ROBLOX FILE FORMAT

Gregory Comer

TEAM HAVEMEAT

1 CONTENTS

2 DISCLAIMER

The contents of this document are not in any way sponsored by, affiliated with, or created by Roblox Corporation and solely represent my own findings. All of my findings, as documented here, came from my own investigation using a hex editor, Roblox Studio, and tools of my own creation. At no time was any Roblox software disassembled. All findings came from inspection of saves generated by Roblox Studio.

3 INTRODUCTORY NOTES

The contents of this document represent my best efforts at documenting the File Format used by Roblox for both places and models. The format used is subject to changes outside my control. I intend to keep this document up to date, but no guarantee is offered. My primary purpose in writing this document is to facilitate the development of third-party tools that interact with saves generated by Roblox.

There are a couple of things in the file format that, admittedly, I do not know the purpose of. However, in each of these cases, the portion that I do not understand remains the same in each file. In other words, it appears that these portions of the file, one can safely write the same value as observed in Roblox Studio-generated saves.

One example of this is near the beginning of the file. It seems likely that version information is present, but I do not currently know. I'll need to observe the file format over time. As for now, the documentation provided in this file is functional. In the few areas where it is not complete, it is documented well enough that the reader should be able to both read and write all important information contained within the file.

4 PRELIMINARY CONCEPTS

4.1 ENDIANNESS

Both big- and little-endian conventions are used by Roblox. As a general rule, property data tends to be big-endian, which file internals, such as string length, tend to be little-endian. The endianness of specific fields are noted individually.

4.2 COMPRESSION

Roblox uses the LZ4 compression algorithm internally to reduce the size of save files. The basics of the algorithm will be presented here. Additional information is available online.

4.2.1 Compression Header

All regions of compressed data in saves are prefixed by a 0xC (12) byte header, consisting of three 32-bit integers (little-endian). The first value indicates the length of the compressed data. The second indicates the length of the decompressed data. These values are important for decompression. The third value appears to always be zero, at least in every case studied here. At the moment, it appears that it can safely be set to zero in every case.

Figure 4-1

4.2.2 Compression Algorithm

The following is a quick overview of the LZ4 compression algorithm. Details and a reference implementation can be found online.

Compressed data is divided into blocks. Each block can contain both *literal data* and *matches*. The final block in a sequence of compressed data does not contain match data. Blocks begin with a *token byte*, which, when divided into higher and lower order segments, indicates the length of literal data (higher order 4 bits) and the length of the match portion (lower order 4 bits).

If the literal length is equal to or exceeds 0xF (15) bytes, the higher order 4 bits of the token byte are set to 0xF (15). The remaining length is stored following the token byte. Each byte of the literal length is added to the literal length. The literal length ends after the first byte not equal to 0xFF (255). Following the literal length comes the literal data. The length of the literal data is equal to the literal length. Literal data is copied directly to the output stream. It is not compressed.

Directly after the literal data comes the *offset* value, which is stored as a 16-bit little-endian value. If the match length value stored in the token byte is equal to or exceeds 0xF (15), the remaining match length is stored in the same manner as literal length following the offset value. The true literal length is 4 more than the value indicated by the literal length. This is because the minimum match length is 4 bytes.

To translate a block into decompressed data, first copy the literal data to the output stream. If match data is present (which it will be, unless this is the final block), then copy *match length* bytes from the output stream, starting *offset* bytes from the end. If the match length exceeds the offset value, then match will overlap. This is allowed.

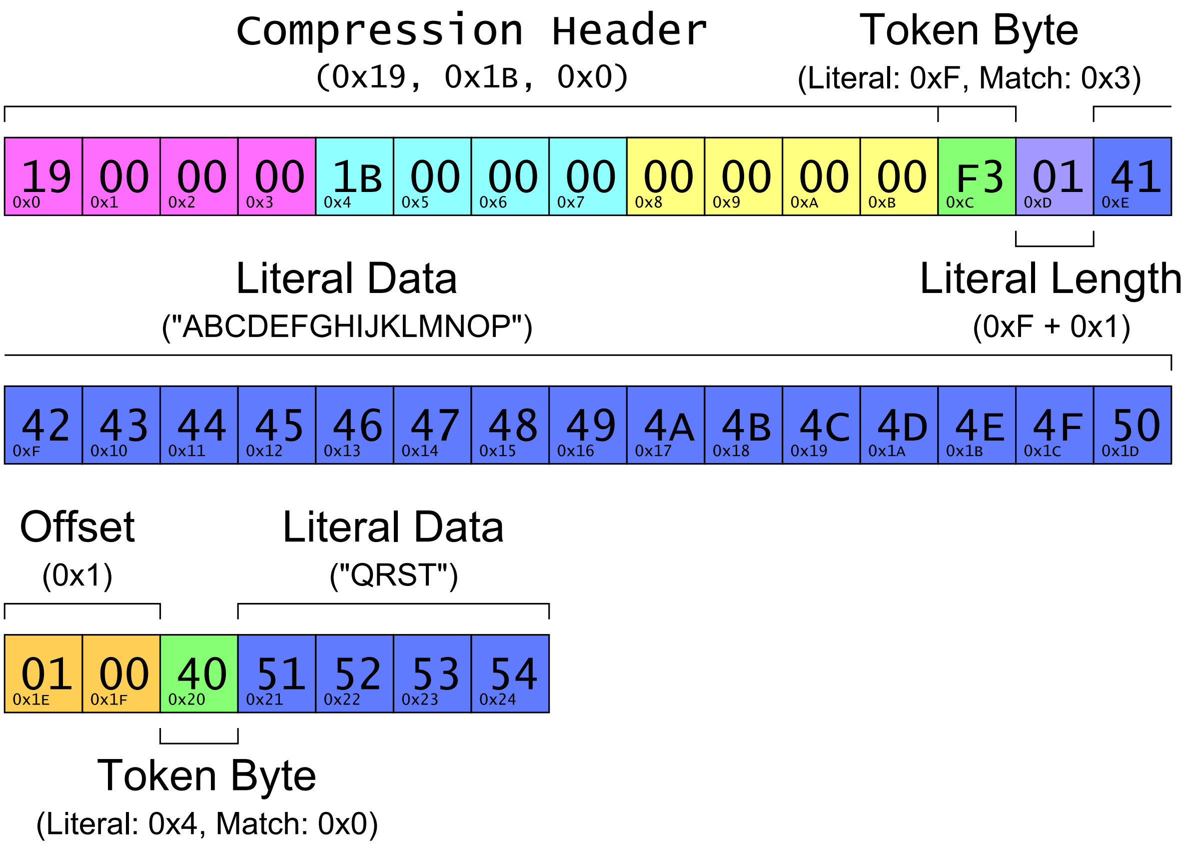


Figure 4-2

In the above example, there are two blocks. The first consists of 0x10 (16) bytes of literal data and 0x7 (7) bytes of match data. The second consists of 0x4 (4) bytes of literal data and no match data, as it is the last block.

The decompressed stream is shown below. The first 0x10 (16) bytes are from Block #1's literal data. Next, starting from 0x1 (1) byte before the end of the output stream, 0x7 (7) bytes are copied to the end of the output stream. In this case, the match overlaps, causing 0x50 ('P') to be repeated 0x7 (7) times. Finally, 0x4 (4) bytes of literal data are copied from block #2.

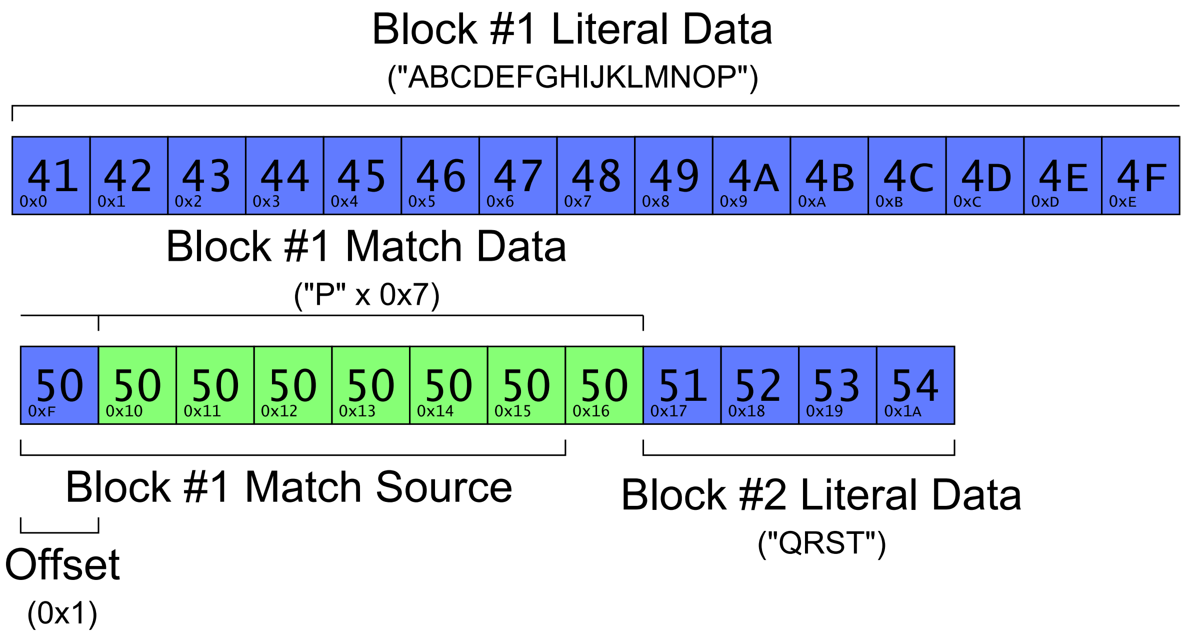

Block #1 Literal Data
("ABCDEFGHIJKLMNOP")
41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
0x0 0x1 0x2 0x3 0x4 0x5 0x6 0x7 0x8 0x9 0xA 0xB 0xC 0xD 0xE
Block #1 Match Data
("P" x 0x7)
50 50 50 50 50 50 50 50 51 52 53 54
0xF 0x10 0x11 0x12 0x13 0x14 0x15 0x16 0x17 0x18 0x19 0x1A
Block #1 Match Source
Block #2 Literal Data
("QRST")
Offset
(0x1)

Figure 4-3

4.2.2.1 Decompression Pseudocode

The following pseudocode describes the decompression procedure:

```
//Read Compression Header
CompressedLen = ReadInt32()
DecompressedLen = ReadInt32()
HeaderReserved = ReadInt32()
While (TotalDecompressed < DecompressedLen)
{
      TokenByte = ReadByte()
      LiteralLen = TokenByte >> 4
      MatchLen = TokenByte & 0xF
      If (LiteralLen == 0xF)
      {
            Do
            {
                  LastByte = ReadByte()
                  LiteralLen += LastByte
            } While (LastByte == 0xF)
      }

      Copy LiteralLen bytes from compressed stream to output

      If (TotalDecompressed + LiteralLen < DecompressedLen)
      {
            Offset = ReadInt16()

            If (MatchLen == 0xF)
            {
                  Do
                  {
                        LastByte = ReadByte()
                        MatchLen += LastByte
                  } While (LastByte == 0xF)
            }

            Copy MatchLen bytes from decompressed stream (starting at Offset
                  bytes from end) to end of decompressed stream
      }
}
```

4.3 DATA TRANSFORMATIONS

Roblox utilizes several types of transformations in order to increase compressibility.

4.3.1 Byte Interleaving

In many cases where an array of values is stored contiguously, the bytes of each value are interleaved. Instead of storing each value consecutively, all of the first bytes are stored consecutively, then all of the second bytes, and so on.

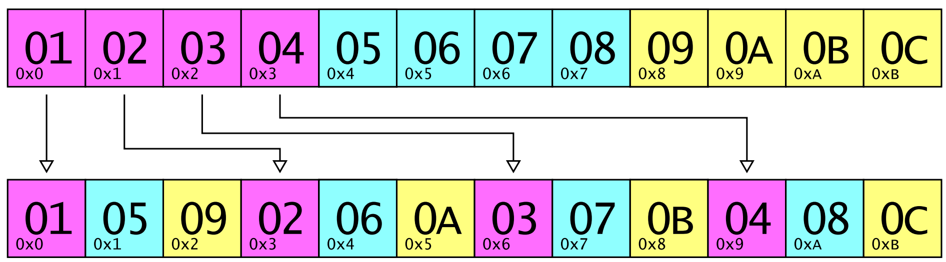


Figure 4-4

4.3.2 Integer Transformation

When integers are stored in property and referent data, they go through a transformation to increase compressibility by increasing the number of zeroes in negative values. This transformation takes place prior to byte interleaving (see 2.3.1). Positive numbers are multiplied by 2. Negative numbers are negated (making them positive), multiplied by two, and reduced by 1.

$$f(x) = \begin{cases} 2x & \text{if } x \geq 0 \\ 2|x|-1 & \text{if } x < 0 \end{cases}$$

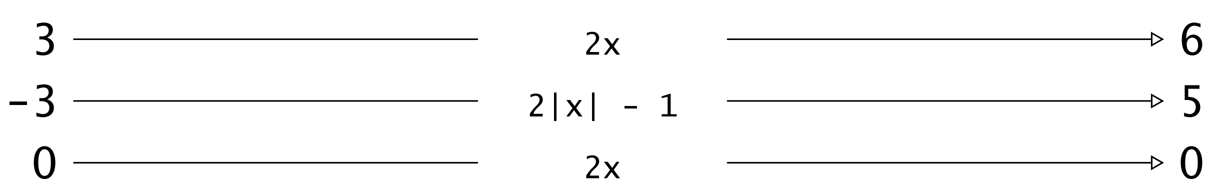


Figure 4-5

4.3.3 Float Storage

When 32-bit floating point values are stored in property data, they are stored in a custom format, presumably for compression purposes. Using IEEE 754 32-bit floating point format as a reference (the common standard for float values), the rearranged data is in the follow order: exponent (8 bits), significand (23 bits), sign (1 bit).

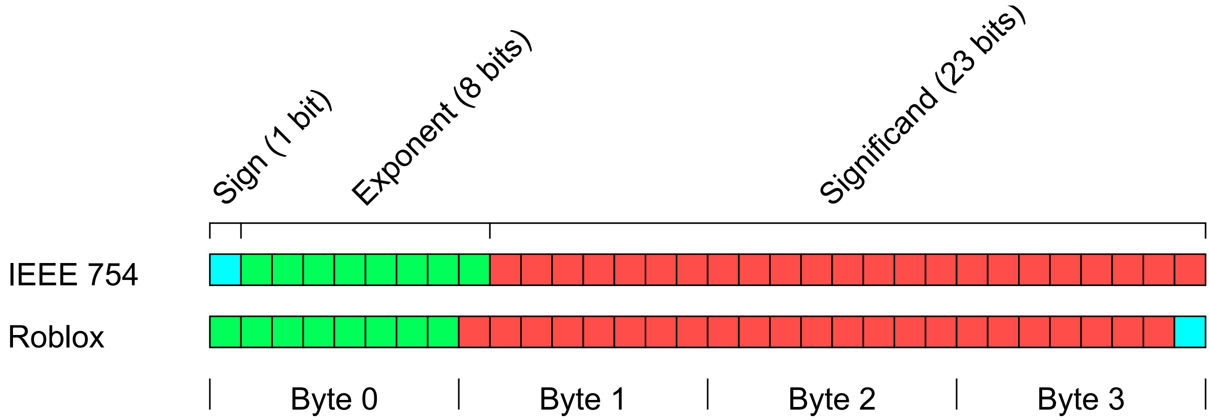


Figure 4-6

4.4 Referents

In order to facilitate references to objects, each object listed in the file is assigned a value known as a *referent.* Referents are 32-bit integers. When stored, they are subject to the integer transformation (see 2.3.2).

4.5 Strings

Roblox stores strings using ANSI Character Set 1 (ISO/IEC 8859-1), a superset of ASCII. Each character takes up only one byte. As such, string lengths represent both the length in bytes and the number of characters.

5 FILE STRUCTURE

5.1 PLACES AND MODELS

Roblox saves both places (.rbxl) and models (.rbxm) using the exact same file format. The only difference is that models cannot contain services. Roblox Studio will not load models that contain services.

5.2 OVERVIEW

Rbxl files are split into roughly three parts: *header*, *property data*, and *parent data*. The header contains the total number of objects, number of types, and referent data (see 2.4). The property data section contain data for each object's properties, such as name, size, and position. Parent data stores the parent of each object.

