, the number of points is no greater than UINT32_MAX, and the Point Data Record Format is less than 6. Otherwise, it must be set to zero.

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, the number of points is no greater than UINT32_MAX, and the Point Data Record Format is less than 6. Otherwise, each member of the array must be set to zero.

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.

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 int32_t 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 Offsets

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.

For example, to compute a given X coordinate from the point record, the point record X is multiplied by the X scale factor and then added to the X offset.

$$X_{coordinate} = (X_{record} * X_{scale}) + X_{offset} \quad (1)$$
$$Y_{coordinate} = (Y_{record} * Y_{scale}) + Y_{offset} \quad (2)$$
$$Z_{coordinate} = (Z_{record} * Z_{scale}) + Z_{offset} \quad (3)$$

Max and Min X, Y, and 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. If there are no point records in the file, these values must be set to zero.

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 or they are stored externally, this value must be zero. It should be noted that LAS 1.4 allows multiple Extended Variable Length Records (EVLRs) 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. If any software adds/removes data to/from the Variable Length Records or Point Records, then this offset value must be updated.

If there are no EVLRs, this value must be zero.

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. If the number of EVLRs changes, then this number must be updated. If there are no EVLRs this value is zero.

If there are no EVLRs, this value must be 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.

2.5 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 a uint32_t ("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**	**Byte Offset**	**Size**	**Required**
Reserved	uint16_t	0	2 bytes	
User ID	char[16]	2	16 bytes	yes
Record ID	uint16_t	18	2 bytes	yes
Record Length After Header	uint16_t	20	2 bytes	yes
Description	char[32]	22	32 bytes	

Reserved

This value must be set to zero.

User ID

The User ID field is ASCII character data that identifies the user that 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 text description of the data. Any remaining unused characters must be null.

2.6 Point Data Records

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 Record Formats (PDRFs) 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 PDRFs (0-10) are supported in LAS 1.4, the preferred formats are 6-10. PDRFs 0-5 are therefore designated as the "legacy" point formats.

Required Point Attributes

Point attributes that are "Required" must be populated with relevant values whenever possible. If unused, point attributes that are not "Required" must be set to the equivalent of zero for the data type (i.e., 0.0 for floating types, null for ASCII, 0 for integers).

If a "Required" point attribute cannot apply to a particular technology (e.g., Scan Direction for a passive sensor) then the attribute must be set to a default value as directed. This default value is zero if unspecified in the attribute description.

Aggregate Model Systems

Points derived from multiple observations in an aggregate model rather than a direct measurement system should be assigned valid values using a consistent scheme for a given dataset. For example, in the case of a photogrammetrically derived point cloud, the Point Source ID, GPS Time, and Scan Angle could be assigned to a point based on the value associated with the most recent photograph from which the point was derived. Example systems to which this recommendation applies include photogrammetrically derived point clouds and Geiger-mode lidar processed with a consensus model. These systems are hereafter collectively denoted as "Aggregate Model Systems."

2.6.1 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	Byte Offset	Size	Required
X	int32_t	0	4 bytes	yes
Y	int32_t	4	4 bytes	yes
Z	int32_t	8	4 bytes	yes
Intensity	uint16_t	12	2 bytes	no
Return Number	3 bits (bits 0-2)	14	3 bits	yes
Number of Returns (Given Pulse)	3 bits (bits 3-5)		3 bits	yes
Scan Direction Flag	1 bit (bit 6)		1 bit	yes
Edge of Flight Line	1 bit (bit 7)		1 bit	yes
Classification	uint8_t	15	1 byte	yes
Scan Angle Rank (-90 to +90) – Left Side	int8_t	16	1 byte	yes
User Data	uint8_t	17	1 byte	no
Point Source ID	uint16_t	18	2 bytes	yes
Minimum PDRF Size[1]		*20 bytes*		

X, Y, and Z

The X, Y, and Z values are stored as int32_t 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. If Intensity is not included, this value must be set to zero.

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). This normalization is required to ensure that data from different sensors can be correctly merged.

For systems based on technology other than pulsed lasers, Intensity values may represent estimated relative reflectivity, rather than a direct measurement of pulse return magnitude, and may be derived from multiple sources.

