LAS Specification 1.5 - R00

Release Information:

Version Approved – August 2025
Revision date – 27 August 2025

PDF build date – 21 May 2026
GitHub commit – bc3d73b183612c3cd9e040b76d4cff6258aade85
GitHub repo – https://github.com/ASPRSorg/LAS

Published by:

The American Society for Photogrammetry & Remote Sensing
8550 United Plaza Blvd. Suite 1001
Baton Rouge, LA 70809
Voice: 225-408-4747
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 fifth version of the LAS file specification since its initial version 1.0 release.

1.1.1 LAS 1.5 Revision History

Summary of LAS 1.5 revisions (topic links included when applicable):

- R00 - Approved Version (August 2025).

1.1.2 Comparison of LAS 1.5 to Previous Versions

The additions of LAS 1.5 include (topic links included when applicable):

- Backward compatibility with 64-bit LAS 1.4 files.
- Header compatibility with LAS 1.1 - 1.4 files.
- Removed support for legacy Point Data Record Formats 0-5. (I-128[3])
- Added Max/Min GPS Time fields to header block. (I-118[4])
- Added Offset GPS Time definition. (I-6[5])
- CRS support expanded to all published WKT versions, removing support for GeoTIFF CRS encoding. (I-95[6], I-104[7], I-129[8])
- Clarify requirement that the Extra Bytes VLR must be unique. (I-150[9])
- Deprecated Classification Lookup VLR. (I-82[10])

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:

[3] https://github.com/ASPRSorg/LAS/issues/128
[4] https://github.com/ASPRSorg/LAS/issues/118
[5] https://github.com/ASPRSorg/LAS/issues/6
[6] https://github.com/ASPRSorg/LAS/issues/95
[7] https://github.com/ASPRSorg/LAS/issues/104
[8] https://github.com/ASPRSorg/LAS/issues/129
[9] https://github.com/ASPRSorg/LAS/issues/150
[10] https://github.com/ASPRSorg/LAS/issues/82

- Online at https://github.com/ASPRSorg/LAS
- By phone at 225-408-4747
- 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[11]) since 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://www.ogc.org/publications/standard/las/.

Future recognition and activity on OGC referencing activities of LAS can be followed at https://www.ogc.org/publications/.

1.4 Official LAS Wiki

The official LAS wiki hosts supplemental guidance pages, links to external resources, public registries, and more LAS-related resources. The wiki can be found at https://github.com/ASPRSorg/LAS/wiki.

[11] https://www.ogc.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, meta-data, 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 1.5 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 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 moved the file specification from a 32 bit file structure to a 64 bit file structure. To maintain the ability to place a LAS 1.1 through LAS 1.3 payload in a LAS 1.4 or LAS 1.5 file header, optional support for the 32 bit field variants is retained alongside the required 64 bit fields. These duplicate 32 bit fields are named "Legacy xxx" where "xxx" denotes the meaning of the field.

A LAS 1.5 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 1.5 field, the LAS reader should use the legacy value to maintain the same behavior as prior versions of LAS. 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 file writer wishing to maintain backward compatibility must use only VLRs. See the *Legacy Compatibility for EVLRs* section for more information.

2.2 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.

2.3 Public Header Block

Table 2: 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
Number of Point Records	uint64_t	247	8 bytes	yes
Number of Points by Return	uint64_t[15]	255	120 bytes	yes
Max GPS Time	double	375	8 bytes	

continues on next page

Table 2 – continued from previous page

Item	Format	Byte Offset	Size	Required
Min GPS Time	double	383	8 bytes	
Time Offset	uint16_t	391	2 bytes	yes
LAS 1.5 Header Size		*393 bytes*		

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, defined as follows:

Table 3: Global Encoding – Bit Field Encoding

Bits	Field Name	Description
0	GPS Time Type	Used in combination with Time Offset Flag to indicate the meaning of *GPS Time* in the point records. If neither bit is 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	Well Known Text (WKT)	If set, the Coordinate Reference System (CRS) is formatted as OGC WKT. This bit must be set in LAS 1.5; GeoTIFF CRS definitions from prior versions of LAS are no longer supported in LAS 1.5. The CRS definition will be located in the WKT VLR(s) or EVLR(s) as defined in the *Coordinate Reference System* section.
5	Reserved	Must be set to zero.
6	Time Offset Flag	Used in combination with GPS Time Type to indicate the meaning of *GPS Time* in the point records. If both bits are set, the GPS Time is defined as Offset GPS Time. The Offset GPS Time is standard GPS Time (satellite GPS Time) minus 1 x 10^6 x Time Offset. The *Time Offset* is specified in the LAS header. The offset moves time to near zero to improve floating-point resolution for a given time period. The origin of standard GPS Time is defined as midnight of the morning of January 6, 1980. The GPS Time Type flag must be set if the Time Offset Flag is set.
7:15	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.

Note

Example implementations of representing the Project ID fields as a GUID can be found on the *Official 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 1.5 would contain 1 in the major field and 5 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 4: 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 1.5 this size is 393 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 format (PDRF) value indicates the type of point data records contained in the file. LAS 1.5 defines types 6 through 10. These types are defined in the *Point Data Records* section of this specification. PDRFs 0-5 as defined in previous versions of LAS are no longer valid for LAS 1.5.

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 PDRF and hence the same length. If the specified size is larger than implied by the PDRF type (e.g., 32 bytes instead of 30 bytes for PDRF 6) 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 1.5 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 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.

> **Note**
>
> Any changes to the number of points or their attributes must also be reflected in the Public Header Block.

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.

Max and Min GPS Time

The max and min GPS Time fields reflect the highest and lowest non-zero GPS Time values stored with the point records in this LAS file. The GPS Time in this field will be in the same encoding described in the file's *Global Encoding* field.

These values must be set to zero if there are no point records in the file or if all point records within the file have GPS Time values set to zero, such as for certain *Aggregate Model Systems*.

Time Offset

The Time Offset field can be used to optimize *GPS Time* precision for a desired time period. Offset GPS Time for a point record is equal to standard GPS Time, minus 10^6 * Time Offset.

For example, Adjusted Standard GPS Time uses a total offset of 1 billion (10^9) seconds, which would be specified by a Time Offset of 1000. This achieves 100 nanosecond precision for the years 2007-2015. The *Official LAS Wiki* provides recommended values and year ranges that users are encouraged to use to maximize compatibility between datasets.

This value must be set to zero if there are no point records in the file, if the file is derived from a source not utilizing GPS Time (such as for certain *Aggregate Model Systems*), or if the Time Offset Flag is unset.

It is invalid for the GPS Time Type bit to be unset if the Offset Time Flag bit is set.

2.4 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 5: 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.

2.5 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) 0-5 as defined in previous versions of LAS are no longer valid for LAS 1.5. PDRFs 6-10 have improved several aspects of the core information in the point data records, particularly support for 256 classes, the definition of a specific "Overlap" bit, support for multiple sensor channels, etc.

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."

2.5.1 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.

Table 6: Point Data Record Format 6

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)	14:0	4 bits	yes
Number of Returns (Given Pulse)	4 bits (bits 4-7)	14:4	4 bits	yes
Classification Flags	4 bits (bits 0-3)	15:0	4 bits	no
Scanner Channel	2 bits (bits 4-5)	15:4	2 bits	yes
Scan Direction Flag	1 bit (bit 6)	15:6	1 bit	yes
Edge of Flight Line	1 bit (bit 7)	15:7	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 Length[1]		*30 bytes*		

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.

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,

[1] Recall that the Point Data Record Length 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 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

The following two fields (Return Number and Number of Returns) are bit fields, encoded into one 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 1, the second a Return Number of 2, and so on up to 15 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 1, 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 2 (Return Number) within a total number of up to 15 returns.

For systems unable to record multiple returns, the Number of Returns should be set to 1, unless it is synthetically derived and the Synthetic Return Number *Global Encoding* bit is set.