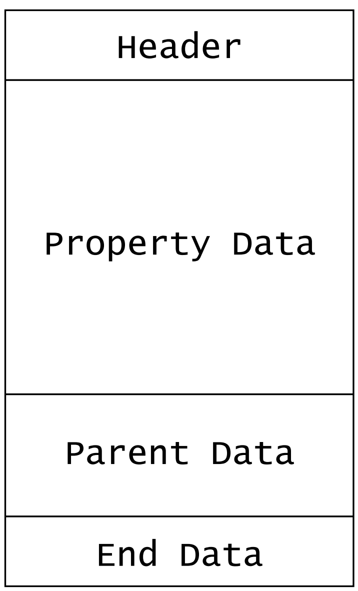


Figure 5-1

5.3 Header

The first 0x10 (16) bytes in each file appear to be constant. A portion of this appears to be a signature (indicating file type). There is also likely a version number contained somewhere. However, as nothing has changed in these values in any examined files, it is not practical at this time to determine the exact meaning of the header. For now, it can safely be set as the following 0x10 (16) bytes: `3C 72 6F 62 6C 6F 78 21 89 FF 0D 0A 1A 0A 00 00`. When interpreted using ANSI Charset – Part 1[1] (ISO/IEC 8859-1, a superset of ASCII used by Roblox), the header looks like this: `<roblox!‰ÿ......`.

Next comes the total number of unique types (32-bit integer, little-endian, no transformation). This is followed by the total number of objects (32-bit integer, little-endian, no transformation). There are 8 bytes of zeroes (0x00) after these two values. They may have some significance, but in every observed case, they have been set to 0. As such, they can, at the current time, likely be safely set as 0s.

Following this comes the type headers. Each type present in the file has its own record. Within the record, the type is assigned a *Type ID* and each object of that type is assigned a referent (see section 2.4). Each type record begins with “INST” (49 4E 53 54). Following this is a region of LZ4 compressed data. It begins with a 0xC (12) byte header, after which follows the compressed data. See section 2.2 for a description of the compression method used.

The following refers to region of data described in the previous paragraph after it has been decompressed:

The first four bytes represent the Type ID of the current record (32-bit integer, little-endian, no transformation). Next comes the length of the type name (32-bit integer, little endian, no transformation). After this is the *Type Name*. The length of the type name is given by the preceding four bytes. The type name can be interpreted as a string (see 2.5). After the type name, there is one byte, indicating the presence of additional data following the referent array. A value of 0x0 indicates no data. A value of 0x1 indicates the presence of additional data.

Next comes the number of objects of that type (32-bit integer, little endian, no transformation). Then comes referent data. Referent data is an array of big-endian, 32-bit integers, subject to both byte interleaving (see 2.3.1) and integer transformation (see 2.3.2). The referent of the first object of the current type is equal to the first value in the transformed array. The next is equal to second value in the array *added to the first*. Each value is relative to the previous. Entries in the array may be negative. Each value is equal to the corresponding entry in the transformed array, plus the previous referent value. Hence, if the transformed array were {3, 2, 1}, the referents for objects of the current type would be {3, 5, 6}.

For some types, primarily services, referent data is followed by a string of bytes (each of which equals 0x01), the length of which is equal to the number of objects of the current type. This data is present if the additional data byte following the name of the type is equal to 0x1. The meaning of this additional data is currently unknown. Besides services, the record containing “INSTANCE” appears to always exhibit this behavior.

---

[1] Non-printable characters are displayed here as dots.

Header Record (Decompressed)

4 Bytes	4 Bytes		1 Byte	4 Bytes	4×N Bytes	N Bytes
Type ID	Name Len	Name	Additional Data Present	Instance Count (N)	Referent Array	Additional Data

Figure 5-2

Decompressed Header Record:

```
13 00 00 00 04 00 00 00 50 61 72 74 00 03 00 00   ........Part....
00 00 00 00 00 00 00 00 00 00 04 02 06            .............
```

The first four bytes indicate that the Type ID is 0x13.

The next four byte indicate that the length of the Type Name is 0x04 bytes.

The next 0x04 bytes show that that the Type Name is "Part".

The next byte (0x0) indicates that there is no additional data beyond the referent array.

The next four bytes indicate that there are 0x3 instances.

The next 0x12 bytes represent a referent array (see section 5.2.16). The raw array is { 0x4, 0x2, 0x6 } (The bytes are interleaved, see section 3.3.1). These values represent relative offsets. The actual data is { 0x4, 0x8, 0xE }. These values are transformed (see section 3.3.2). The referents are { 0x2, 0x4, 0x7 }.

The header ends after the number of records read is equal to the number of types (as specified at the beginning of the header). The next four bytes will be `PROP` (`50 52 4F 50`).

Header

22 Bytes	4 Bytes	4 Bytes
3C 72 6F 62 6C 6F 78 21 89 FF 0D 0A 1A 0A 00 00	Num Types	Num Objects

Num Objects

4 Bytes	12 Bytes	
49 4E 53 54 "INST"	Compression Header	Compressed Record

4 Bytes	12 Bytes	
49 4E 53 54 "INST"	Compression Header	Compressed Record

4 Bytes	12 Bytes	
49 4E 53 54 "INST"	Compression Header	Compressed Record

. . .

4 Bytes	12 Bytes	
49 4E 53 54 "INST"	Compression Header	Compressed Record

Figure 5-3

5.4 Property Data

Similar to header records, the property data section is an array of identically structured regions, each of which represents one property of one type. Each region starts with the same four bytes: `PROP` (`50 52 5F 50`). Next comes the standard 0xC (12) byte compression header, followed by a region of compressed data. See section 2.2 for a description of the compression method used.

The following refers to the region of data described in the previous paragraph, after it has been decompressed:

The first four bytes indicate the type to which this property belongs (see 3.2) (32-bit integer, little-endian, no transformation). Next comes the length of the property name (32-bit integer, little-endian, no transformation), and then the property name. The next byte indicates the type of the property (see section 4.1). After this comes an array of property values. The exact format of the stored property values depends on the property type (see section 4.2).

Property sections, as generated by Roblox Studio saves, are always ordered first by associated Type ID, then alphabetically by property name. Roblox does not require this. It will load sections in any order.

Property Record (Decompressed)

4 Bytes	4 Bytes		1 Byte	
Type ID	Name Len	Name	Data Type	Data Array

Figure 5-4

```
Decompressed Property Record:
            13 00 00 00 0A 00 00 00 42 72 69 63 6B 43 6F 6C  ........BrickCol
            6F 72 0B 00 00 00 00 00 00 00 03 03 01 EB F2     or..........ëò
```

The first four bytes indicate that the Type ID is 0x13.

The next four byte indicate that the length of the Property Name is 0x0A bytes.

The next 0x0A bytes show that that the Property Name is "BrickColor".

The next byte (0x0B) indicates that the property's data type is BrickColor (see section 5.1).

The next 0xC bytes represent the data array (see section 5.2.16). The bytes are interleaved, see section 3.3.1. The deinterleaved array is { 0x1, 0x3EB, 0x3F2 }. These correspond[2] to { "White", "Really black", "Really blue" }.

---

[2] See Roblox wiki page on BrickColor codes (http://wiki.roblox.com/index.php?title=BrickColor_codes).

The property data section ends when the following four bytes are read following the end of a property descriptor: `PRNT` (`50 52 4E 54`).

Property Data

4 Bytes	12 Bytes	
50 52 5F 50 "PROP"	Compression Header	Compressed Property Record

4 Bytes	12 Bytes	
50 52 5F 50 "PROP"	Compression Header	Compressed Property Record

4 Bytes	12 Bytes	
50 52 5F 50 "PROP"	Compression Header	Compressed Property Record

. . .

4 Bytes	12 Bytes	
50 52 5F 50 "PROP"	Compression Header	Compressed Property Record

Figure 5-5

5.5 Parent Data

The parent data section indicates the parent object of each stored object. It begins with a standard 0xC (12) byte compression header, followed by a region of compressed data. See section 2.2 for a complete description of the compression method used.

The following refers to the region of data described in the previous paragraph, after it has been decompressed. The first byte appears to always be zero. Its purpose is unknown. Next comes the number of objects present in the parent section (32-bit integer, little-endian, no transformation). It appears to always match the object count in the header, though whether or not this is required is unknown. The remaining data is split into two regions, each *Length* * 4 bytes long. Both are arrays of 32-bit, big-endian integers, subject to both integer and byte interleaving transformations.

After transformation, both arrays represent a list of referents. The parent of the n^{th} object in the first array is the n^{th} object in the second. See sections 2.5 and 3.2 for a description of referents. A parent value of -1 indicates the object's parent is game (DataModel).

Then end of the parent data section is denoted by `END` (`45 4E 44`).

4 Bytes	4 x N Bytes	4 x N Bytes
Object Count	Ref Array	Parent Array

Figure 5-6

5.6 Ending Data

The meaning of the data following END (45 4E 44) is not currently known. It appears the same in every file studied. It consists of 0x16 (22) bytes of data: 00 00 00 00 00 09 00 00 00 00 00 00 00 3C 2F 72 6F 62 6C 6F 78 3E. When visualized in the format described in section 2.5, it appears like this: ....</roblox>.

22 Bytes
00 00 00 00 00 09 00 00 00 00 00 00 00 3C 2F 72 6F 62 6C 6F 78 3E

Figure 5-7

5.7 BASE TYPES

Every save file generated by Roblox Studio always includes certain types, regardless of place content. Even an empty place will include these types. A listing is given below (accurate 8/27/14):

AdService	FWService	PointsService	SoundService
AssetService	GamePassService	RenderHooksService	StarterGui
BadgeService	Geometry	ReplicatedFirst	StarterPack
CSGDictionaryService	Instance	ReplicatedStorage	TeleportService
Camera	Lighting	ScriptInformationProvider	Terrain
ChangeHistoryService	LogService	ScriptService	TimerService
CollectionService	NonReplicatedCSGDictionaryService	Selection	UserInputService
ContextActionService	NotificationService	ServerScriptService	Workspace
CookiesService	PhysicsService	ServerStorage	
Debris	Players	SocialService	

Table 1 - Included Types

6 Property Data Formats

6.1 Property Type Values

In property data sections (see section 3.3), the type of a given property is denoted by a single bytes. The meaning of these bytes is given here:

ID	Type
0x1	String
0x2	Boolean
0x3	Int32
0x4	Float
0x5	Double
0x7	UDim2
0x8	Ray
0x9	Faces
0xA	Axis
0xB	BrickColor
0xC	Color3
0xD	Vector2
0xE	Vector3
0x10	CFrame
0x12	Enumeration/Token
0x13	Referent

Note: It seems likely that 0x6 refers to UDim. This would be consistent with Float/Double and Vector2/Vector3. However, since no object actually saves UDim values, whether or not 0x6 actually refers to UDim is unknown.

6.2 Property Data Type Storage Formats

6.2.1 String (0x1) (4+ Bytes)

Strings are always prefixed by length (32-bit, little-endian, no transformation), followed by *length* bytes of data. As specified in section 2.5, strings are encoded using ISO/IEC 8859-1. Arrays of strings are stored consecutively (ex. Length1, String1, Length2, String2, etc.).

Raw Data { "Name1", "OtherName" }

Offset	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15
Hex	05	00	00	00	4E	61	6D	65	31	09	00	00	00	4F	74	68	65	72	4E	61	6D	65
Text	.	.	.	.	N	a	m	e	1	.	.	.	.	O	t	h	e	r	N	a	m	e
	Length (0x5)				String ("Name1")					Length (0x9)				String ("OtherName")								

Figure 6-1

6.2.2 Boolean (0x2) (1 Byte)

Boolean values are stored using a single byte. A value of 0 represents false, while a value of 1 indicates true. Arrays of Boolean values are stored consecutively.

Raw Data { True, False, False, True }

Stored

Offset	00	01	02	03
Hex	01	00	00	01

Figure 6-2

6.2.3 Int32 (0x3) (4 Bytes)

Integer values, when stored as property data, are stored in big-endian format, and are subject to byte interleaving (also known as column rearranging) and integer transformation (see sections 2.3.1 and 2.3.2). Single integers can be decoded by applying the reverse of the integer transformation described above. Arrays of integers must first be deinterleaved and then the reverse of the integer transform must be applied.

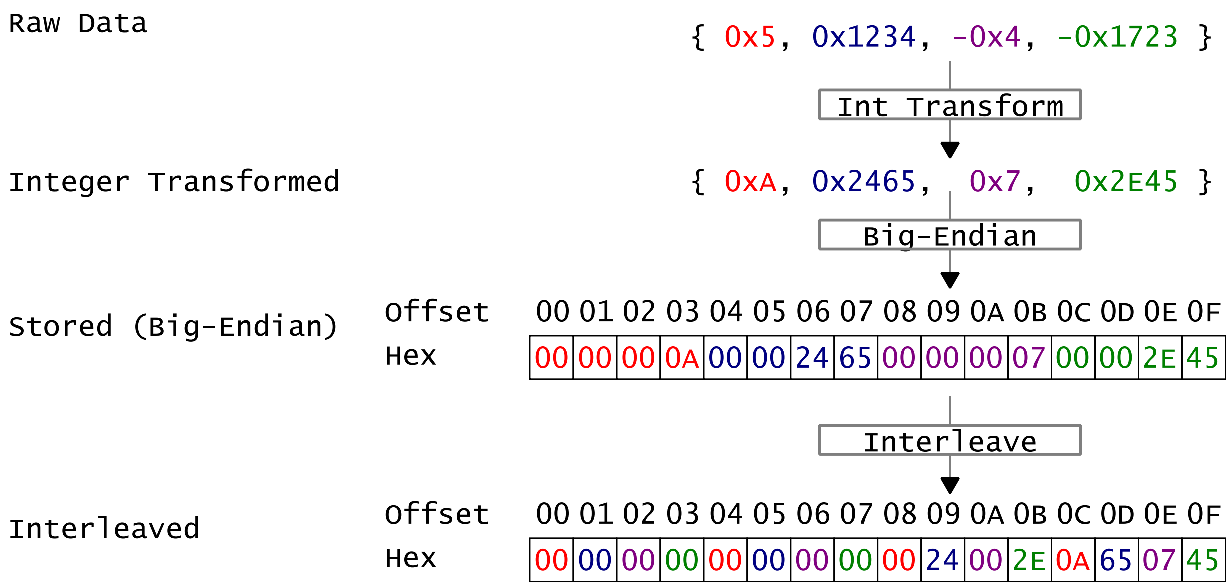


Figure 6-3

6.2.4 Float (0x4) (4 Bytes)

Float values, when stored as property data, are stored in big-endian/Roblox format (see 2.3.3) and are subject to byte interleaving (also known as column rearranging) (see sections 2.3.1). Arrays of integers must first be deinterleaved and then reverse float transformed.

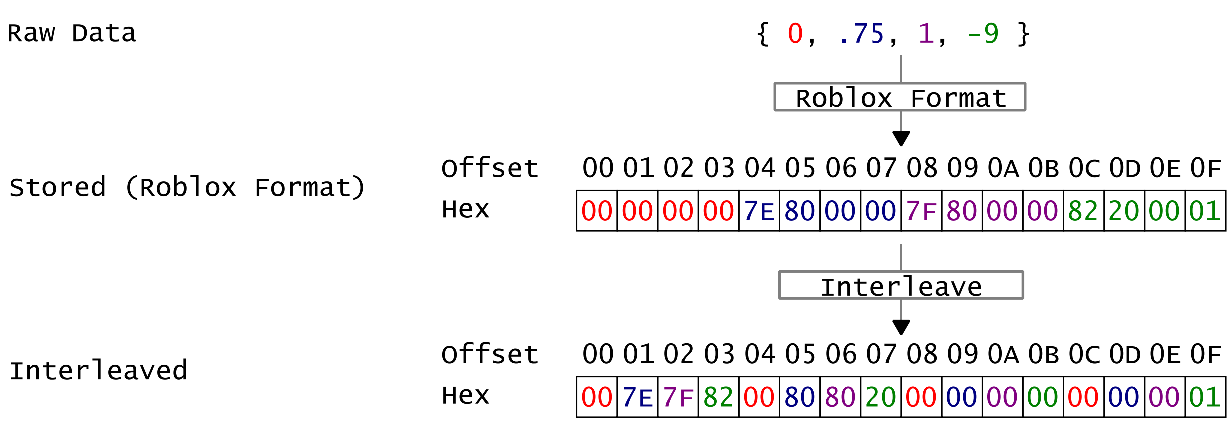


Figure 6-4

6.2.5 Double (Lua Number) (0x5) (8 Bytes)

Double values are stored in IEEE 754 format. No transformation is applied. Bytes are not interleaved. They are stored in little-endian format. When stored in an array, values are stored consecutively.

Raw Data: { 737.535, -25 }

↓ Little-Endian

IEEE 754 64-Bit (Little Endian)

Offset	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
Hex	E1	7A	14	AE	47	0C	87	40	00	00	00	00	00	00	39	C0

Figure 6-5

6.2.6 UDim2 (0x7) (16 Bytes)

UDim2 consist of four values, ScaleX, ScaleY, OffsetX, and OffsetY. Scale values are stored as transformed floating point values (see 2.3.3). Offset values are stored as signed, transformed 32-bit integers (see 2.3.2). The order of values is ScaleX, ScaleY, OffsetX, OffsetY. When an array of UDim2 values is stored, all of each value is stored together and interleaved. For example, all of the ScaleX values are listed first and byte-interleaved. Then all of the ScaleY values. Then OffsetX. Then OffsetY. Each group is byte-interleaved independently of the others.