Note: 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

[1] Recall that the Point Data Record Size can be greater than the minimum required for a PDRF. These "extra bytes" follow the standard Point Record fields and are described in the *Extra Bytes* VLR section.

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. The Return Number must be between 1 and the Number of Returns, inclusive.

For systems unable to record multiple returns, the Return Number should be set to one, unless it is synthetically derived and the Synthetic Return Number Global Encoding bit is set.

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 five returns.

For systems unable to record multiple returns, the Number of Returns should be set to one, unless it is synthetically derived and the Synthetic Return Number Global Encoding bit is set.

Scan Direction Flag

The Scan Direction Flag denotes the direction in 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).

For *Aggregate Model Systems* or if the measurement system does not include a rotational component, the Scan Direction Flag should be set to zero.

Edge of Flight Line Flag

The Edge of Flight Line Flag 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 *Aggregate Model Systems* or 360 degree Field of View scanners (e.g., terrestrial lidar scanners). In these cases, the Edge of Flight Line Flag should be set to zero.

Classification

This field represents the "class" attributes of a point. 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. If a point has never been classified, the lowest five bits must be set to zero.

Table 8: Classification Bit Field Encoding (Point Data Record Formats 0-5)

Bit	Field Name	Description
0:4	Classification	Standard ASPRS classification from 0 to 31 as defined in the classification table for legacy point formats (see *Reserved Point Classes*).
5	Synthetic	If set, this point was created by a technique other than direct observation such as digitized from a photogrammetric stereo model or by traversing a waveform. Point attribute interpretation might differ from non-Synthetic points. Unused attributes must be set to the appropriate default value.
6	Key-Point	If set, this point is considered to be a model key-point and therefore 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: 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*.

Reserved Point Classes

Classification must adhere to the following standard:

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

Classification Value (Bits 0:4)	**Meaning**
0	Created, Never Classified
1	Unclassified[2]
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[3]
13-31	*Reserved for ASPRS Definition*

Note: 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 integer 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.

For *Aggregate Model Systems*, the Scan Angle Rank should be set to zero unless assigned from a component measurement.

User Data

This field may be used at the user's discretion.

[2] 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.

[3] 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.

Point Source ID

This value indicates the source from which this point originated. A source is typically defined as a grouping of temporally consistent data, such as a flight line or sortie number for airborne systems, a route number for mobile systems, or a setup identifier for static systems. Valid values for this field are 1 to 65,535 inclusive. Zero is reserved as a convenience to system implementers.

For *Aggregate Model Systems*, the Point Source ID should be set to one (1) unless assigned from a component measurement.

2.6.2 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	Byte Offset	Size	Required
X	int32_t	0	4 bytes	yes
Y	int32_t	4	4 bytes	yes
Z	int32_t	8	4 bytes	yes
Intensity	uint16_t	12	2 bytes	no
Return Number	3 bits (bits 0-2)	14	3 bits	yes
Number of Returns (Given Pulse)	3 bits (bits 3-5)		3 bits	yes
Scan Direction Flag	1 bit (bit 6)		1 bit	yes
Edge of Flight Line	1 bit (bit 7)		1 bit	yes
Classification	uint8_t	15	1 byte	yes
Scan Angle Rank (-90 to +90) – Left Side	int8_t	16	1 byte	yes
User Data	uint8_t	17	1 byte	no
Point Source ID	uint16_t	18	2 bytes	yes
GPS Time	double	20	8 bytes	yes
Minimum PDRF Size		*28 bytes*		

GPS Time

The GPS Time is the double floating point time tag value at which the point was observed. It is GPS Week Time if the *Global Encoding* low bit is clear and Adjusted Standard GPS Time if the bit is set.

It is intended that each return of a given pulse would have an identical GPS Time. In the example of pulsed LiDAR systems, the GPS Time would be the time at which the originating pulse was emitted, not the time at which each return was recorded.

For *Aggregate Model Systems*, the GPS Time should be set to zero unless assigned from a component measurement.

2.6.3 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 point using ancillary data, typically from a camera.