Note

The following four fields (Classification Flags, Scanner Channel, Scan Direction Flag, and Edge of Flight Line) are bit fields, encoded into one byte.

Classification Flags

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

Table 7: 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

This field represents the "class" attributes of a point. Classifications 0-63 are reserved for ASPRS-standard definitions, as defined in this specification and approved *LAS Domain Profiles*. Classifications 64 and above may be assigned meaning at the user's discretion.

Classification must adhere to the following standard:

Table 8: ASPRS Standard Point Classes (Point Data Record Formats 6-10)

Value	Meaning	Notes
0	Created, Never Classified	See note[2]
1	Unclassified	See note[2]
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	

User Data

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

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 coordinate system of the data). The maximum value

[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.

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.

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.

GPS Time

The GPS Time is the double floating point time tag value at which the point was observed. GPS Time encoding is specified by the *Global Encoding* bit field. The time value encoding is defined as GPS Week Time if the GPS Time Type bit is unset, Adjusted Standard GPS Time if the GPS Time Type bit is set, and Offset GPS Time if the Time Offset Flag bit is set.

Table 9: GPS Time Encoding

GPS Time Type bit	Time Offset Flag bit	Timestamp Encoding
Unset	Unset	GPS Week Time
Set	Unset	Adjusted Standard GPS Time
Unset	Set	*INVALID*
Set	Set	Offset GPS Time

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.

2.5.2 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 10: 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)	14:0	4 bits	yes
Number of Returns (Given Pulse)	4 bits (bits 4-7)	14:4	4 bits	yes
Classification Flags	4 bits (bits 0-3)	15:0	4 bits	no
Scanner Channel	2 bits (bits 4-5)	15:4	2 bits	yes
Scan Direction Flag	1 bit (bit 6)	15:6	1 bit	yes
Edge of Flight Line	1 bit (bit 7)	15:7	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 Length		*36 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.

2.5.3 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 11: 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)	14:0	4 bits	yes
Number of Returns (Given Pulse)	4 bits (bits 4-7)	14:4	4 bits	yes
Classification Flags	4 bits (bits 0-3)	15:0	4 bits	no
Scanner Channel	2 bits (bits 4-5)	15:4	2 bits	yes
Scan Direction Flag	1 bit (bit 6)	15:6	1 bit	yes
Edge of Flight Line	1 bit (bit 7)	15:7	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 Length		*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.

2.5.4 Point Data Record Format 9

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

Table 12: Point Data Record Format 9

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)	14:0	4 bits	yes
Number of Returns (Given Pulse)	4 bits (bits 4-7)	14:4	4 bits	yes
Classification Flags	4 bits (bits 0-3)	15:0	4 bits	no
Scanner Channel	2 bits (bits 4-5)	15:4	2 bits	yes
Scan Direction Flag	1 bit (bit 6)	15:6	1 bit	yes
Edge of Flight Line	1 bit (bit 7)	15:7	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 Length		*59 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:

$$X = X_0 + t * dx \tag{4}$$
$$Y = Y_0 + t * dy \tag{5}$$
$$Z = Z_0 + t * dz \tag{6}$$

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:

$$X_0 = X_P + L * dx \tag{7}$$
$$Y_0 = Y_P + L * dy \tag{8}$$
$$Z_0 = Z_P + L * dz \tag{9}$$

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*.

2.5.5 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 13: Point Data Record Format 10

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)	14:0	4 bits	yes
Number of Returns (Given Pulse)	4 bits (bits 4-7)	14:4	4 bits	yes
Classification Flags	4 bits (bits 0-3)	15:0	4 bits	no
Scanner Channel	2 bits (bits 4-5)	15:4	2 bits	yes
Scan Direction Flag	1 bit (bit 6)	15:6	1 bit	yes
Edge of Flight Line	1 bit (bit 7)	15:7	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 Length		*67 bytes*		

2.6 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 14: 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.