Raw Data: { {(.1,100),(.6,-200)}, {(0,20),(1,-50)} }

↓ Store

Stored

Offset	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
Hex	7B	99	99	9A	00	00	00	00	7E	33	33	34	7F	00	00	00
	ScaleX								ScaleY							

Offset	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
Hex	00	00	00	C8	00	00	00	28	FF	FF	FF	38	FF	FF	FF	CE
	OffsetX								OffsetY							

↓ Interleave

Interleaved

Offset	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
Hex	7B	00	99	00	99	00	9A	00	7E	7F	33	00	33	00	34	00

Offset	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
Hex	00	00	00	00	00	00	C8	28	FF	FF	FF	FF	FF	FF	38	CE

Figure 6-6

6.2.7 Ray (0x8) (24 Bytes)

Rays consist of six floating point values, stored in the following order: OriginX, OriginY, OriginZ, DirectionX, DirectionY, and DirectionZ. Each value is stored as a little-endian IEEE 754 32-bit floating point value. Arrays of Rays are stored consecutively. There are no transformations or interleaving. All of the values of the first ray are stored before those of the second, and so on.

Raw Data { {(1,2,3),(.33,.66,.66)}, {(-2,5,7),(-.8,-.36,.48)} }

Origin Direction

Store

Stored

Offset	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
Hex	00	00	80	3F	00	00	00	40	00	00	40	40	C3	F5	A8	3E

Offset	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
Hex	C3	F5	28	3F	C3	F5	28	3F	00	00	00	C0	00	00	A0	40

Offset	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
Hex	00	00	E0	40	CD	CC	4C	BF	EC	51	B8	BE	8F	C2	F5	BE

Figure 6-7

6.2.8 Faces (0x9) (1 Byte)

Faces values consist of a single byte representing a bit field. The highest-order two bits appear to always be zero when saved by Roblox Studio. If changed, their values are ignored. The remaining bits represent flags, listed from highest-order bit to lowest: Right, Top, Back, Left, Bottom, and Front.

Bitfield Format

		Right	Top	Back	Left	Bottom	Front
0	0	1	1	1	1	1	1

Raw Data { (Back,Front), (Top,Left), (Right,Left,Bottom) }

Bits

Bitfield { 001001, 010100, 100110 }

Store

Stored

Offset	00	01	02
Hex	09	18	25

Figure 6-8

6.2.9 Axis (0xA) (1 Byte)

Axis values consist of a single byte representing a bit field. The highest-order five bits appear to always be zero when saved by Roblox Studio. If changed, their values are ignored. The remaining three bits represent flags, listed from highest-order bit to lowest: Z, Y, then X.

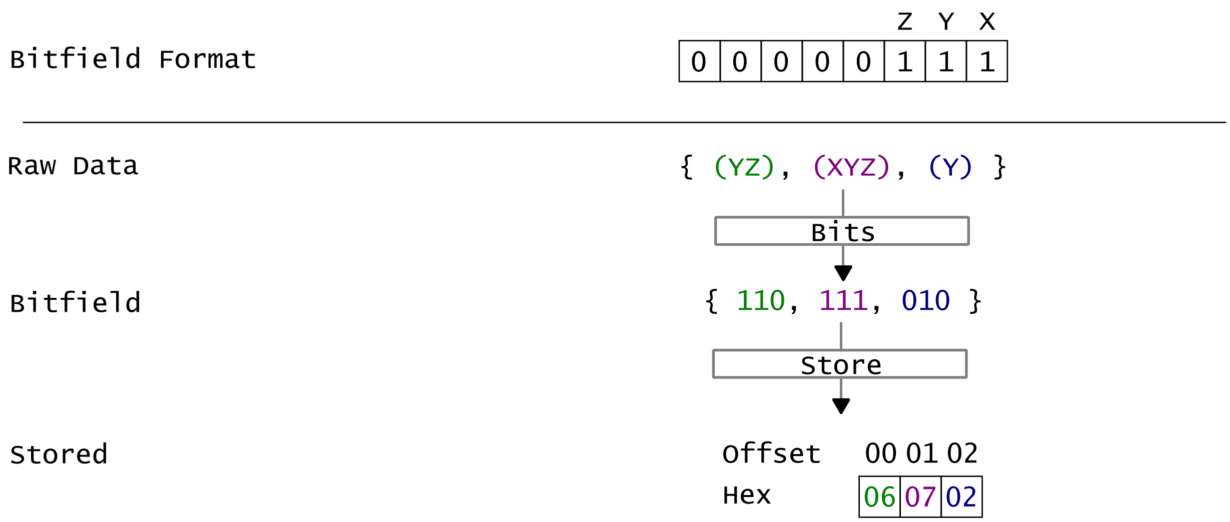


Figure 6-9

6.2.10 BrickColor (0xB) (4 Bytes)

BrickColor values are stored as big-endian 32-bit integers. They are not transformed using the integer transform described in section 2.2. The value of each BrickColor is well known and documented on the Wiki (wiki.roblox.com). Arrays of BrickColor values are byte-interleaved (see section 2.1).

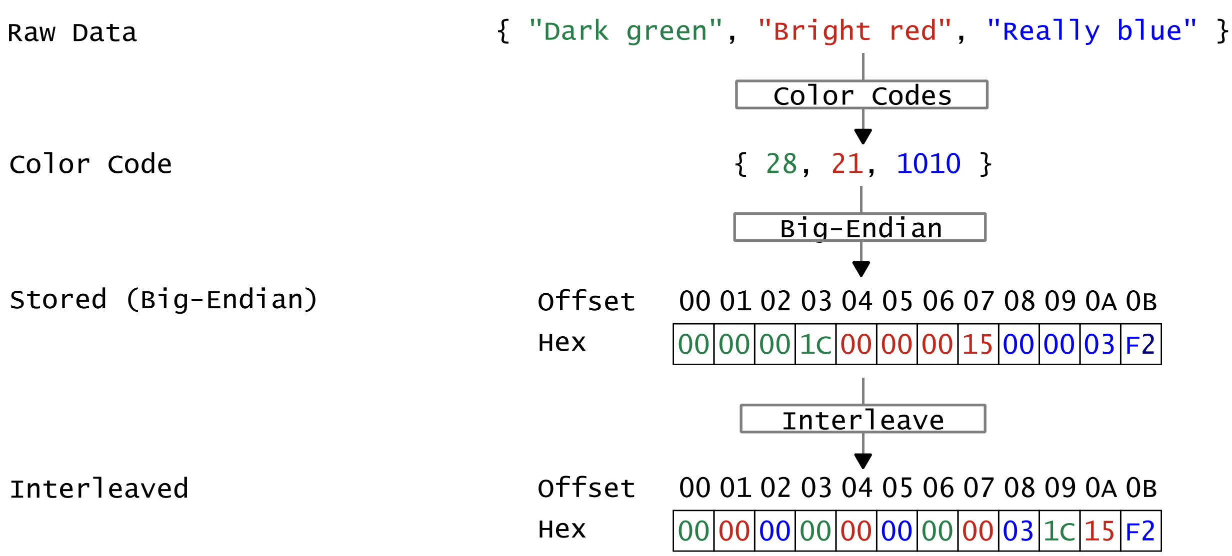


Figure 6-10

6.2.11 Color3 (0xC) (12 Bytes)

Color3 values consist of three 32-bit floating point values (Roblox format, see section 2.3), stored in the following order: R, G, B. When arrays of Color3 values are stored, R values, G values, and B values are grouped together and interleaved independently.

Raw Data: { (51, 130, 87), (113, 30, 130) }

↓ To Float (X/255)

Float: { (.2, .5098, .3412), (.4431, .1176, .5098) }

↓ Roblox Format

Stored (Roblox Format)

Offset	00	01	02	03	04	05	06	07	08	09	0A	0B
Hex	7C	99	99	9A	7D	C5	C5	C8	7E	05	05	06

Offset	0C	0D	0E	0F	10	11	12	13	14	15	16	17
Hex	7B	E1	E1	E4	7D	5D	5D	5E	7E	05	05	06

↓ Interleave

Interleaved

Offset	00	01	02	03	04	05	06	07	08	09	0A	0B
Hex	7C	7D	99	C5	99	C5	9A	C8	7E	7B	05	E1

Offset	0C	0D	0E	0F	10	11	12	13	14	15	16	17
Hex	05	E1	06	E4	7D	7E	5D	05	5D	05	5E	06

Figure 6-11

6.2.12 Vector2 (0xD) (8 Bytes)

Vector3 values consist of two 32-bit floating point values, stored in the following order: X, Y. Each float is stored in the manner described in section 2.3 (Roblox format). When arrays of Vector2 values are stored, X values and Y values are grouped separately and byte-interleaved separately. When converted float values to a scale of 0-255, values appear to always round down.

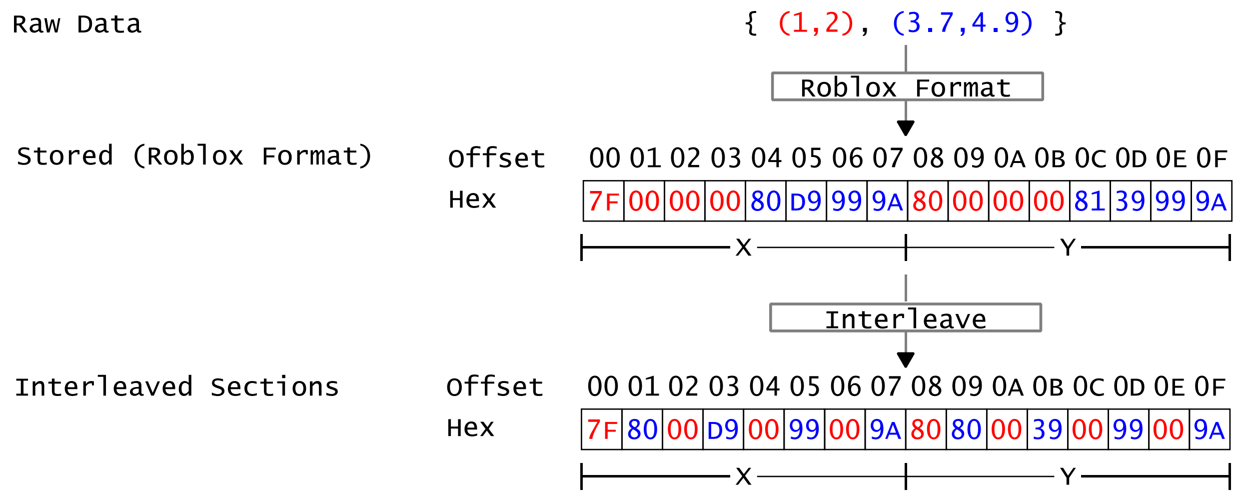


Figure 6-12

6.2.13 Vector3 (0xE) (12 Bytes)

Vector3 values consist of three 32-bit floating point values, stored in the following order: X, Y, Z. Each float is subject to the floating point transformation described in section 2.3. When arrays of Vector3 values are stored, X values, Y values, and Z values are grouped together and byte-interleaved separately

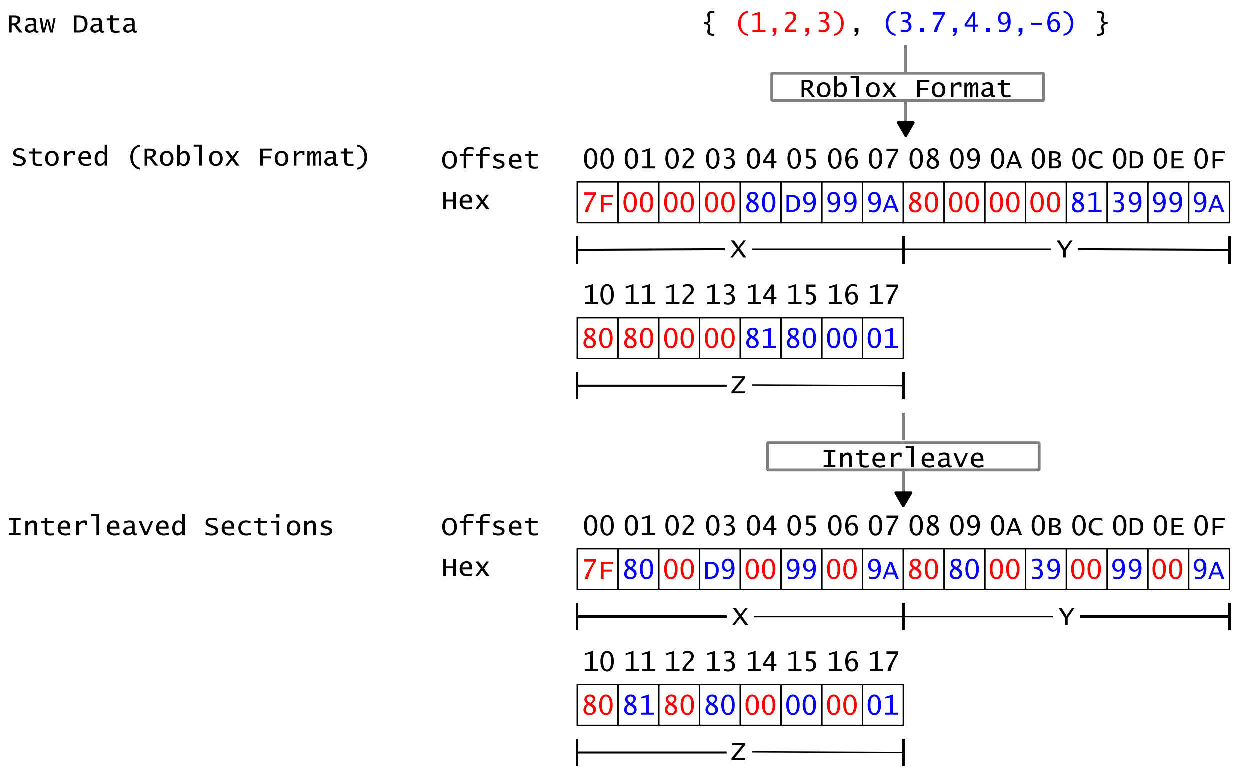


Figure 6-13

6.2.14 CFrame (0x10) (13/49 Bytes)

CFrame storage is more complex than other types. There are 24 special values which require only 13 bytes to store, instead of the full 49 bytes required to store position and rotation matrix values. Each CFrame value starts with a single byte, which indicates whether or not the following value is a short value, and if so, which it is. When stored individually, if the value is not a short value, then 9 32-bit untransformed, little-endian, floating-point values are stored (R00, R01, R02, R11… R22). The three position values (X, Y, and Z) come at the end. They are stored as big-endian, transformed floats (see section 2.3). When CFrames are stored in arrays, the data is split into two parts: special value bytes & matrix values, and position data. For each CFrame value, the special value byte is written first, and then the matrix values are written directly after, if present. Following the matrix/special byte data for all values, the position data is written. Position data is stored as a Vector3 array (see section 4.2.13), bytes are interleaved within categories (X, Y, Z).

The “special byte” values are given to the below:

Special Byte/Rotation Angle Pairs

0x02	(0, 0, 0)	0x0D	(-90, 0, -90)	0x19	(0, 0, -90)
0x03	(90, 0, 0)	0x0E	(0, -90, 0)	0x1B	(90, -90, 0)
0x05	(-180, 0, 0)	0x10	(90, 0, 90)	0x1C	(180, 0, 90)
0x06	(-90, 0, 0)	0x11	(180, 90, 0)	0x1E	(-90, 90, 0)
0x07	(-180, 0, -90)	0x14	(-180, 0, -180)	0x1F	(90, 0, -90)
0x09	(90, 90, 0)	0x15	(-90, 0, -180)	0x20	(0, 90, 0)
0x0A	(0, 0, 90)	0x17	(0, 0, -180)	0x22	(-90, 0, 90)
0x0C	(-90, -90, 0)	0x18	(90, 0, -180)	0x23	(-180, -90, 0)

Figure 6-14

		Position	Rotation Angles	Type	Rotation Matrix
Data	[0]	(2,3,4)	(40,50,60)	(0x0)	[.32 -.56 .77; .91 -.04 -.41; .26 .83 .49]
	[1]	(18,19,12)	(0,0,0)	(0x2)	[.32 -.56 .77; .91 -.04 -.41; .26 .83 .49]
	[2]	(5,6,7)	(10,20,30)	(0x0)	[.81 -.47 .34; .54 .82 -.16; -.20 .32 .93]
	[3]	(14,22,0)	(90,0,90)	(0x10)	[.81 -.47 .34; .54 .82 -.16; -.20 .32 .93]

Stored

Matrix Data: rows 00–40; Position Data: rows 50–70

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	00	BA	8D	A4	3E	F4	81	0E	BF	7D	1B	44	3F	96	DC	68
10	3F	E0	D0	31	BD	CB	8B	D3	BE	CE	C9	86	3E	8E	60	54
20	3F	5D	1C	FC	3E	02	00	0B	55	50	3F	B2	8F	F0	BE	43
30	1D	AF	3E	FA	38	0B	3F	76	BB	52	3F	98	17	27	BE	83
40	CA	51	BE	34	39	A3	3E	19	E8	6C	3F	10	80	83	81	82
50	00	20	40	C0	00	00	00	00	00	00	00	00	80	83	81	83
60	7F	30	80	60	FF	00	00	00	FE	00	00	00	81	82	81	00
70	00	80	C0	00	00	00	00	00	00	00	00	00				

*Matrix values are rounded to two digits for display.