Table 11: Point Data Record Format 2

Item	**Format**	**Byte Offset**	**Size**	**Required**
X	int32_t	0	4 bytes	yes
Y	int32_t	4	4 bytes	yes
Z	int32_t	8	4 bytes	yes
Intensity	uint16_t	12	2 bytes	no
Return Number	3 bits (bits 0-2)	14	3 bits	yes
Number of Returns (Given Pulse)	3 bits (bits 3-5)		3 bits	yes
Scan Direction Flag	1 bit (bit 6)		1 bit	yes
Edge of Flight Line	1 bit (bit 7)		1 bit	yes
Classification	uint8_t	15	1 byte	yes
Scan Angle Rank (-90 to +90) – Left Side	int8_t	16	1 byte	yes
User Data	uint8_t	17	1 byte	no
Point Source ID	uint16_t	18	2 bytes	yes
Red	uint16_t	20	2 bytes	yes
Green	uint16_t	22	2 bytes	yes
Blue	uint16_t	24	2 bytes	yes
Minimum PDRF Size		*26 bytes*		

Red, Green, and Blue

The Red, Green, and Blue image channel values associated with this point.

Note: The Red, Green, and 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.

2.6.4 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**	**Byte Offset**	**Size**	**Required**
X	int32_t	0	4 bytes	yes
Y	int32_t	4	4 bytes	yes
Z	int32_t	8	4 bytes	yes
Intensity	uint16_t	12	2 bytes	no
Return Number	3 bits (bits 0-2)	14	3 bits	yes
Number of Returns (Given Pulse)	3 bits (bits 3-5)		3 bits	yes
Scan Direction Flag	1 bit (bit 6)		1 bit	yes
Edge of Flight Line	1 bit (bit 7)		1 bit	yes
Classification	uint8_t	15	1 byte	yes
Scan Angle Rank (-90 to +90) – Left Side	int8_t	16	1 byte	yes
User Data	uint8_t	17	1 byte	no
Point Source ID	uint16_t	18	2 bytes	yes
GPS Time	double	20	8 bytes	yes
Red	uint16_t	28	2 bytes	yes
Green	uint16_t	30	2 bytes	yes
Blue	uint16_t	32	2 bytes	yes
Minimum PDRF Size		*34 bytes*		

2.6.5 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	Byte Offset	Size	Required
X	int32_t	0	4 bytes	yes
Y	int32_t	4	4 bytes	yes
Z	int32_t	8	4 bytes	yes
Intensity	uint16_t	12	2 bytes	no
Return Number	3 bits (bits 0-2)	14	3 bits	yes
Number of Returns (Given Pulse)	3 bits (bits 3-5)		3 bits	yes
Scan Direction Flag	1 bit (bit 6)		1 bit	yes
Edge of Flight Line	1 bit (bit 7)		1 bit	yes
Classification	uint8_t	15	1 byte	yes
Scan Angle Rank (-90 to +90) – Left Side	int8_t	16	1 byte	yes
User Data	uint8_t	17	1 byte	no
Point Source ID	uint16_t	18	2 bytes	yes
GPS Time	double	20	8 bytes	yes
Wave Packet Descriptor Index	uint8_t	28	1 byte	yes
Byte Offset to Waveform Data	uint64_t	29	8 bytes	yes
Waveform Packet Size in Bytes	uint32_t	37	4 bytes	yes
Return Point Waveform Location	float	41	4 bytes	yes
Parametric dx	float	45	4 bytes	yes
Parametric dy	float	49	4 bytes	yes
Parametric dz	float	53	4 bytes	yes
Minimum PDRF Size		*57 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 Point Record. 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 Point Record.

Byte Offset to Waveform Data

The waveform packet data are stored in the LAS file in an Extended Variable Length Record or in an auxiliary *.wdp file. The Byte Offset represents the location of the start of this Point Record's waveform packet within the waveform data variable length record (or external file) relative to the beginning of the *Waveform Data Packets* 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 Data

for waveform packets stored within the LAS file and

Byte Offset to Waveform Data

for data stored in an auxiliary *.wdp 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 Waveform Location

The temporal offset in picoseconds (10^{-12}) from the arbitrary "anchor point" to the location within the waveform packet for this Point Record.