2.6.1 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

Coordinate Reference System (CRS) information is required for a LAS 1.5 file and must be placed in *Variable Length Records (VLRs)* or *Extended Variable Length Records (EVLRs)*. LAS 1.5 CRS information shall be represented as OGC Well Known Text[12] (WKT), as indicated by the *Global Encoding* bit.

GeoTIFF CRS representations from prior versions of LAS are not supported in LAS 1.5.

It is considered a file error to have more than one WKT VLR or EVLR in the file. A writer can efficiently revise a CRS definition without rewriting the entire LAS file by "superseding" the existing CRS VLR or EVLR and appending a new CRS definition in an EVLR. Superseding is performed by changing the Record ID of the VLR or EVLR to match the *Superseded* VLR definition.

3.1 OGC Coordinate System WKT Record (Required)

The WKT record is **required** and can be specified as either a VLR or an EVLR. If an EVLR is to override an already existing CRS VLR, writers should also write a *Superseded VLR* to label it.

User ID	LASF_Projection
Record ID	2112

This record must contain the textual data representing a WKT as defined by any published OGC Well Known Text standard with the following considerations:

- 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 with no localization of the numeric strings within the WKT being performed.

Note

Any published version of an OGC WKT standard is a valid definition for LAS 1.5. OGC has published multiple versions of WKT, including the following noteworthy versions:

- 18-010r11[13] - Commonly called "WKT2" or "WKTv2" and is the most current version of WKT as of this publication.
- 18-010r7[14] - This variant of "WKT2" added support for the definition of `EPOCH` for defining the date against which coordinates in the LAS file are referenced in a dynamic coordinate reference system.
- 12-063r5[15] - Commonly called "WKT1" and the formally ratified variant of WKT that was used for LAS 1.4.

If OGC were to publish a new WKT standard, it would also be a valid definition for LAS 1.5.

[12] https://www.ogc.org/standard/wkt-crs/
[13] https://docs.ogc.org/is/18-010r11/18-010r11.pdf
[14] https://docs.ogc.org/is/18-010r11/18-010r11.pdf
[15] https://portal.opengeospatial.org/files/12-063r5

3.2 OGC Math Transform WKT Record (Optional)

Writers may include an optional OGC Math Transform WKT Record to provide a supplemental parameterized math transform definition.

User ID	LASF_Projection
Record ID	2111

This record must contain a Math Transform WKT Record consistent with the same WKT specification being used by the *Coordinate System WKT Record*, with the following considerations:

- 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.

4 Optional VLR Definitions

Definitions for official *Variable Length Records (VLRs)* and *Extended Variable Length Records (EVLRs)* are provided in this section as a means to extend the LAS file format. Additional VLR and EVLR definitions contributed by the LAS Working Group and the LAS community can be found on the *Official LAS Wiki*.

4.1 Classification Lookup (Deprecated)

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
```

The legacy Classification Lookup VLR (Record ID 0) has been a feature of the LAS specification since LAS 1.0. This VLR is deprecated in LAS 1.5 due to the limitation of 15 characters being insufficient for a detailed description. An updated version of the VLR may be found on the *Official LAS Wiki*.

4.2 Text Area Description

User ID	LASF_Spec
Record ID	3

This VLR 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 was added to LAS 1.4 to formalize a process that had been used in prior versions of LAS, and is a core feature of the LAS 1.5 specification. 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 length is set to a larger value than required by the point type. For example, if a LAS file that contains point type 6 has a point record length of 34 instead of 30, 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.

Only one Extra Bytes VLR may exist in a given LAS file. If an Extra Bytes VLR already exists in a LAS file, then new Extra Byte definitions must be added to the existing Extra Bytes VLR.

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.

Note

Previous versions of the LAS specification utilized data_type values 11-30 to define two- and three-member arrays, but this feature was never widely implemented and was deprecated in LAS 1.4 R14[16] to simplify implementation.

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

Table 15: 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 16: 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

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

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 or EVLR when rewriting the file (to remove the undesired VLR or 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 17: 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*.

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 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[17].

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