Figure 6-15

6.2.15 Enumeration/Token (0x12) (4 Bytes)

Enumeration values stored as 32-bit big-endian integers, subject to byte-interleaving (section 2.1). The integer transformation is not applied. For the meaning of specific enumeration values, see the wiki (wiki.roblox.com).

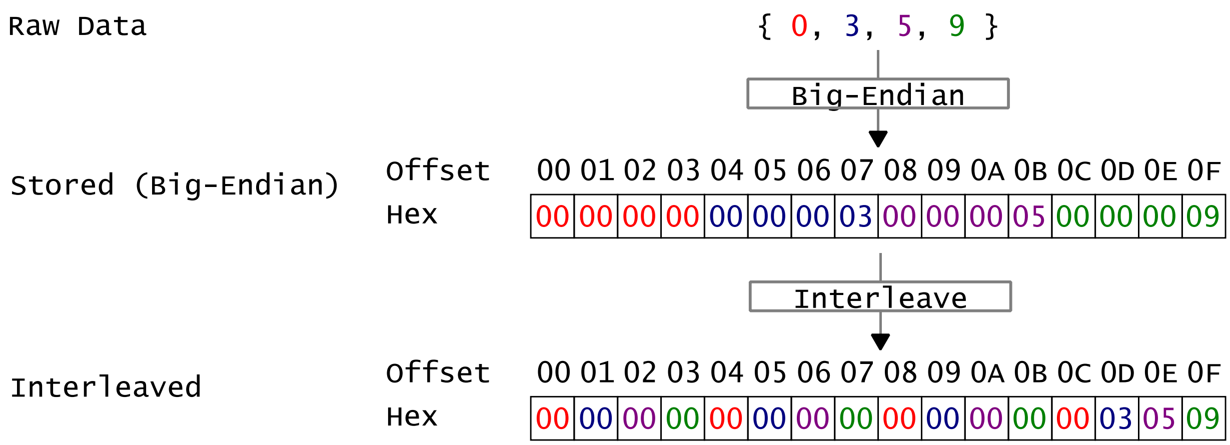


Figure 6-16

6.2.16 Referent (0x13) (4 Bytes)

Referent values stored as 32-bit, big-endian, signed integers, subject to both integer transformations (section 2.2) and byte-interleaving (section 2.1). See sections 2.4 and 3.2 for a description of referents. Referent arrays **do not** store the referents directly, rather they store the difference between the value and the previous. The array { 1, 4, 3, 6 } would be stored as {1, 3, -1, 3}.

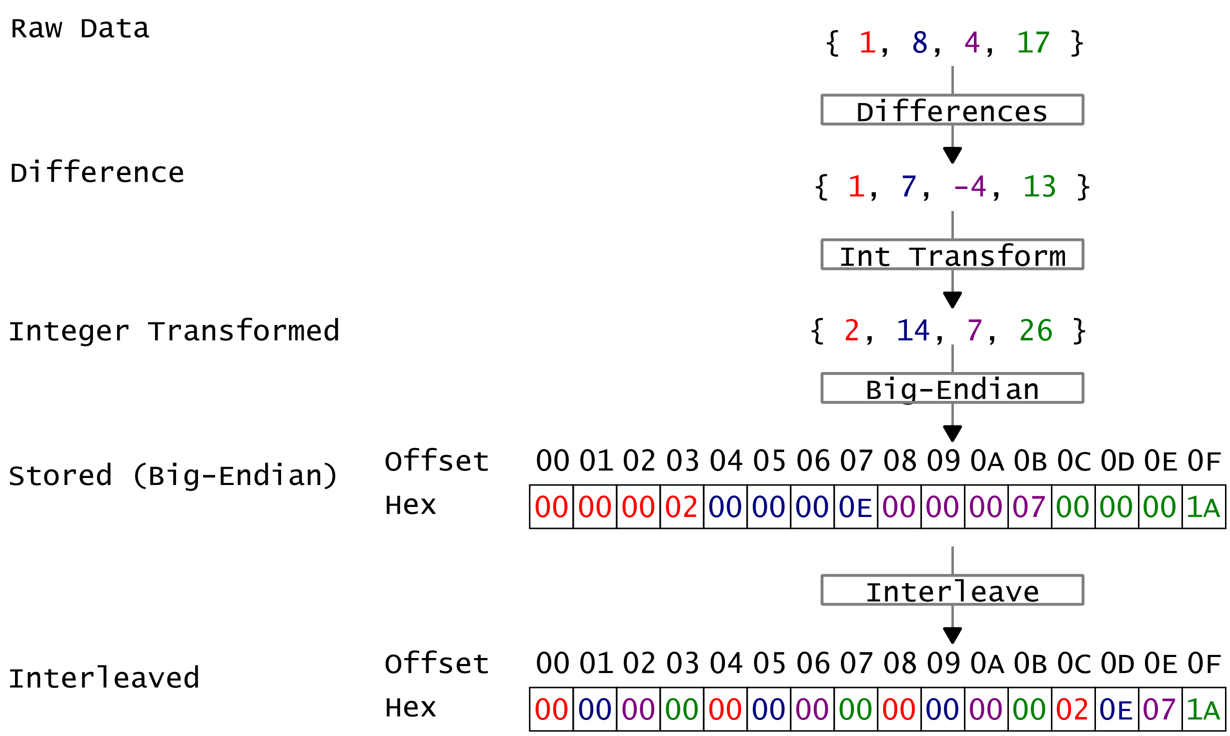


Figure 6-17

7 TERRAIN

7.1 TERRAIN OVERVIEW

While Roblox Terrain is technically property data, it is drastically different from any other property or data type in this document. As such, it deserves a section of its own. Terrain data is stored in the ClusterGridV3 property of the Terrain object. The terrain object is contained in the Workspace. It is technically stored as a “string” value, but is better thought of as an array of bytes.

Terrain consists of voxels, referred to as *cells*. Each cell is 4×4×4 game units in size. Each cell’s position is stored as three signed integers, representing the x, y, and z position of the cell. Given cell coordinates *x*, *y*, and z, the bounds of the cell are given by (4x, 4y, 4z) and (4x + 4, 4y + 4, 4z + 4). Currently, cells must stay within (-32000, -16000, -32000) and (32000, 16000, 32000).

Cells are organized into *chunks*, each representing a 32×16×32 cell area, an area of 16384 cells. Chunks form a grid, similar to cells. For the purposes of this document, each chunk will be represented by three signed integer coordinates, representing the x, y, and z axes. Given the chunk coordinates *x*, *y*, and z, the bounds of the cell region are given by (32x, 16y, 32z) and (32x + 31, 16y + 15, 32z + 31). Thus, chunk (0, 0, 0) contains cells (0, 0, 0) through (31, 15, 31).

Each cell has a *material* type. There are currently 17 possible material values (including water, which is not stored explicitly). A listing of material values is given in section 6.3.2.3. For all materials except for water, the cell also has a *block* type, representing the shape of the cell, as well as an *orientation* value, representing the direction which the cell faces. A listing of block and orientation values is given in sections 6.3.2.1 and 6.3.2.2.

Water cells are different. Each cell contains a water force, specifying the strength of the water cell’s current, and a water direction, specifying the flow direction flow the water cell. A listing of force and direction values for water cells is given in sections 6.3.3.1 and 6.3.3.2.

7.2 Storage Format

Because the terrain data is technically a string, the first four bytes of data are a 32-bit integer (little-endian) representation of the remaining length. Following this comes the real data. The data can be thought of as an array of chunks. To read the data, simply read a chunk from the stream, repeating until the end of the data is reached. Each chunk starts with three signed 16-bit integer values, representing the chunk's position. They are ordered x, y, then z.

The remaining chunk data stores the cells present in the chunk. Each cell has two 16-bit values associated with it. Their meaning varies depending on the type of cell. For development purposes, they can be considered unsigned 16-bit integers. For the purposes of this document, these values will be called *d1* and *d2*. The chunk data after the chunk x, y, and z values is split into two sections. The first contains the d1 values for each cell. The second contains the d2 values.

Each chunk contains an array of 0x4000 cells. Some of these cells may be empty. Cell are numbered 0x0 to 0x3FFF. Cell 0 is at offset (0, 0, 0), relative to the start of the chunk. To get the next cell, add one to x. If x reaches 32, wrap it around to 0 and add 1 to z. If z reaches 32, wrap it around to 0 and add 1 to y. To get the position of cell number *n*, consider the bits of n. The first five bits represent the relative x position. The next five represent the relative z position. The next four represent the relative y position. The remaining bits should be set to zero.

Roblox reduces file size by storing cells in what are referred to in this document as runs. A run is group of cells with consecutive indices. Runs have a length. A run of cells starting at cell 7, with a length of 4 would include cells 7, 8, 9, and 10. Runs can wrap around edges. For example, a run starting at (31, 2, 3), with a length of 4 would include cells (31, 2, 3), (0, 2, 4), (1, 2, 4), and (2, 2, 4).

Terrain Cell Numbering

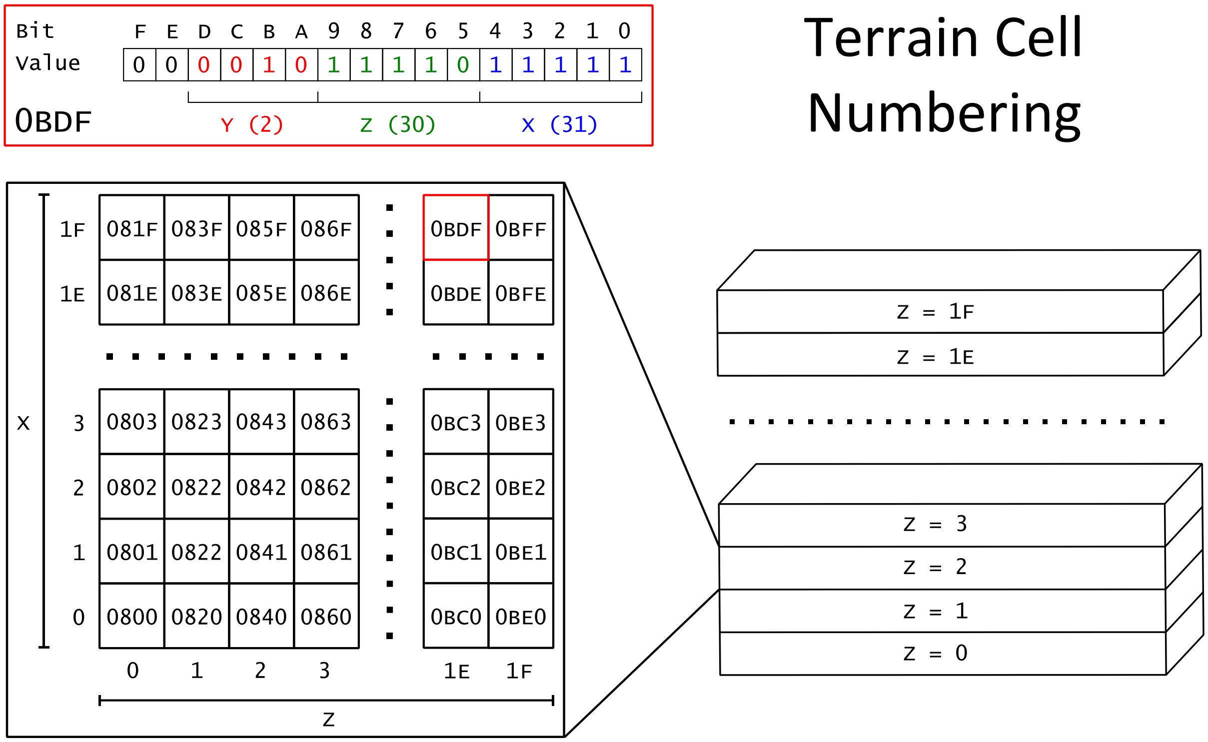


Figure 7-1

The first section of the chunk data, which stores d1 values, can be read by repeatedly reading segments. Segments whose lead byte is 0x28 represent a run of empty cells. Their offset value represents the number of empty cells in a row. If the lead bytes is not equal to 0x28, then the segment represents a run of non-empty cells. The lead byte indicates the d1 value for each cell in the run, and the offset value indicates the length of the run.

The section containing d1 values ends when the end of the chunk is reached. This is equivalent to the current cell index being 0x4000, which is one more than (31, 15, 31). Often, much of the chunk is empty. In many cases like these, the last segment in the section is 0x28 section containing the distance between the last full cell and the end. This is not a special case and can be treated the same as if the section is completely full.

Following the d1 section comes the d2 section. It is structured exactly the same as the d1 section, but with one difference. Instead of empty segments starting with 0x28, they start with 0x11. Everything else is the same. The lengths of segments in the d2 section need not correspond with those in the d1 section. They do, however, refer to the same cells. For example, the d1 section could contain a run of 8 cells, while the d2 section could contain a run 2, then a run of 3, and then a run of 2. All cells in a run have the same d1/d2 value. Neighboring cells that share d1/d2 values are compressed into runs to reduce storage space. The exception to this is water cells. Water cells do not have a d2 value. In the d2 section, they are treated like empty cells.

7.3 Determining Cell Attributes From D1/D2 Values

The meaning of d1 and d2 values differs for water cells and non-water cells. The meaning in each case is described in detail below.

7.3.1 Determining Whether a Cell is a Water Cell

Water cells do not explicitly store a material value. Adopting the convention that the lowest order bit is bit 0, determining if a cell is or is not a water cell can be done by inspecting bits 3-5. The cell is a water cell only if these bits equal 101_2 (5_{10}).

7.3.2 Non-Water Cells

Non-water cells store three attribute values: material, block type, and orientation. The values for each type are given below. Their values match the corresponding Roblox Lua-available enumerations at the time of writing (excluding the material value for water, as water is stored differently). Their values are given below:

7.3.2.1 Block Value

Adopting the convention that the lowest-order bit is bit 0, the Block value is contained in bits 3-5 of the d1 value.

Value	Block
0	Square Block
1	Vertical Wedge
2	Corner Wedge
3	Inverse Corner Wedge
4	Horizontal Wedge

Figure 7-2

7.3.2.2 Orientation Value

Adopting the convention that the lowest-order bit is bit 0, the Block value is contained in bits 6-7 of the d1 value.

Value	Orientation
0	-Z
1	X
2	Z
3	-X

Figure 7-3

7.3.2.3 Material Value

The material value is the d2 value.

Value	Orientation
0	Empty
1	Grass
2	Sand
3	Brick
4	Granite
5	Asphalt
6	Iron
7	Aluminum
8	Gold
9	Wooden Plank
10	Wooden Log
11	Gravel
12	Cinder Block
13	Mossy Stone
14	Cement
15	Red Plastic
16	Blue Plastic

7.3.3 Water Cells

Non-water cells store two attribute values: water direction and water force. The values for each type are given below. Their values match the corresponding Roblox Lua-available enumerations at the time of writing. Their values are given below:

7.3.3.1 Water Direction

To retrieve the Water Direction value, being by concatenating the highest order two bits of d1 with the lowest order three bits of d1. Call this number n. Water Direction is equal to $floor(\frac{n-1}{6})$.

Value	Water Direction
0	-X
1	X
2	-Y
3	Y
4	-Z
5	Z

7.3.3.2 Water Force

To retrieve the Water Force value, being by concatenating the highest order two bits of d1 with the lowest order three bits of d1. Call this number n. Water Force is equal to $(n - 1)\ \%\ 6$.

Value	Water Force
0	None
1	Small
2	Medium
3	Strong
4	Max

Figure 7-4

7.4 Terrain Examples

7.4.1 A Single Cell (Origin)

The first example here contains a single terrain cell at position (0, 0, 0). Its material is grass. Looking at the decompressed hex data below, it shows that the total length of the data is **0x12** bytes, not including the length itself.

Next comes the chunk position. The first (and only) chunk in this example is located at **(0, 0, 0)**. This means that the position within the chunk is the same as the world space value for each cell. The next two bytes indicate that the next **0x1** cell(s) have a Block value of **0** and an Orientation value of **0**. Since this is the first cell defined in the chunk, it is at position (0, 0, 0). The following four bytes indicate the remaining 0x3FFF cells are blank.

The next two bytes indicate that the next **0x1** cell(s) have a material value of **0x1**. Once again, since this is the first cell defined, this refers to the cell at (0, 0, 0). The final four bytes indicate that the remaining 0x3FFF cells have no material (because they are blank).