Parametric dx, dy, dz

These parameters define a parametric line equation for extrapolating points along the associated waveform. The position along the wave is given by:

$$X = X_0 + t * dx \quad (4)$$

$$Y = Y_0 + t * dy \quad (5)$$

$$Z = Z_0 + t * dz \quad (6)$$

where (X, Y, Z) is the spatial position of a derived point, (X_0, Y_0, Z_0) is the position of the "anchor" point, and t is the time, in picoseconds, relative to the anchor point.

The anchor point is an arbitrary location at the origin of the associated waveform – i.e. $t = 0$ at the anchor point – with coordinates defined by:

$$X_0 = X_P + L * dx \quad (7)$$

$$Y_0 = Y_P + L * dy \quad (8)$$

$$Z_0 = Z_P + L * dz \quad (9)$$

where (X_P, Y_P, Z_P) is this Point Record's transformed position (as a double) and L is this Point Record's Return Point Waveform Location.

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.

Note: Users seeking further clarity regarding LAS waveform encoding are encouraged to learn more on the official LAS wiki: https://github.com/ASPRSorg/LAS/wiki

---

2.6.6 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**	**Byte Offset**	**Size**	**Required**
X	int32_t	0	4 bytes	yes
Y	int32_t	4	4 bytes	yes
Z	int32_t	8	4 bytes	yes
Intensity	uint16_t	12	2 bytes	no
Return Number	3 bits (bits 0-2)	14	3 bits	yes
Number of Returns (Given Pulse)	3 bits (bits 3-5)		3 bits	yes
Scan Direction Flag	1 bit (bit 6)		1 bit	yes
Edge of Flight Line	1 bit (bit 7)		1 bit	yes
Classification	uint8_t	15	1 byte	yes
Scan Angle Rank (-90 to +90) – Left Side	int8_t	16	1 byte	yes
User Data	uint8_t	17	1 byte	no
Point Source ID	uint16_t	18	2 bytes	yes
GPS Time	double	20	8 bytes	yes
Red	uint16_t	28	2 bytes	yes
Green	uint16_t	30	2 bytes	yes
Blue	uint16_t	32	2 bytes	yes
Wave Packet Descriptor Index	uint8_t	34	1 byte	yes
Byte Offset to Waveform Data	uint64_t	35	8 bytes	yes
Waveform Packet Size in Bytes	uint32_t	43	4 bytes	yes
Return Point Waveform Location	float	47	4 bytes	yes
Parametric dx	float	51	4 bytes	yes
Parametric dy	float	55	4 bytes	yes
Parametric dz	float	59	4 bytes	yes
Minimum PDRF Size		*63 bytes*		

2.6.7 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**	**Byte Offset**	**Size**	**Re-quired**
X	int32_t	0	4 bytes	yes
Y	int32_t	4	4 bytes	yes
Z	int32_t	8	4 bytes	yes
Intensity	uint16_t	12	2 bytes	no
Return Number	4 bits (bits 0-3)	14	4 bits	yes
Number of Returns (Given Pulse)	4 bits (bits 4-7)		4 bits	yes
Classification Flags	4 bits (bits 0-3)	15	4 bits	no
Scanner Channel	2 bits (bits 4-5)		2 bits	yes
Scan Direction Flag	1 bit (bit 6)		1 bit	yes
Edge of Flight Line	1 bit (bit 7)		1 bit	yes
Classification	uint8_t	16	1 byte	yes
User Data	uint8_t	17	1 byte	no
Scan Angle	int16_t	18	2 bytes	yes
Point Source ID	uint16_t	20	2 bytes	yes
GPS Time	double	22	8 bytes	yes
Minimum PDRF Size		*30 bytes*		

Note: The following six fields (Return Number, Number of Returns, Classification Flags, Scanner Channel, 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.

For systems unable to record multiple returns, the Return Number should be set to one, unless it is synthetically derived and the Synthetic Return Number Global Encoding bit is set.

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.

For systems unable to record multiple returns, the Number of Returns should be set to one, unless it is synthetically derived and the Synthetic Return Number Global Encoding bit is set.