Decompressed:

```
12 00 00 00 00 00 00 00 00 00 00 01 28 FF 3F FF 01 01 11 FF 3F FF
```

Position	X	Y	Z	Block	Orientation	Bytes		Material	Bytes
				Chunk (0, 0, 0)				00 00 00 00 00 00	
0x0	**0**	**0**	**0**	0 (Solid)	0 (Neg-Z)	00 01		1 (Grass)	01 01
				Skip 0x3FFF Cells		28 FF 3F FF		Skip 0x3FFF Cells	11 FF 3F FF

7.4.2 A Single Cell

This next example is similar to the first, but with two critical differences. The first is that the cell is no longer located in chunk (0, 0, 0). Instead the chunk header indicates that the chunk position is **(26, 33, -19)**. The Z value is negative, since the values are stored as 16-bit signed integers. Since each chunk is 32×16×32 units, this means that each cell's position is equal to the offset in the chunk + (832, 528, -608). Second, the cell is not located at offset (0, 0, 0) from the start of the chunk. **0x10F5** cells are skipped before declaring the cell, leading to an offset of 0x10F5 (21, 4, 7). The offset of the chunk is added to the offset within the chunk to compute the location of the cell in world coordinates. The cell's location is (853, 532, -601).

Decompressed:

```
1A 00 00 00 1A 00 21 00 ED FF 28 FF 10 F5 00 01 28 FF 2F 0A 11 FF 10 F5
01 01 11 FF 2F 0A
```

Position	X	Y	Z	Block	Orientation	Bytes		Material	Bytes
				Chunk (26, 33, -19)				1A 00 21 00 ED FF	
				Skip 0x10F5 Cells		28 FF 10 F5		Skip 0x105F Cells	11 FF 10 F5
0x0	**0**	**0**	**0**	0 (Solid)	0 (Neg-Z)	00 01		1 (Grass)	01 01
				Skip 0x2F0A Cells		28 FF 2F 0A		Skip 0x2F0A Cells	11 FF 2F 0A

7.4.3 Two Cells, Two Chunks

This example contains two separate terrain cells in two separate chunks. The first chunk is chunk **(1, 1, 1)**. This means that each cell contained in it is offset by (32, 16, 32). The first bytes in the chunk data indicate that the first **0x0D44** cells are skipped. One cell is declared after that (at offset (4, 3, 10)). In world coordinates, this is (36, 19, 42).

The second chunk is chunk **(0, 0, 0)**. This means that chunk offsets are equal to world coordinates. The cell in this chunk is at location (5, 8, 3). Its material is grass (**0x1**) and the orientation and block values are both **0**.

Decompressed:

```
34 00 00 00 01 00 01 00 01 00 28 FF 0D 44 00 01 28 FF 32 BB 11 FF 0D 44
01 01 11 FF 32 BB 00 00 00 00 00 00 28 FF 20 65 00 01 28 FF 1F 9A 11 FF
20 65 01 01 11 FF 1F 9A
```

Position	X	Y	Z	Block	Orientation	Bytes		Material	Bytes
				Chunk (1, 1, 1)				01 00 01 00 01 00	
				Skip 0xD44 Cells		28 FF 0D 44		Skip 0xD44 Cells	11 FF 0D 44
0xD44	**4**	**3**	**10**	0 (Solid)	0 (Neg-Z)	00 01		1 (Grass)	01 01
				Skip 0x32BB Cells		28 FF 32 BB		Skip 0x32BB Cells	11 FF 32 BB
				Chunk (0, 0, 0)				00 00 00 00 00 00	
				Skip 0x2065 Cells		28 FF 20 65		Skip 0x2065 Cells	11 FF 20 65
0x2065	**5**	**8**	**3**	0 (Solid)	0 (Neg-Z)	00 01		1 (Grass)	01 01
				Skip 0x1F9A Cells		28 FF 1F 9A		Skip 0x1F9A Cells	11 FF 1F 9A

7.4.4 Two Cells, One Chunk

This example is similar to the last, except that both cells are in the same chunk **(0, 0, 0)**. The first four bytes of chunk data (following the header) indicate to skip **0x10A3** cells. The location of the following cell is (3, 4, 5). The data for one cell follows the skip bytes. Next, **0x62** bytes are skipped. The location of the next cell is equal to **0x10A3** (first skip) + 0x1 (for the first cell) + **0x62** (second skip). This corresponds to a chunk offset of (6, 4, 8).

```
Decompressed:
        26 00 00 00 00 00 00 00 00 00 28 FF 10 A3 00 01 28 62 00 01 28 FF 22 F9
        11 FF 10 A3 01 01 11 62 01 01 11 FF 22 F9
```

Position	X	Y	Z	Block	Orientation	Bytes		Material	Bytes
				Chunk (0, 0, 0)					00 00 00 00 00 00
				Skip 0x10A3 Cells		28 FF 10 A3		Skip 0x10A3 Cells	11 FF 10 A3
0x10A3	**3**	**4**	**5**	0 (Solid)	0 (Neg-Z)	00 01		1 (Grass)	01 01
				Skip 0x62 Cells		28 62		Skip 0x62 Cells	11 62
0x1106	**6**	**4**	**8**	0 (Solid)	0 (Neg-Z)	00 01		1 (Grass)	01 01
				Skip 0x22F9 Cells		28 FF 22 F9		Skip 0x22F9 Cells	11 FF 22 F9

7.4.5 A Run

This example introduces a new concept, named a *run* for the purpose of this document. A run is a set of cells with identical Block/Orientation values or material values, located at consecutive positions in a chunk. Essentially, Roblox uses a form of run-length encoding to compactly store terrain data. Single cells can be thought of as runs with a length of one. The run length is given following Block/Orientation values and Material values. Since Material and Block/Orientation bytes are stored separately, they can have separate run lengths (see example 7.4.9). Runs must stay within a chunk and can wrap around to a new Z, then Y value. In this example, the run wraps around to a new Z column.

The run in this example starts at location 0xC5E (30, 3, 2) and ends at 0xC61 (1, 3, 3).

The bytes **00 04** show that there are **0x4** cells in a row that have a Block/Orientation byte equal to **0x0**.
The bytes **01 04** show that there are **0x4** cells in a row that have a material value equal to **0x1**.

Decompressed:

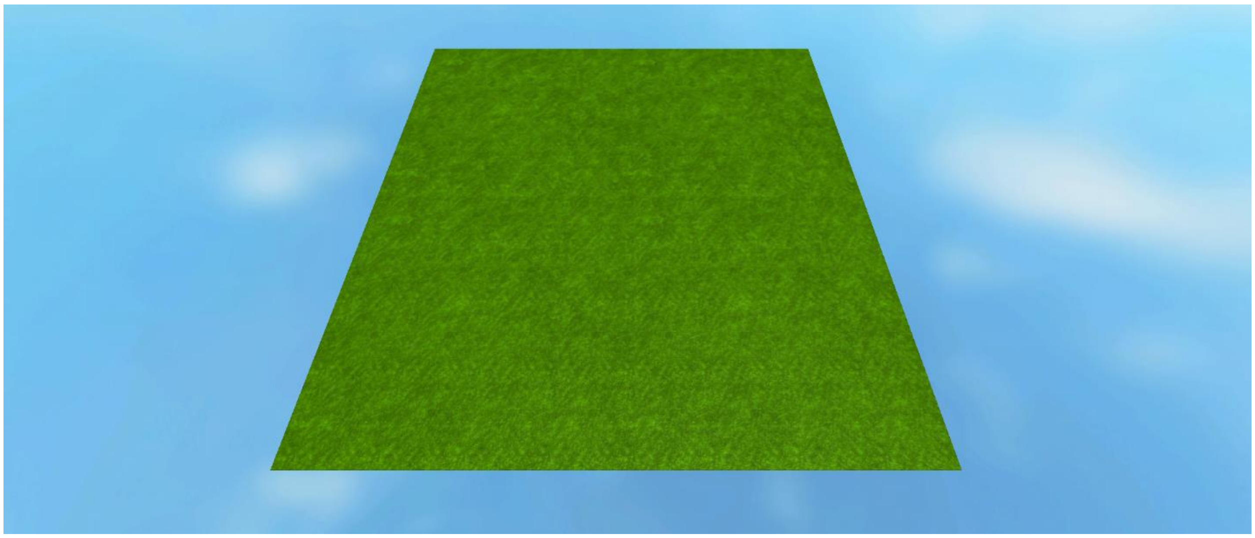
```
1A 00 00 00 00 00 00 00 00 00 28 FF 0C 5E 00 04 28 FF 33 9E 11 FF 0C 5E
01 04 11 FF 33 9E
```

Position	X	Y	Z	Block	Orientation	Bytes		Material	Bytes
				Chunk (0, 0, 0)				00 00 00 00 00 00	
				Skip 0xC5E Cells		28 FF 0C 5E		Skip 0xC5E Cells	11 FF 0C 5E
0xC5E	30	3	2	0 (Solid)	0 (Neg-Z)	00 4		1 (Grass)	01 4
0xC5F	31	3	2						
0xC60	0	3	3						
0xC61	1	3	3						
				Skip 0x339E Cells		28 FF 33 9E		Skip 0x339E Cells	11 FF 33 9E

7.4.6 A Plane

This example demonstrates a 1-thick layer of cells spanning an entire chunk (32×1×32). This is in fact a single run of 1024 (32 × 32) cells, starting at location 0x0. The run length for both Block/Orientation and Material values is 0x400 (1024) cells. Since the run length exceeds that which can be stored in a single byte, a 0xFF byte is prefixed to the true value (stored as two bytes, Little-Endian). This occurs whenever the run length is greater than or equal to 0xFF.

This run starts at location 0x0 (0, 0, 0). It ends at location 0x3FF (31, 0, 31).



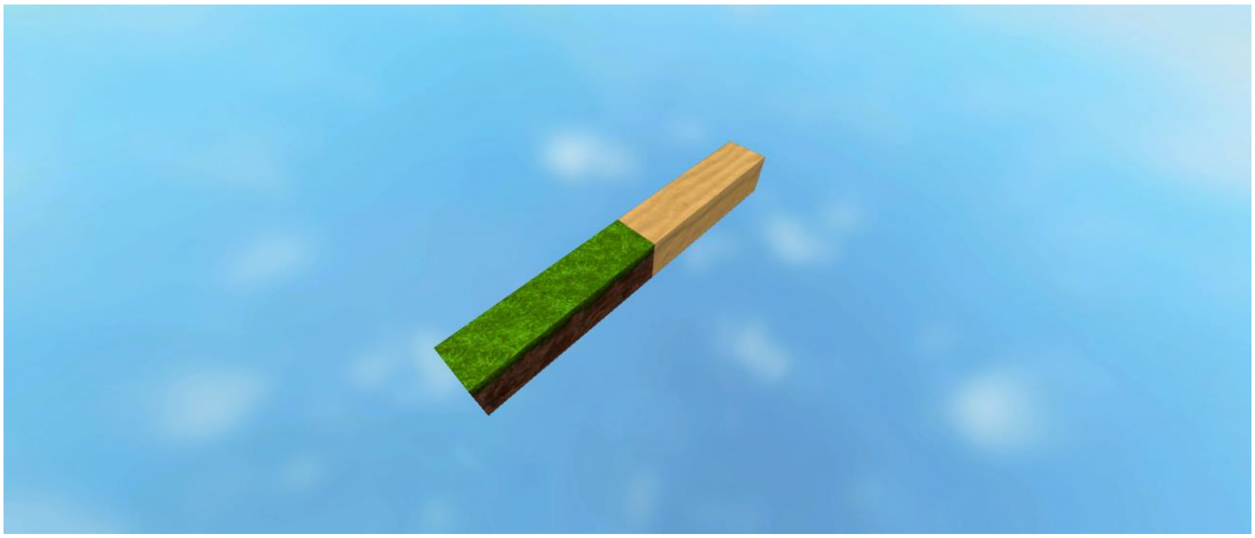
Decompressed:

```
16 00 00 00 00 00 00 00 00 00 00 FF 04 00 28 FF 3C 00 01 FF 04 00 11 FF
3C 00
```

Position	X	Y	Z	Block	Orientation	Bytes		Material	Bytes
				Chunk (0, 0, 0)				00 00 00 00 00 00	
0x0	0	0	0	0 (Solid)	0 (Neg-Z)	00 FF 04 00		1 (Grass)	01 FF 04 00
0x1	1	0	0						
0x2	2	0	0						
. . . .									
0x1F	1F	0	0						
. . . .									
0x20	0	0	1						
0x21	1	0	1						
0x22	2	0	1						
. . . .									
0x3FF	1F	0	1F						
				Skip 0x3C00 Cells		28 FF 3C 00		Skip 0x3C00 Cells	11 FF 3c 00

7.4.7 Multiple Materials

This example shows a chunk containing two different materials. The cells all have the same Block/Orientation byte, and occupy consecutive locations, so the Block/Orientation data is run-length encoded with a length of **0x8**. There are two separate runs of **0x4** for the material values. The first (0x421-0x424) describes the grass cells. The second (0x425-0x428), describes the sand.



Decompressed:

```
1C 00 00 00 00 00 00 00 00 00 28 FF 04 21 00 08 28 FF 3B D7 11 FF 04 21
01 04 02 04 11 FF 3B D7
```

Position	X	Y	Z	Block	Orientation	Bytes		Material	Bytes
				Chunk (0, 0, 0)				00 00 00 00 00 00	
				Skip 0x421 Cells		28 FF 04 21		Skip 0x421 Cells	11 FF 04 21
0x421	**1**	**1**	**1**	0 (Solid)	0 (Neg-Z)	00 08		1 (Grass)	01 04
0x422	**2**	**1**	**1**						
0x423	**3**	**1**	**1**						
0x424	**4**	**1**	**1**						
0x425	**5**	**1**	**1**					2 (Sand)	02 04
0x426	**6**	**1**	**1**						
0x427	**7**	**1**	**1**						
0x428	**8**	**1**	**1**						
				Skip 0x3BD7 Cells		28 FF 3B D7		Skip 0x3BD7 Cells	11 FF 3B D7

7.4.8 Multiple Orientations

This example is nearly the same as the previous, except instead of one Block/Orientation value and two materials, it has two Block/Orientation values and one material value. The first Block/Orientation value (**0x80**) is repeated **0x4** times. The second (**0xC0**) is also repeated **0x4** times. The material value (0x1) is repeated **0x8** times, covering all cells.

Decompressed:

```
1C 00 00 00 00 00 00 00 00 00 28 FF 04 21 80 04 C0 04 28 FF 3B D7 11 FF
04 21 01 08 11 FF 3B D7
```

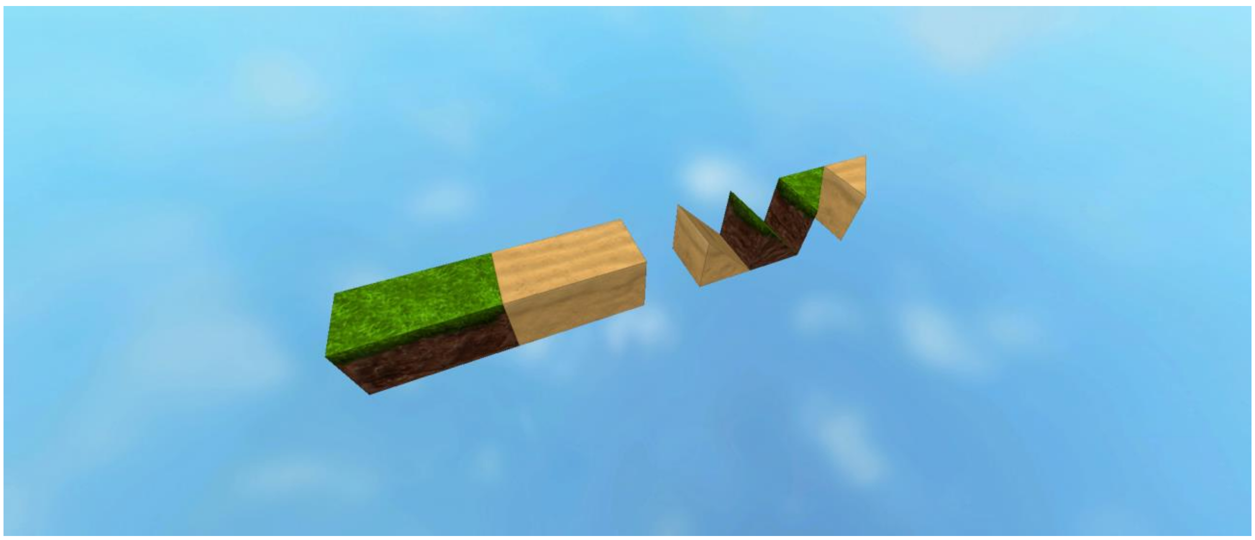
Position	X	Y	Z	Block	Orientation	Bytes		Material	Bytes
				Chunk (0, 0, 0)				00 00 00 00 00 00	
				Skip 0x421 Cells		28 FF 04 21		Skip 0x421 Cells	11 FF 04 21
0x421	**1**	**1**	**1**	0 (Solid)	2 (Z)	80 04		1 (Grass)	01 08
0x422	**2**	**1**	**1**						
0x423	**3**	**1**	**1**						
0x424	**4**	**1**	**1**						
0x425	**5**	**1**	**1**	0 (Solid)	3 (-X)	C0 04			
0x426	**6**	**1**	**1**						
0x427	**7**	**1**	**1**						
0x428	**8**	**1**	**1**						
				Skip 0x3BD7 Cells		28 FF 3B D7		Skip 0x3BD7 Cells	11 FF 3B D7

7.4.9 A Bit of Everything

This example combines several of the concepts illustrated in the previous examples. A table, describing each cell, along with its attributes, is provided for reference. Note that the orientation values and the block values are encoded using separate runs. This occurs often in Roblox Studio-generated save files.

X	Mat	Orientation	Block
1	Grass	X	Solid
2	Grass	Z	Solid
3	Sand	Z	Solid
4	Sand	X	Solid
5	Blank		
6	Sand	X	VerticalWedge
7	Grass	X	VerticalWedge
8	Grass	X	HorizontalWedge
9	Sand	X	HorizontalWedge

(All Y, Z = 1)



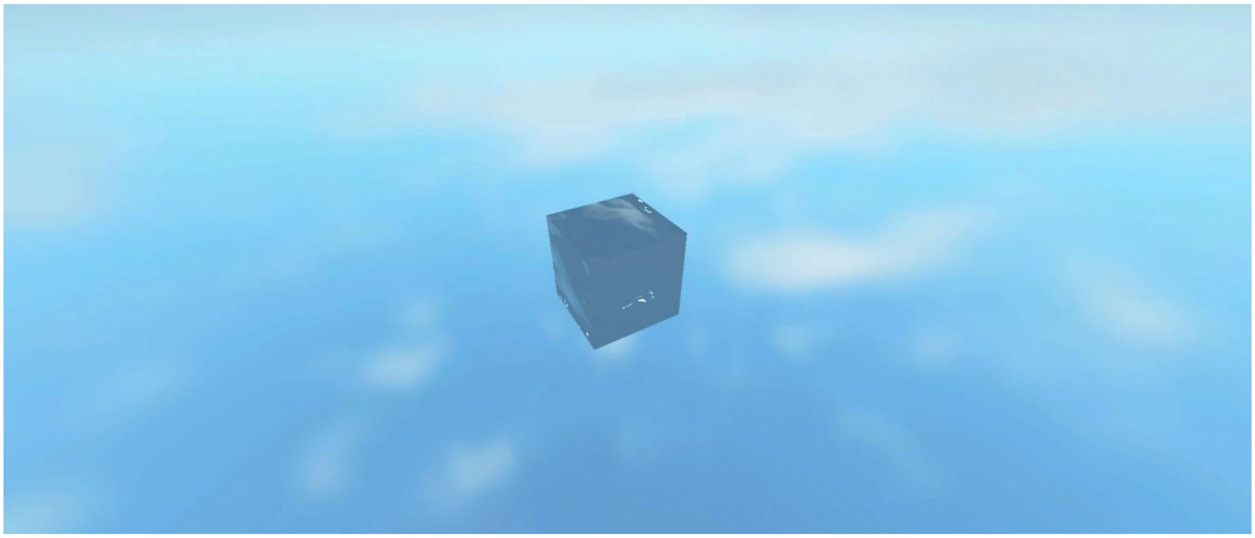
Decompressed:

```
2E 00 00 00 00 00 00 00 00 00 28 FF 04 21 40 01 80 02 40 01 28 01 48 02
60 02 28 FF 3B D6 11 FF 04 21 01 02 02 02 11 01 02 01 01 02 02 01 11 FF
3B D6
```

Position	X	Y	Z	Block	Orientation	Bytes		Material	Bytes
				Chunk (0, 0, 0)				00 00 00 00 00 00	
				Skip 0x421 Cells		28 FF 04 21		Skip 0x421 Cells	11 FF 04 21
0x421	**1**	**1**	**1**	0 (Solid)	1 (X)	40 01		1 (Grass)	01 02
0x422	**2**	**1**	**1**	0 (Solid)	2 (Z)	80 02			
0x423	**3**	**1**	**1**					2 (Sand)	02 02
0x424	**4**	**1**	**1**	0 (Solid)	1 (X)	40 01			
0x425	**5**	**1**	**1**	Skip 0x1 Cell		28 01		Skip 0x1 Cell	11 01
0x426	**6**	**1**	**1**	1 (VWedge)	1 (X)	48 02		2 (Sand)	02 01
0x427	**7**	**1**	**1**					1 (Grass)	01 02
0x428	**8**	**1**	**1**	4 (HWedge)	1 (X)	60 02			
0x429	**9**	**1**	**1**					2 (Sand)	02 01
				Skip 0x3BD6 Cells		28 FF 3B D6		Skip 0x3BD6 Cells	11 FF 3B D6

7.4.10 A Water Cell

This example illustrates a single water cell at location 0x10A3 (3, 4, 5) in chunk **(0, 0, 0)**. Since the material data is not present for water cells, the material section consists solely a single skip (**0x4000** cells). The fact that the cell is a water cell can also be identified by the Block/Orientation byte. If the Block/Orientation byte & 0x29 = 0x29, then the cell is a water cell, as in this case.

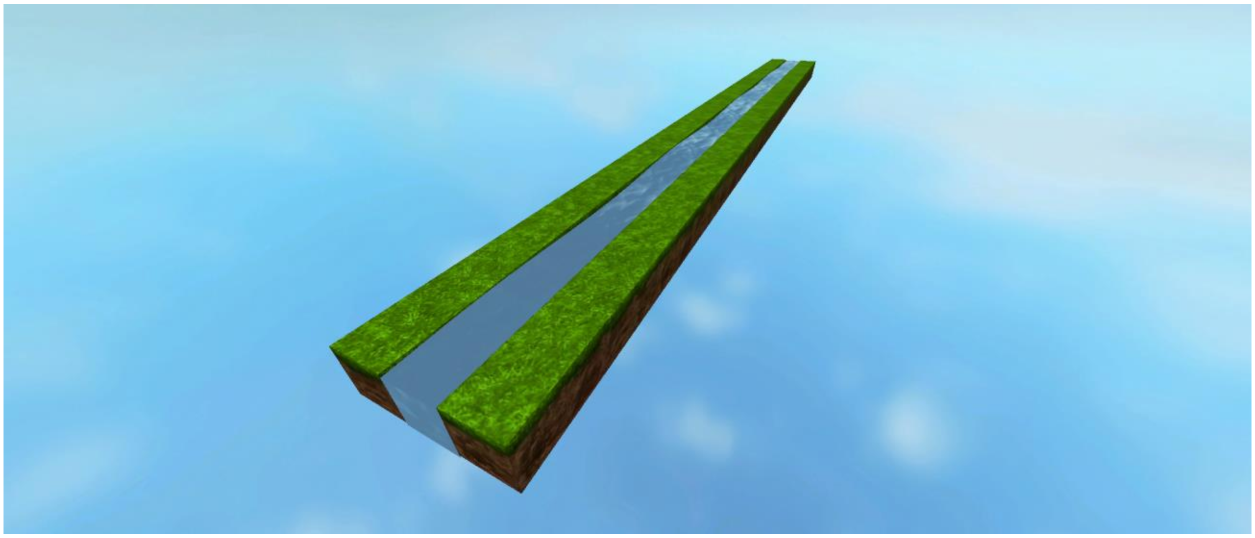


14 00 00 00 00 00 00 00 00 00 28 FF 10 A3 29 01 28 FF 2F 5C 11 FF 40 00

Position	X	Y	Z	Block	Orientation	Bytes		Material	Bytes
				Chunk (0, 0, 0)					00 00 00 00 00 00
				Skip 0x10A3 Cells		28 FF 10 A3		Skip 0x4000 Cells	11 FF 40 00
0x10A3	**3**	**4**	**5**	0 (Solid)	0 (Neg-Z)	29 01			
				Skip 0x2F5C Cells		28 FF 2F 5C			

7.4.11 Water and Land

This example contains three rows of cells. The outer two are grass, while the inner is water. The grass runs from (0, 0, 0) – (31, 0, 0) and from (0, 0, 2) – (31, 0, 2). The water runs from (0, 0, 1) – (31, 0, 1). Since these cells have contiguous location values, the data is encoded using runs. Because water has no material value, those cells are skipped (11 20) in the material section.



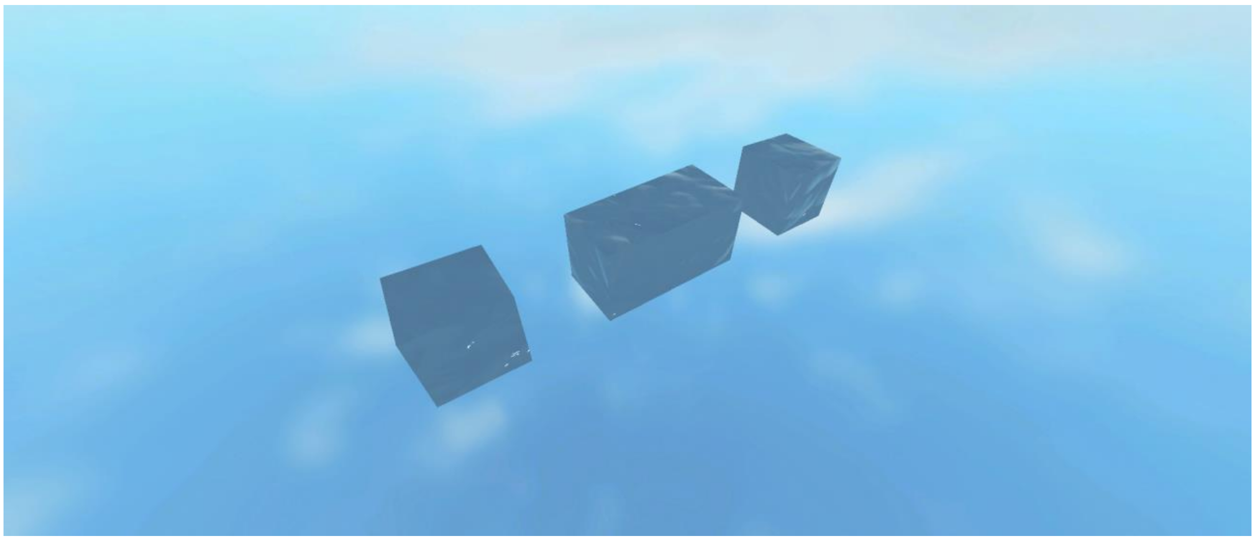
1A 00 00 00 00 00 00 00 00 00 00 20 29 20 00 20 28 FF 3F A0 01 20 11 20
01 20 11 FF 3F A0

Position	X	Y	Z	Block	Orientation	Bytes		Material	Bytes
				Chunk (0, 0, 0)					00 00 00 00 00 00
0x0	0	0	0	0 (Solid)	1 (X)	00 20		1 (Grass)	01 20
. . . .									
0x1F	31	0	0						
0x20	0	0	1	0 (Solid)	1 (X)	29 20		Skip 0x20 Cells	11 20
. . . .									
0x3F	31	0	1						
0x40	0	0	2	0 (Solid)	1 (X)	00 20		1 (Grass)	01 20
. . . .									
0x5F	31	0	2						
				Skip 0x3FA0 Cells		28 FF 3F A0		Skip 0x3FA0 Cells	11 FF 3F A0

7.4.12 Force and Direction

This final example demonstrates storage of Water Force and Direction attributes. This example contains four cells. The Force and Direction values are stored in the same byte as Block/Orientation for non-water cells. The format in which the values are encoded is described in section 7.3.3. For your convenience, the cells and their attributes, as represented in this example, are provided in table form.

X	Y	Z	Direction	Force
2	3	5	-X	None
2	3	7	Y	None
2	3	8	-X	Strong
2	3	10	Z	Medium



20 00 00 00 00 00 00 00 00 00 28 FF 0C A2 29 01 28 3F 2C 01 28 1F AB 01
28 3F AA 01 28 FF 32 BD 11 FF 40 00

Position	X	Y	Z	Block	Orientation	Bytes		Material	Bytes
				Chunk (0, 0, 0)				00 00 00 00 00 00	
				Skip 0xCA2 Cells		28 FF 0C A2		Skip 0x4000 Cells	11 FF 40 00
0xCA2	**2**	**3**	**5**	0 (Solid)	1 (X)	29 01			
				Skip 0x3F Cells		28 3F			
0xCE2	**2**	**3**	**7**	0 (Solid)	1 (X)	2C 01			
				Skip 0x1F Cells		28 1F			
0xD02	**2**	**3**	**8**	0 (Solid)	1 (X)	AB 01			
				Skip 0x3F Cells		28 3F			
0xD42	**2**	**3**	**10**	0 (Solid)	1 (X)	AA 01			
				Skip 0x32BD Cells		28 FF 32 BD			

8 Example File

8.1 Header

```
Offset    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F    ASCII View
00000000  3C 72 6F 62 6C 6F 78 21 89 FF 0D 0A 1A 0A 00 00  <roblox!‰ÿ......
00000010  27 00 00 00 2D 00 00 00 00 00 00 00 00 00 00 00  '...-...........
```

The file starts off with the standard identifier (see section 4.2).

The next 4 bytes represent the number of unique types present in the file (some types have been removed from this file for compactness). There are 0x27 types present.

Next is the number of unique instances (some instances have been removed from this file for compactness). There are 0x2D instances present.

Following this are 8 unknown (possibly reserved) bytes.

8.1.1 Camera

```
00000020  49 4E 53 54 19 00 00 00 17 00 00 00 00 00 00 00  INST............
00000030  F0 08 04 00 00 00 06 00 00 00 43 61 6D 65 72 61  ð.........Camera
00000040  00 01 00 00 00 00 00 00 06                        .........
```

```
Decompressed:
          04 00 00 00 06 00 00 00 43 61 6D 65 72 61 00 01  ........Camera..
          00 00 00 00 00 00 06                              .......
```

The first four bytes indicate that the Type ID is 0x04.

The next four byte indicate that the length of the Type Name is 0x06 bytes.

The next 0x06 bytes show that that the Type Name is "Camera".

The next byte, 0x0, indicates that no additional data is present at the end of the block.

The next four bytes indicate that there is 0x1 camera instance.

The last four bytes represent a referent array (see section 5.2.16). Since there is only one camera instance, the untransformed referent value for this camera is 0x06. The transformed value (see section 3.3.2) is 0x03.

8.1.2 Decal

```
00000040                               49 4E 53 54 18 00 00            INST...
00000050  00 16 00 00 00 00 00 00 00 F0 07 0A 00 00 00 05   .........ð......
00000060  00 00 00 44 65 63 61 6C 00 01 00 00 00 00 00 00   ...Decal........
00000070  0A                                                .
```

Decompressed:

```
          0A 00 00 00 05 00 00 00 44 65 63 61 6C 00 01 00   ........Decal...
          00 00 00 00 00 0A                                 ......
```

The first four bytes indicate that the Type ID is 0x0A.

The next four byte indicate that the length of the Type Name is 0x05 bytes.

The next 0x06 bytes show that that the Type Name is “Decal”.

The next byte, 0x0, indicates that no additional data is present at the end of the block.

The next four bytes indicate that there is 0x1 decal instance.

The last four bytes represent a referent array (see section 5.2.16). Since there is only one camera instance, the untransformed referent value for this camera is 0x0A. The transformed value (see section 3.3.2) is 0x05.

8.1.3 Instance

```
00000070     49 4E 53 54 1E 00 00 00 24 00 00 00 00 00 00    INST....$......
00000080  00 F7 04 0E 00 00 00 08 00 00 00 49 6E 73 74 61   .÷.........Insta
00000090  6E 63 65 01 03 00 01 00 60 4A 04 0A 01 01 01      nce.....`L.....
```

Decompressed:

```
          0E 00 00 00 08 00 00 00 49 6E 73 74 61 6E 63 65   ........Instance
          01 03 00 00 00 00 00 00 00 00 00 00 00 00 4A 04   ..............L.
          0A 01 01 01                                       ....
```

The first four bytes indicate that the Type ID is 0x0E.

The next four byte indicate that the length of the Type Name is 0x08 bytes.

The next 0x06 bytes show that that the Type Name is “Instance”.

The next byte, 0x1, indicates that additional data is present at the end of the block.

The next four bytes indicate that there are 0x3 instances.

The next 0xC bytes represent a referent array (see section 5.2.16). The raw array is { 0x4A, 0x4, 0xA }. These values represent relative offsets. The actual data is { 0x4A, 0x4E, 0x58 }. These values are transformed (see section 3.3.2). The referents are { 0x25, 0x27, 0x2C }.