Classification Flags

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

Table 16: Classification Bit Field Encoding (Point Data Record Formats 6-10)

Bit	Field Name	Description
0	Synthetic	If set, this point was created by a technique other than direct observation such as digitized from a photogrammetric stereo model or by traversing a waveform. Point attribute interpretation might differ from non-Synthetic points. Unused attributes must be set to the appropriate default value.
1	Key-Point	If set, this point is considered to be a model key-point and therefore 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 required by a specification other than this document) but allows Classification of overlap points to be preserved. For example, this may be useful for designating Ground points that are valid but are not needed to meet coverage or density requirements.

Note: 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).

For *Aggregate Model Systems*, the Channel should be set to zero unless assigned from a component measurement.

Scan Direction Flag

The Scan Direction Flag denotes the direction in 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).

For *Aggregate Model Systems* or if the measurement system does not include a rotational component, the Scan Direction Flag should be set to zero.

Edge of Flight Line Flag

The Edge of Flight Line Flag 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 *Aggregate Model Systems* or 360 degree Field of View scanners (e.g., terrestrial lidar scanners). In these cases, the Edge of Flight Line Flag should be set to zero.

Classification

Classification must adhere to the following standard:

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

Value	Meaning	Notes
0	Created, Never Classified	See note[4]
1	Unclassified	
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., insulators
17	Bridge Deck	
18	High Noise	
19	Overhead Structure	e.g., conveyors, mining equipment, traffic lights
20	Ignored Ground	e.g., breakline proximity
21	Snow	
22	Temporal Exclusion	Features excluded due to changes over time between data sources – e.g., water levels, landslides, permafrost
23-63	*Reserved*	
64-255	User Definable	

Scan Angle

The Scan Angle is an int16_t that represents the rotational position of the emitted laser pulse with respect to the vertical dimension of the coordinate system of the data. Down in the data coordinate system is the 0.0 position. Each increment represents 0.006 degrees. Counter-clockwise rotation, as viewed from the rear of the sensor, facing in the along-track (positive trajectory) direction, is positive. The maximum value in the positive sense is 30,000 (180 degrees which is up in the

[4] 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.

coordinate system of the data). The maximum value in the negative direction is -30,000 which is also directly up.

For *Aggregate Model Systems*, the Scan Angle should be set to zero unless assigned from a component measurement.

2.6.8 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 point using ancillary data, typically from a camera.

Table 18: Point Data Record Format 7

Item	**Format**	**Byte Offset**	**Size**	**Required**
X	int32_t	0	4 bytes	yes
Y	int32_t	4	4 bytes	yes
Z	int32_t	8	4 bytes	yes
Intensity	uint16_t	12	2 bytes	no
Return Number	4 bits (bits 0-3)	14	4 bits	yes
Number of Returns (Given Pulse)	4 bits (bits 4-7)		4 bits	yes
Classification Flags	4 bits (bits 0-3)	15	4 bits	no
Scanner Channel	2 bits (bits 4-5)		2 bits	yes
Scan Direction Flag	1 bit (bit 6)		1 bit	yes
Edge of Flight Line	1 bit (bit 7)		1 bit	yes
Classification	uint8_t	16	1 byte	yes
User Data	uint8_t	17	1 byte	no
Scan Angle	int16_t	18	2 bytes	yes
Point Source ID	uint16_t	20	2 bytes	yes
GPS Time	double	22	8 bytes	yes
Red	uint16_t	30	2 bytes	yes
Green	uint16_t	32	2 bytes	yes
Blue	uint16_t	34	2 bytes	yes
Minimum PDRF Size		*36 bytes*		

2.6.9 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	Byte Offset	Size	Required
X	int32_t	0	4 bytes	yes
Y	int32_t	4	4 bytes	yes
Z	int32_t	8	4 bytes	yes
Intensity	uint16_t	12	2 bytes	no
Return Number	4 bits (bits 0-3)	14	4 bits	yes
Number of Returns (Given Pulse)	4 bits (bits 4-7)		4 bits	yes
Classification Flags	4 bits (bits 0-3)	15	4 bits	no
Scanner Channel	2 bits (bits 4-5)		2 bits	yes
Scan Direction Flag	1 bit (bit 6)		1 bit	yes
Edge of Flight Line	1 bit (bit 7)		1 bit	yes
Classification	uint8_t	16	1 byte	yes
User Data	uint8_t	17	1 byte	no
Scan Angle	int16_t	18	2 bytes	yes
Point Source ID	uint16_t	20	2 bytes	yes
GPS Time	double	22	8 bytes	yes
Red	uint16_t	30	2 bytes	yes
Green	uint16_t	32	2 bytes	yes
Blue	uint16_t	34	2 bytes	yes
NIR	uint16_t	36	2 bytes	yes
Minimum PDRF Size		*38 bytes*		

NIR

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

Note: 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.