The last 0x3 bytes always equal 0x1, if present. Their presence is indicated by the byte following the type name.

8.1.4 Lighting

```
00000090                                               49                I
000000A0  4E 53 54 1C 00 00 00 1A 00 00 00 00 00 00 00 F0  NST...........ð
000000B0  0B 0F 00 00 00 08 00 00 00 4C 69 67 68 74 69 6E  .........Lightin
000000C0  67 01 01 00 00 00 00 00 00 5A 01                 g........\.
```

```
Decompressed:
          0F 00 00 00 08 00 00 00 4C 69 67 68 74 69 6E 67  ........Lighting
          01 01 00 00 00 00 00 00 5A 01                    ........\.
```

The first four bytes indicate that the Type ID is 0x0F.

The next four byte indicate that the length of the Type Name is 0x08 bytes.

The next 0x06 bytes show that that the Type Name is "Lighting".

The next byte, 0x0, indicates that no additional data is present at the end of the block.

The next four bytes indicate that there is 0x1 instance.

The last four bytes represent a referent array (see section 5.2.16). Since there is only one camera instance, the untransformed referent value for this instance is 0x5A. The transformed value (see section 3.3.2) is 0x2D.

8.1.5 Part

```
000000C0                                   49 4E 53 54 19            INST.
000000D0  00 00 00 25 00 00 00 00 00 00 00 FD 00 12 00 00  ...%.......ý....
000000E0  00 04 00 00 00 50 61 72 74 00 05 00 01 00 50 04  .....Part.....P.
000000F0  04 02 02 02                                      ....
```

```
Decompressed:
        12 00 00 00 04 00 00 00 50 61 72 74 00 05 00 00   ........Part....
        00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
        04 04 02 02 02                                    .....
```

The first four bytes indicate that the Type ID is 0x12.

The next four byte indicate that the length of the Type Name is 0x04 bytes.

The next 0x06 bytes show that that the Type Name is "Part".

The next byte, 0x0, indicates that no additional data is present at the end of the block.

The next four bytes indicate that there are 0x5 instances.

The next 0x10 bytes represent a referent array (see section 5.2.16). The raw array is { 0x4, 0x4, 0x2, 0x2, 0x2 }. These values represent relative offsets. The actual data is { 0x4, 0x8, 0xA, 0xC, 0xE }. These values are transformed (see section 3.3.2). The referents are { 0x2, 0x4, 0x5, 0x6, 0x7 }.

```
000000F0             49 4E 53 54 1B 00 00 00 19 00 00 00       INST........
00000100    00 00 00 00 F0 0A 15 00 00 00 07 00 00 00 50 6C   ....ð.........Pl
00000110    61 79 65 72 73 01 01 00 00 00 00 00 00 18 01      ayers..........
```

```
Decompressed:
        15 00 00 00 07 00 00 00 50 6C 61 79 65 72 73 01   ........Players.
        01 00 00 00 00 00 00 18 01                        .........
```

The first four bytes indicate that the Type ID is 0x15.

The next four byte indicate that the length of the Type Name is 0x07 bytes.

The next 0x06 bytes show that that the Type Name is "Players".

The next byte, 0x0, indicates that no additional data is present at the end of the block.

The next four bytes indicate that there is 0x1 instance.

The last four bytes represent a referent array (see section 5.2.16). Since there is only one camera instance, the untransformed referent value for this instance is 0x18. The transformed value (see section 3.3.2) is 0xC.

8.1.6 Workspace

```
00000110                                                     49                  I
00000120  4E 53 54 1D 00 00 00 1B 00 00 00 00 00 00 00 F0   NST............ð
00000130  0C 29 00 00 00 09 00 00 00 57 6F 72 6B 73 70 61   .).......Workspa
00000140  63 65 01 01 00 00 00 00 00 00 00 01               ce..........
```

```
Decompressed:
          29 00 00 00 09 00 00 00 57 6F 72 6B 73 70 61 63   ).......Workspac
          65 01 01 00 00 00 00 00 00 00 01                  e..........
```

The first four bytes indicate that the Type ID is 0x29.

The next four byte indicate that the length of the Type Name is 0x09 bytes.

The next 0x06 bytes show that that the Type Name is "Workspace".

The next byte, 0x0, indicates that no additional data is present at the end of the block.

The next four bytes indicate that there is 0x1 instance.

The last four bytes represent a referent array (see section 5.2.16). Since there is only one camera instance, the untransformed referent value for this instance is 0x0. The transformed value (see section 3.3.2) is 0x0.

8.2 REFERENTS

The following is a summary of the referent data from the previous header fields (with added Instance names, found in Property Data):

Referent	Instance Name	Type
0x0	Workspace	Workspace
0x2	BasePlate	Part
0x3	Camera	Camera
0x4	Sphere	Part
0x5	Cylinder	Part
0x6	Block	Part
0x7	Flat	Part
0x8	Decal	Decal
0xC	Players	Players
0x25	FilteredSelection	Instance
0x27	FilteredSelection	Instance
0x2C	FilteredSelection	Instance
0x2D	Lighting	Lighting

8.3 Property Data

8.3.1 Camera

```
00000140                                      50 52 4F 50              PROP
00000150  1C 00 00 00 1A 00 00 00 00 00 00 00 F0 0B 04 00  ............ð...
00000160  00 00 0D 00 00 00 43 61 6D 65 72 61 53 75 62 6A  ......CameraSubj
00000170  65 63 74 13 00 00 00 01 50 52 4F 50 19 00 00 00  ect.....PROP....
00000180  17 00 00 00 00 00 00 00 F0 08 04 00 00 00 0A 00  ........ð.......
00000190  00 00 43 61 6D 65 72 61 54 79 70 65 12 00 00 00  ..CameraType....
000001A0  00 50 52 4F 50 4B 00 00 00 49 00 00 00 00 00 00  .PROPK...I......
000001B0  00 F0 3A 04 00 00 00 0F 00 00 00 43 6F 6F 72 64  .ð:........Coord
000001C0  69 6E 61 74 65 46 72 61 6D 65 10 00 43 0E 69 3F  inateFrame..C.i?
000001D0  C4 7D A5 3E 81 46 84 BE 00 00 00 80 D3 D5 1F 3F  Ä}¥>.F„¾...€ÓÕ.?
000001E0  C2 F8 47 3F DA DB D3 3E 8D 0C 36 BF 85 82 11 3F  ÂøG?ÚÛÓ>..6¿…‚.?
000001F0  82 8F FC 3D 83 9E 81 E4 81 B4 AF 84 50 52 4F 50  ‚.ü=ƒž.ä.´¯„PROP
00000200  1A 00 00 00 18 00 00 00 00 00 00 00 F0 09 04 00  ............ð...
00000210  00 00 0B 00 00 00 46 69 65 6C 64 4F 66 56 69 65  ......FieldOfVie
00000220  77 04 85 18 00 00 50 52 4F 50 1D 00 00 00 1B 00  w.…...PROP......
00000230  00 00 00 00 00 00 F0 0C 04 00 00 00 05 00 00 00  ......ð.........
00000240  46 6F 63 75 73 10 02 82 7F 73 6D 83 85 82 CC 81  Focus..‚.smƒ…‚Ì.
00000250  6B EE 42 50 52 4F 50 18 00 00 00 17 00 00 00 00  kîBPROP.........
00000260  00 00 00 40 04 00 00 00 04 00 F0 00 4E 61 6D 65  ...@......ð.Name
00000270  01 06 00 00 00 43 61 6D 65 72 61                 .....Camera
```

Decompressed:

```
04 00 00 00 0D 00 00 00 43 61 6D 65 72 61 53 75  ........CameraSu
62 6A 65 63 74 13 00 00 00 01 50 52 4F 50 04 00  bject.....PROP..
00 00 0A 00 00 00 43 61 6D 65 72 61 54 79 70 65  ......CameraType
12 00 00 00 00 50 52 4F 50 04 00 00 00 0F 00 00  .....PROP.......
00 43 6F 6F 72 64 69 6E 61 74 65 46 72 61 6D 65  .CoordinateFrame
10 00 43 0E 69 3F C4 7D A5 3E 81 46 84 BE 00 00  ..C.i?Ä}¥>.F„¾..
00 80 D3 D5 1F 3F C2 F8 47 3F DA DB D3 3E 8D 0C  .€ÓÕ.?ÂøG?ÚÛÓ>..
36 BF 85 82 11 3F 82 8F FC 3D 83 9E 81 E4 81 B4  6¿…‚.?‚.ü=ƒž.ä.´
```

```
AF 84 50 52 4F 50 04 00 00 00 0B 00 00 00 46 69   ¯„PROP........Fi
65 6C 64 4F 66 56 69 65 77 04 85 18 00 00 50 52   eldOfView.…....PR
4F 50 04 00 00 00 05 00 00 00 46 6F 63 75 73 10   OP........Focus.
02 82 7F 73 6D 83 85 82 CC 81 6B EE 42 50 52 4F   .‚.smƒ…‚Ì.kîBPRO
50 04 00 00 00 04 00 00 00 4E 61 6D 65 01 06 00   P........Name...
00 00 43 61 6D 65 72 61                           ..Camera
```

8.3.1.1 CameraSubject

Decompressed:

```
04 00 00 00 0D 00 00 00 43 61 6D 65 72 61 53 75   ........CameraSu
62 6A 65 63 74 13 00 00 00 01                     bject.....
```

A data type value of **0x13** indicates that this property's data type is "**Referent**", meaning that it refers to another instance (see section 5.2.16).

The raw data is { **0x1** }. Referent values are subject to the integer transformation described in section 3.3.2. Thus, the true value is **-1**. This indicates no subject.

8.3.1.2 CameraType

Decompressed:

```
04 00 00 00 0A 00 00 00 43 61 6D 65 72 61 54 79   ........CameraTy
70 65 12 00 00 00 00                              pe.....
```

A data type value of **0x12** indicates that this property's data type is "**Enumeration**". The exact enumeration values are described on the Roblox wiki or in the Object Browser in Roblox Studio.

The raw data is { **0x0** }. This indicates[3] a CameraType of "**Fixed**".

8.3.1.3 CoordinateFrame

Decompressed:

```
04 00 00 00 0F 00 00 00 43 6F 6F 72 64 69 6E 61   ........Coordina
74 65 46 72 61 6D 65 10 00 43 0E 69 3F C4 7D A5   teFrame..C.i?Ä}¥
3E 81 46 84 BE 00 00 00 80 D3 D5 1F 3F C2 F8 47   >.F„¾...€ÓÕ.?ÂøG
```

[3] See Roblox wiki page on CameraTypes (http://wiki.roblox.com/index.php?title=CameraType_(Enum)).

```
3F DA DB D3 3E 8D 0C 36 BF 85 82 11 3F 82 8F FC   ?ÚÛÓ>..6¿…,.?‚.ü
3D 83 9E 81 E4 81 B4 AF 84                        =ƒž.ä.´¯„
```

A data type value of **0x10** indicates that this property's data type is "**CFrame**".

The raw data indicates that the position component is $(-12.49, 25.91, 6.82)$. The rotation matrix values are as follows:

$$\begin{bmatrix} .91037387 & .32322514 & -.2583504 \\ -0.0 & .62435645 & .7811395 \\ .4137867 & -.711129 & .5683978 \end{bmatrix}$$

8.3.1.4 FieldOfView

Decompressed:

```
04 00 00 00 0B 00 00 00 46 69 65 6C 64 4F 66 56   ........FieldOfV
69 65 77 04 85 18 00 00                           iew.…...
```

A data type value of **0x04** indicates that this property's data type is "**Float**" (see section 5.2.4).

The raw data is { **0x85180000** }. Float values are stored in a unique format (see section 3.3.3). The IEEE 754 representation is **0x428C0000**. This indicates a value of **70** (degrees).

8.3.1.5 Focus

Decompressed:

```
04 00 00 00 05 00 00 00 46 6F 63 75 73 10 02 82   ........Focus..‚
7F 73 6D 83 85 82 CC 81 6B EE 42                  .smƒ…‚Ì.kîB
```

A data type value of **0x10** indicates that this property's data type is "**CFrame**".

The raw data indicates that the position component is $(-11.98, 24.34, 5.69)$. The rotation matrix is as follows (identity matrix, indicating no rotation): $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

8.3.1.6 Name

Decompressed:

```
04 00 00 00 04 00 00 00 4E 61 6D 65 01 06 00 00   ........Name....
00 43 61 6D 65 72 61                              .Camera
```

A data type value of **0x01** indicates that this property's data type is "**String**" (see section 5.2.1). The length of the string is **0x06** bytes. The name of this instance is "**Camera**".

8.3.2 Decal

```
00000270                                      50 52 4F 50 13               PROP.
00000280 00 00 00 11 00 00 00 00 00 00 00 F0 02 0A 00 00  ...........ð....
00000290 00 04 00 00 00 46 61 63 65 12 00 00 00 01 50 52  .....Face.....PR
000002A0 4F 50 18 00 00 00 16 00 00 00 00 00 00 00 F0 07  OP............ð.
000002B0 0A 00 00 00 04 00 00 00 4E 61 6D 65 01 05 00 00  ........Name....
000002C0 00 44 65 63 61 6C 50 52 4F 50 14 00 00 00 12 00  .DecalPROP......
000002D0 00 00 00 00 00 00 F0 03 0A 00 00 00 05 00 00 00  ......ð.........
000002E0 53 68 69 6E 79 04 83 40 00 00 50 52 4F 50 17 00  Shiny.ƒ@..PROP..
000002F0 00 00 15 00 00 00 00 00 00 00 F0 06 0A 00 00 00  ..........ð.....
00000300 08 00 00 00 53 70 65 63 75 6C 61 72 04 00 00 00  ....Specular....
00000310 00 50 52 4F 50 38 00 00 00 39 00 00 00 00 00 00  .PROP8...9......
00000320 00 F3 10 0A 00 00 00 07 00 00 00 54 65 78 74 75  .ó.........Textu
00000330 72 65 01 25 00 00 00 72 62 78 61 73 73 65 74 3A  re.%...rbxasset:
00000340 2F 2F 17 00 F0 04 73 2F 53 70 61 77 6E 4C 6F 63  //..ð.s/SpawnLoc
00000350 61 74 69 6F 6E 2E 70 6E 67 50 52 4F 50 1B 00 00  ation.pngPROP...
00000360 00 19 00 00 00 00 00 00 00 F0 0A 0A 00 00 00 0C  .........ð......
00000370 00 00 00 54 72 61 6E 73 70 61 72 65 6E 63 79 04  ...Transparency.
00000380 00 00 00 00                                      ....
```

```
Decompressed:
         0A 00 00 00 04 00 00 00 46 61 63 65 12 00 00 00  ........Face....
         01 50 52 4F 50 0A 00 00 00 04 00 00 00 4E 61 6D  .PROP........Nam
         65 01 05 00 00 00 44 65 63 61 6C 50 52 4F 50 0A  e.....DecalPROP.
         00 00 00 05 00 00 00 53 68 69 6E 79 04 83 40 00  .......Shiny.ƒ@.
         00 50 52 4F 50 0A 00 00 00 08 00 00 00 53 70 65  .PROP........Spe
         63 75 6C 61 72 04 00 00 00 00 50 52 4F 50 0A 00  cular.....PROP..
         00 00 07 00 00 00 54 65 78 74 75 72 65 01 25 00  ......Texture.%.
         00 00 72 62 78 61 73 73 65 74 3A 2F 2F 54 65 78  ..rbxasset://Tex
         74 75 72 65 73 2F 53 70 61 77 6E 4C 6F 63 61 74  tures/SpawnLocat
         69 6F 6E 2E 70 6E 67 50 52 4F 50 0A 00 00 00 0C  ion.pngPROP.....
```

```
00 00 00 54 72 61 6E 73 70 61 72 65 6E 63 79 04  ...Transparency.
00 00 00 00                                      ....
```

8.3.2.1 Face

Decompressed:

```
0A 00 00 00 04 00 00 00 46 61 63 65 12 00 00 00  ........Face....
01                                               .
```

A data type value of **0x12** indicates that this property's data type is "**Enumeration**".

The raw data value is { **0x1** }. This indicates the face value is "**Bottom**".

8.3.2.2 Name

Decompressed:

```
0A 00 00 00 04 00 00 00 4E 61 6D 65 01 05 00 00  ........Name....
00 44 65 63 61 6C                                .Decal
```

A data type value of **0x1** indicates that this property's data type is "**String**".

This instance's name is "**Decal**".

8.3.2.3 Shiny

Decompressed:

```
0A 00 00 00 05 00 00 00 53 68 69 6E 79 04 83 40  ........Shiny.ƒ@
00 00                                            ..
```

A data type value of **0x4** indicates that this property's data type is "**Float**".

The raw data is { **0x83400000** }. Float values are stored in a unique format (see section 3.3.3). The IEEE 754 representation is **0x41A00000**. This indicates a value of **20**.