2.6.10 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**	**Byte Offset**	**Size**	**Re-quired**
X	int32_t	0	4 bytes	yes
Y	int32_t	4	4 bytes	yes
Z	int32_t	8	4 bytes	yes
Intensity	uint16_t	12	2 bytes	no
Return Number	4 bits (bits 0-3)	14	4 bits	yes
Number of Returns (Given Pulse)	4 bits (bits 4-7)		4 bits	yes
Classification Flags	4 bits (bits 0-3)	15	4 bits	no
Scanner Channel	2 bits (bits 4-5)		2 bits	yes
Scan Direction Flag	1 bit (bit 6)		1 bit	yes
Edge of Flight Line	1 bit (bit 7)		1 bit	yes
Classification	uint8_t	16	1 byte	yes
User Data	uint8_t	17	1 byte	no
Scan Angle	int16_t	18	2 bytes	yes
Point Source ID	uint16_t	20	2 bytes	yes
GPS Time	double	22	8 bytes	yes
Wave Packet Descriptor Index	uint8_t	30	1 byte	yes
Byte Offset to Waveform Data	uint64_t	31	8 bytes	yes
Waveform Packet Size in Bytes	uint32_t	39	4 bytes	yes
Return Point Waveform Location	float	43	4 bytes	yes
Parametric dx	float	47	4 bytes	yes
Parametric dy	float	51	4 bytes	yes
Parametric dz	float	55	4 bytes	yes
Minimum PDRF Size		*59 bytes*		

2.6.11 Point Data Record Format 10

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

Table 21: Point Data Record Format 10

Item	**Format**	**Byte Offset**	**Size**	**Re-quired**
X	int32_t	0	4 bytes	yes
Y	int32_t	4	4 bytes	yes
Z	int32_t	8	4 bytes	yes
Intensity	uint16_t	12	2 bytes	no
Return Number	4 bits (bits 0-3)	14	4 bits	yes
Number of Returns (Given Pulse)	4 bits (bits 4-7)		4 bits	yes
Classification Flags	4 bits (bits 0-3)	15	4 bits	no
Scanner Channel	2 bits (bits 4-5)		2 bits	yes
Scan Direction Flag	1 bit (bit 6)		1 bit	yes
Edge of Flight Line	1 bit (bit 7)		1 bit	yes
Classification	uint8_t	16	1 byte	yes
User Data	uint8_t	17	1 byte	no
Scan Angle	int16_t	18	2 bytes	yes
Point Source ID	uint16_t	20	2 bytes	yes
GPS Time	double	22	8 bytes	yes
Red	uint16_t	30	2 bytes	yes
Green	uint16_t	32	2 bytes	yes
Blue	uint16_t	34	2 bytes	yes
NIR	uint16_t	36	2 bytes	yes
Wave Packet Descriptor Index	uint8_t	38	1 byte	yes
Byte Offset to Waveform Data	uint64_t	39	8 bytes	yes
Waveform Packet Size in Bytes	uint32_t	47	4 bytes	yes
Return Point Waveform Location	float	51	4 bytes	yes
Parametric dx	float	55	4 bytes	yes
Parametric dy	float	59	4 bytes	yes
Parametric dz	float	63	4 bytes	yes
Minimum PDRF Size		*67 bytes*		

2.7 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**	**Byte Offset**	**Size**	**Required**
Reserved	uint16_t	0	2 bytes	
User ID	char[16]	2	16 bytes	yes
Record ID	uint16_t	18	2 bytes	yes
Record Length After Header	uint64_t	20	8 bytes	yes
Description	char[32]	28	32 bytes	

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

2.7.1 Legacy Compatibility for EVLRs

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*" VLR (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.

3 Coordinate Reference System VLRs (Required)

LAS 1.4 defines Variable Length Records (VLRs) and Extended Variable Length Records (EVLRs). Coordinate Reference System (CRS) VLRs are defined in this section, while other VLRs and EVLRs are defined in the following two sections.

3.1 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 Known Text (WKT) as indicated by the WKT *Global Encoding* bit. Point Record Formats 0-5 can use either GeoTIFF or WKT, but not both simultaneously. Point Data Record Formats 6-10 must use WKT.

3.2 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 Service Spec"[34]). 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).

3.2.1 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 Service 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.

[34] https://www.opengeospatial.org/standards/ct

3.2.2 OGC Coordinate System WKT Record

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 Service Spec, with the following notes:

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

3.3 Georeferencing Information Using GeoTIFF

The GeoTIFF specification is defined by http://geotiff.osgeo.org/.

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.

3.3.1 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 an array of uint16_t values. Programmatically, the data can be structured as follows:

```
struct sGeoKeys {
    uint16_t wKeyDirectoryVersion;
    uint16_t wKeyRevision;
    uint16_t wMinorRevision;
    uint16_t wNumberOfKeys;

    struct sKeyEntry {
        uint16_t wKeyID;
        uint16_t wTIFFTagLocation;
        uint16_t wCount;
        uint16_t wValue_Offset;
    } pKey[1];
};
```

…where…

```
wKeyDirectoryVersion = 1; // Always
wKeyRevision = 1;         // Always
wMinorRevision = 0;       // Always
wNumberOfKeys         // Number of sets of 4 uint16_t integers to follow
```

3.3.2 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.

3.3.3 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.

4 Other Specification Defined VLRs (Optional)

4.1 Classification Lookup

User ID	LASF_Spec
Record ID	0
Record Length after Header	256 records * 16 bytes per struct

```
struct CLASSIFICATION {
    uint8_t ClassNumber;
    char Description[15];
}; //total of 16 bytes
```

4.2 Text Area Description

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.

4.3 Extra Bytes

User ID	LASF_Spec
Record ID	4
Record Length after Header	n descriptors * 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 VLR is only required 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, if 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 extra bytes descriptor is defined as follows:

```
struct EXTRA_BYTES {
    uint8_t reserved[2];     // 2 bytes
    uint8_t data_type;       // 1 byte
    uint8_t options;         // 1 byte
    char    name[32];        // 32 bytes
    uint8_t unused[4];       // 4 bytes
    anytype no_data;         // 8 bytes
    uint8_t deprecated1[16]; // 16 bytes
    anytype min;             // 8 bytes
    uint8_t deprecated2[16]; // 16 bytes
    anytype max;             // 8 bytes
    uint8_t deprecated3[16]; // 16 bytes
    double  scale;           // 8 bytes
    uint8_t deprecated4[16]; // 16 bytes
    double  offset;          // 8 bytes
    uint8_t deprecated5[16]; // 16 bytes
    char    description[32]; // 32 bytes
};                                            // total of 192 bytes
```

The 4 "extra bytes" could, for example, be of data_type 9 - a 4-byte 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. In another example, four EXTRA_BYTES structs in the VLR payload could describe 14 "extra bytes" in each point record:

1) "laser pulse direction [0]" - data_type = 9 (float)
2) "laser pulse direction [1]" - data_type = 9 (float)
3) "laser pulse direction [2]" - data_type = 9 (float)
4) "pulse width" - data_type = 3 (uint16_t)

In this example, an array of three individual floats collectively specify a "laser pulse direction" for that point, and one uint16_t integer specifies a "pulse width" for that point.

The "extra bytes" are made accessible via a unique name. The "name" field distinguishes the additional point attributes that 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.

Multiple sequential "extra byte" records can compose an array of associated values. It is recommended that each member's name be consistent with other members, only varying by an index number wrapped in square brackets, as in the above example. Zero-indexed arrays are encouraged. Previous revisions of the LAS 1.4 specification utilized data_types 11-30 to define standard two- and three-member arrays, but this feature was never widely implemented and was deprecated in R14[35] to simplify implementation.

[35] https://github.com/ASPRSorg/LAS/issues/1

Any unused characters in the "name" or "description" fields must be set to zero.

Table 23: Values for `data_type` Field

Value	**Meaning**	**Size on Disk**
0	undocumented extra bytes	specify value in `options` field
1	uint8_t	1 byte
2	int8_t	1 byte
3	uint16_t	2 bytes
4	int16_t	2 bytes
5	uint32_t	4 bytes
6	int32_t	4 bytes
7	uint64_t	8 bytes
8	int64_t	8 bytes
9	float	4 bytes
10	double	8 bytes
11-30	*Deprecated*	deprecated
31-255	*Reserved*	not assigned

Table 24: Encoding of `options` Bit Field

Bits	**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).

The bit mask in the "options" field specifies whether the min and max range of the value has been set (i.e., is 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. Any unused "no_data", "min", "max", "scale", or "offset" fields must be set to zero.

If the selected data_type is less than 8 bytes, the no_data, min, and max fields should be upcast into 8-byte storage. For any float these 8 bytes would be upcast to a double, for any uint8_t, uint16_t, or uint32_t they would be upcast to a uint64_t and for any int8_t, int16_t, or int32_t, they would be upcast to a int64_t.

If used, the min and max fields reflect the actual minimum and maximum values of the attribute in the LAS file, in its raw form, without any scale or offset values applied.

The "reserved" field, the "unused" field, and the "deprecated" fields must be set to zero and may be used in a future revision.

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 designated 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.

4.4 Superseded

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.

4.5 Waveform Packet Descriptor

User ID	LASF_Spec
Record ID	n: where n > 99 and n < 355

Warning: This VLR is REQUIRED when using Point Data Record Formats 4, 5, 9, or 10.

These records contain information that describes the configuration of the waveform packets. Since system configuration may vary throughout a dataset, the LAS file supports up to 255 Waveform Packet Descriptors.

Table 25: Waveform Packet Descriptor User Defined Record

Item	**Format**	**Size**	**Required**
Bits per Sample	uint8_t	1 byte	yes
Waveform Compression Type	uint8_t	1 byte	yes
Number of Samples	uint32_t	4 bytes	yes
Temporal Sample Spacing	uint32_t	4 bytes	yes
Digitizer Gain	double	8 bytes	yes
Digitizer Offset	double	8 bytes	yes

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 and Offset

The digitizer gain and offset are used to convert the raw digitized value to an absolute digitizer voltage using the formula:

$$VOLTS = OFFSET + GAIN * Raw\_Waveform\_Amplitude$$

Note: Users seeking further clarity regarding LAS waveform encoding are encouraged to learn more on the official LAS wiki: https://github.com/ASPRSorg/LAS/wiki

---

5 Defined Extended Variable Length Records (EVLRs)

5.1 Waveform Data Packets

Warning: This EVLR is REQUIRED internally or externally when using Point Data Record Formats 4, 5, 9, or 10.

User ID	LASF_Spec
Record ID	65535

The packet of Raw Waveform Amplitude values for all records immediately follow this VLR 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.

6 LAS Domain Profiles

A derivative of the base LAS specification that adds (but does not remove or alter existing) point classes and attributes to meet the application-specific needs of a particular subset of the broad point cloud community (e.g., the coastal/bathymetric lidar community, or the powerline mapping community). So as to not alter the LAS base specification, new classes can only be added to Point Data Record Formats 6-10, and classification values cannot start below 39. New attributes must be incorporated using *Extra Bytes* VLRs. It is strongly recommended that the development of Domain Profiles be coordinated to avoid unnecessary overlap or conflicts (e.g., conflicting class numbers) between profiles.

6.1 LAS Domain Profile Description

The specification for a particular domain profile. The Description must contain:

1) an overview of the purpose and intended use;
2) table of new point classifications (PDRF 6-10); and
3) table of new attributes to be stored using Extra Bytes VLRs (must contain fields for units, data type, name, no data, scale, and description).

LAS Domain Profile Descriptions reviewed and approved by the LAS Working Group will be posted on the ASPRS website. A template Domain Profile Description is available on the ASPRS website[36].

[36] https://www.asprs.org/committee-general/laser-las-file-format-exchange-activities.html