8.3.2.4 Specular

Decompressed:

```
0A 00 00 00 08 00 00 00 53 70 65 63 75 6C 61 72  ........Specular
04 00 00 00 00                                   .....
```

A data type value of **0x4** indicates that this property's data type is "**Float**".

The raw data is { **0x00000000** }. Float values are stored in a unique format (see section 3.3.3). The IEEE 754 representation is **0x00000000**. This indicates a value of **0**.

8.3.2.5 Texture

```
Decompressed:
                0A 00 00 00 07 00 00 00 54 65 78 74 75 72 65 01  ........Texture.
                25 00 00 00 72 62 78 61 73 73 65 74 3A 2F 2F 54  %...rbxasset://T
                65 78 74 75 72 65 73 2F 53 70 61 77 6E 4C 6F 63  extures/SpawnLoc
                61 74 69 6F 6E 2E 70 6E 67                       ation.png
```

A data type value of **0x1** indicates that this property's data type is "**String**".

This texture URI is "**rbxasset://Textures/SpawnLocation.png**".

8.3.2.6 Transparency

```
Decompressed:
                0A 00 00 00 0C 00 00 00 54 72 61 6E 73 70 61 72  ........Transpar
                65 6E 63 79 04 00 00 00 00                       ency.....
```

A data type value of **0x4** indicates that this property's data type is "**Float**".

The raw data is { **0x00000000** }. Float values are stored in a unique format (see section 3.3.3). The IEEE 754 representation is **0x00000000**. This indicates a value of **0**.

8.3.3 Instance

```
00000380                50 52 4F 50 2D 00 00 00 4C 00 00 00      PROP-...L...
00000390    00 00 00 00 FF 13 0D 00 00 00 04 00 00 00 4E 61  ....ÿ.........Na
000003A0    6D 65 01 11 00 00 00 46 69 6C 74 65 72 65 64 53  me.....FilteredS
000003B0    65 6C 65 63 74 69 6F 6E 15 00 12 50 63 74 69 6F  election...Pctio
000003C0    6E                                               n
```

```
Decompressed:
                0D 00 00 00 04 00 00 00 4E 61 6D 65 01 11 00 00  ........Name....
                00 46 69 6C 74 65 72 65 64 53 65 6C 65 63 74 69  .FilteredSelecti
                6F 6E 11 00 00 00 46 69 6C 74 65 72 65 64 53 65  on....FilteredSe
```

```
            6C 65 63 74 69 6F 6E 11 00 00 00 46 69 6C 74 65  lection....Filte
            72 65 64 53 65 6C 65 63 74 69 6F 6E              redSelection
```

8.3.3.1 Name

```
Decompressed:
            0D 00 00 00 04 00 00 00 4E 61 6D 65 01 11 00 00  ........Name....
            00 46 69 6C 74 65 72 65 64 53 65 6C 65 63 74 69  .FilteredSelecti
            6F 6E 11 00 00 00 46 69 6C 74 65 72 65 64 53 65  on....FilteredSe
            6C 65 63 74 69 6F 6E 11 00 00 00 46 69 6C 74 65  lection....Filte
            72 65 64 53 65 6C 65 63 74 69 6F 6E              redSelection
```

A data type value of **0x1** indicates that the data type of this property is "**String**".

The data array is { "**FilteredSelection**", "**FilteredSelection**", "**FilteredSelection**" }. Thus, each of the three instances have the name "FilteredSelection".

8.3.4 Lighting

```
000003C0     50 52 4F 50 1E 00 00 00 1C 00 00 00 00 00 00   PROP...........
000003D0  00 F0 0D 0F 00 00 00 07 00 00 00 41 6D 62 69 65  .ð.........Ambie
000003E0  6E 74 0C 00 00 00 00 00 00 00 00 00 00 00 00 50  nt.............P
000003F0  52 4F 50 19 00 00 00 17 00 00 00 00 00 00 00 F0  ROP............ð
00000400  08 0F 00 00 00 0A 00 00 00 42 72 69 67 68 74 6E  .........Brightn
00000410  65 73 73 04 7F 00 00 00 50 52 4F 50 28 00 00 00  ess.....PROP(...
00000420  26 00 00 00 00 00 00 00 F0 17 0F 00 00 00 11 00  &.......ð.......
00000430  00 00 43 6F 6C 6F 72 53 68 69 66 74 5F 42 6F 74  ..ColorShift_Bot
00000440  74 6F 6D 0C 00 00 00 00 00 00 00 00 00 00 00 00  tom.............
00000450  50 52 4F 50 25 00 00 00 23 00 00 00 00 00 00 00  PROP%...#.......
00000460  F0 14 0F 00 00 00 0E 00 00 00 43 6F 6C 6F 72 53  ð.........ColorS
00000470  68 69 66 74 5F 54 6F 70 0C 00 00 00 00 00 00 00  hift_Top........
00000480  00 00 00 00 00 50 52 4F 50 1F 00 00 00 1D 00 00  .....PROP.......
00000490  00 00 00 00 00 F0 0E 0F 00 00 00 08 00 00 00 46  .....ð.........F
000004A0  6F 67 43 6F 6C 6F 72 0C 7E 80 00 00 7E 80 00 00  ogColor.~€..~€..
000004B0  7E 80 00 00 50 52 4F 50 15 00 00 00 13 00 00 00  ~€..PROP........
000004C0  00 00 00 00 F0 04 0F 00 00 00 06 00 00 00 46 6F  ....ð.........Fo
000004D0  67 45 6E 64 04 8F 86 A0 00 50 52 4F 50 17 00 00  gEnd..† .PROP...
000004E0  00 15 00 00 00 00 00 00 F0 06 0F 00 00 00 08     .........ð......
```

```
000004F0  00 00 00 46 6F 67 53 74 61 72 74 04 00 00 00 00  ...FogStart.....
00000500  50 52 4F 50 21 00 00 00 1F 00 00 00 00 00 00 00  PROP!...........
00000510  F0 10 0F 00 00 00 12 00 00 00 47 65 6F 67 72 61  ð.........Geogra
00000520  70 68 69 63 4C 61 74 69 74 75 64 65 04 84 4D DD  phicLatitude.„MÝ
00000530  CC 50 52 4F 50 19 00 00 00 17 00 00 00 00 00 00  ÌPROP...........
00000540  00 F0 08 0F 00 00 00 0D 00 00 00 47 6C 6F 62 61  .ð.........Globa
00000550  6C 53 68 61 64 6F 77 73 02 01 50 52 4F 50 1B 00  lShadows..PROP..
00000560  00 00 19 00 00 00 00 00 00 00 F0 0A 0F 00 00 00  ..........ð.....
00000570  04 00 00 00 4E 61 6D 65 01 08 00 00 00 4C 69 67  ....Name.....Lig
00000580  68 74 69 6E 67 50 52 4F 50 25 00 00 00 23 00 00  htingPROP%...#..
00000590  00 00 00 00 00 F0 14 0F 00 00 00 0E 00 00 00 4F  .....ð.........O
000005A0  75 74 64 6F 6F 72 41 6D 62 69 65 6E 74 0C 7E 0F  utdoorAmbient.~.
000005B0  0F 10 7E 0F 0F 10 7E 01 01 02 50 52 4F 50 14 00  ..~...~...PROP..
000005C0  00 00 12 00 00 00 00 00 00 00 F0 03 0F 00 00 00  ..........ð.....
000005D0  08 00 00 00 4F 75 74 6C 69 6E 65 73 02 01 50 52  ....Outlines..PR
000005E0  4F 50 22 00 00 00 20 00 00 00 00 00 00 00 F0 11  OP"... .......ð.
000005F0  0F 00 00 00 0B 00 00 00 53 68 61 64 6F 77 43 6F  ........ShadowCo
00000600  6C 6F 72 0C 7E 66 66 66 7E 66 66 66 7E 70 A3 D8  lor.~fff~fff~p£Ø
00000610  50 52 4F 50 20 00 00 00 1E 00 00 00 00 00 00 00  PROP ...........
00000620  F0 0F 0F 00 00 00 09 00 00 00 54 69 6D 65 4F 66  ð.........TimeOf
00000630  44 61 79 01 08 00 00 00 31 35 3A 30 30 3A 30 30  Day.....15:00:00
```

Decompressed:

```
          0F 00 00 00 07 00 00 00 41 6D 62 69 65 6E 74 0C  ........Ambient.
          00 00 00 00 00 00 00 00 00 00 00 00 50 52 4F 50  ............PROP
          0F 00 00 00 0A 00 00 00 42 72 69 67 68 74 6E 65  ........Brightne
          73 73 04 7F 00 00 00 50 52 4F 50 0F 00 00 00 11  ss.....PROP.....
          00 00 00 43 6F 6C 6F 72 53 68 69 66 74 5F 42 6F  ...ColorShift_Bo
          74 74 6F 6D 0C 00 00 00 00 00 00 00 00 00 00 00  ttom............
          00 50 52 4F 50 0F 00 00 00 0E 00 00 00 43 6F 6C  .PROP........Col
          6F 72 53 68 69 66 74 5F 54 6F 70 0C 00 00 00 00  orShift_Top.....
          00 00 00 00 00 00 00 00 50 52 4F 50 0F 00 00 00  ........PROP....
          08 00 00 00 46 6F 67 43 6F 6C 6F 72 0C 7E 80 00  ....FogColor.~€.
          00 7E 80 00 00 7E 80 00 00 50 52 4F 50 0F 00 00  .~€..~€..PROP...
```

```
00 06 00 00 00 46 6F 67 45 6E 64 04 8F 86 A0 00   .....FogEnd..† .
50 52 4F 50 0F 00 00 00 08 00 00 00 46 6F 67 53   PROP........FogS
74 61 72 74 04 00 00 00 00 50 52 4F 50 0F 00 00   tart.....PROP...
00 12 00 00 00 47 65 6F 67 72 61 70 68 69 63 4C   .....GeographicL
61 74 69 74 75 64 65 04 84 4D DD CC 50 52 4F 50   atitude.„MÝÌPROP
0F 00 00 00 0D 00 00 00 47 6C 6F 62 61 6C 53 68   ........GlobalSh
61 64 6F 77 73 02 01 50 52 4F 50 0F 00 00 00 04   adows..PROP.....
00 00 00 4E 61 6D 65 01 08 00 00 00 4C 69 67 68   ...Name.....Ligh
74 69 6E 67 50 52 4F 50 0F 00 00 00 0E 00 00 00   tingPROP........
4F 75 74 64 6F 6F 72 41 6D 62 69 65 6E 74 0C 7E   OutdoorAmbient.~
0F 0F 10 7E 0F 0F 10 7E 01 01 02 50 52 4F 50 0F   ...~...~...PROP.
00 00 00 08 00 00 00 4F 75 74 6C 69 6E 65 73 02   .......Outlines.
01 50 52 4F 50 0F 00 00 00 0B 00 00 00 53 68 61   .PROP........Sha
64 6F 77 43 6F 6C 6F 72 0C 7E 66 66 66 7E 66 66   dowColor.~fff~ff
66 7E 70 A3 D8 50 52 4F 50 0F 00 00 00 09 00 00   f~p£ØPROP.......
00 54 69 6D 65 4F 66 44 61 79 01 08 00 00 00 31   .TimeOfDay.....1
35 3A 30 30 3A 30 30                              5:00:00
```

8.3.4.1 Ambient

Decompressed:

```
0F 00 00 00 07 00 00 00 41 6D 62 69 65 6E 74 0C   ........Ambient.
00 00 00 00 00 00 00 00 00 00 00 00               ............
```

A data type value of **0xC** indicates that this property's data type is "**Color3**".

The value of this property is **(R: 0, G: 0, B: 0)**.

8.3.4.2 Brightness

Decompressed:

```
0F 00 00 00 0A 00 00 00 42 72 69 67 68 74 6E 65   ........Brightne
73 73 04 7F 00 00 00                              ss.....
```

A data type value of **0x4** indicates that this property's data type is **"Float"**.

The raw data is { **0x7F000000** }. Float values are stored in a unique format (see section 3.3.3). The IEEE 754 representation is { **0x3F800000** }. This indicates a value of **1.0**.

8.3.4.3 ColorShift_Bottom

```
Decompressed:
          0F 00 00 00 11 00 00 00 43 6F 6C 6F 72 53 68 69  ........ColorShi
          66 74 5F 42 6F 74 74 6F 6D 0C 00 00 00 00 00 00  ft_Bottom.......
          00 00 00 00 00 00                                ......
```

A data type value of 0xC indicates that this property's data type is "Color3".

The value of this property is **(R: 0, G: 0, B: 0)**.

8.3.4.4 ColorShift_Top

```
Decompressed:
          0F 00 00 00 0E 00 00 00 43 6F 6C 6F 72 53 68 69  ........ColorShi
          66 74 5F 54 6F 70 0C 00 00 00 00 00 00 00 00 00  ft_Top..........
          00 00 00                                         ...
```

A data type value of **0xC** indicates that this property's data type is "**Color3**".

The value of this property is **(R: 0, G: 0, B: 0)**.

8.3.4.5 FogColor

```
Decompressed:
          0F 00 00 00 08 00 00 00 46 6F 67 43 6F 6C 6F 72  ........FogColor
          0C 7E 80 00 00 7E 80 00 00 7E 80 00 00           .~€..~€..~€..
```

A data type value of **0xC** indicates that this property's data type is "**Color3**".

The value of this property is **(R: .75f, G: .75f, B: .75f)**, or when the floats are converted to bytes, **(R: 191, 191, 191)**.

FogEnd

```
Decompressed:
          0F 00 00 00 06 00 00 00 46 6F 67 45 6E 64 04 8F  ........FogEnd..
          86 A0 00                                         † .
```

A data type value of **0x4** indicates that this property's data type is "**Float**".

The raw data is { **0x8F86A000** }. Float values are stored in a unique format (see section 3.3.3). The IEEE 754 representation is { **0x47C35000** }. This indicates a value of **10000**.

8.3.4.6 FogStart

Decompressed:

```
0F 00 00 00 08 00 00 00 46 6F 67 53 74 61 72 74  ........FogStart
04 00 00 00 00                                   .....
```

A data type value of **0x4** indicates that this property's data type is "**Float**".

The raw data is { **0x00000000** }. Float values are stored in a unique format (see section 3.3.3). The IEEE 754 representation is { **0x00000000** }. This indicates a value of **0**.

8.3.4.7 GeographicLatitude

Decompressed:

```
0F 00 00 00 12 00 00 00 47 65 6F 67 72 61 70 68  ........Geograph
69 63 4C 61 74 69 74 75 64 65 04 84 4D DD CC     icLatitude.„MÝÌ
```

A data type value of **0x4** indicates that this property's data type is "**Float**".

The raw data is { **0x844DDDCC** }. Float values are stored in a unique format (see section 3.3.3). The IEEE 754 representation is { **0x4226EEE6** }. This indicates a value of **41.733**.

8.3.4.8 GlobalShadows

Decompressed:

```
0F 00 00 00 0D 00 00 00 47 6C 6F 62 61 6C 53 68  ........GlobalSh
61 64 6F 77 73 02 01                             adows..
```

A data type value of **0x2** indicates that this property's data type is "**Boolean**".

The value of this property is **True**.

8.3.4.9 Name

Decompressed:

```
0F 00 00 00 04 00 00 00 4E 61 6D 65 01 08 00 00  ........Name....
```

```
00 4C 69 67 68 74 69 6E 67                            .Lighting
```

A data type value of **0x1** indicates that this property's data type is "**String**".

The name of this instance is "**Lighting**".

8.3.4.10 OutdoorAmbient

Decompressed:

```
0F 00 00 00 0E 00 00 00 4F 75 74 64 6F 6F 72 41  ........OutdoorA
6D 62 69 65 6E 74 0C 7E 0F 0F 10 7E 0F 0F 10 7E  mbient.~...~...~
01 01 02                                         ...
```

A data type value of **0xC** indicates that this property's data type is "**Color3**".

The value of this property is **(R: .5294f, G: .5294f, B: .5020f)**, or when the floats are converted to bytes, **(R: 135, 135, 128).**

8.3.4.11 Outlines

Decompressed:

```
0F 00 00 00 08 00 00 00 4F 75 74 6C 69 6E 65 73  ........Outlines
02 01                                            ..
```

A data type value of **0x2** indicates that this property's data type is "**Boolean**".

The value of this property is **True**.

8.3.4.12 ShadowColor

Decompressed:

```
0F 00 00 00 0B 00 00 00 53 68 61 64 6F 77 43 6F  ........ShadowCo
6C 6F 72 0C 7E 66 66 66 7E 66 66 66 7E 70 A3 D8  lor.~fff~fff~p£Ø
```

A data type value of **0xC** indicates that this property's data type is "**Color3**".

The value of this property is **(R: .7f, G: .7f, B: .72f)**, or when the floats are converted to bytes, **(R: 178, 178, 183).**

8.3.4.13 TimeOfDay

Decompressed:

```
0F 00 00 00 09 00 00 00 54 69 6D 65 4F 66 44 61   ........TimeOfDa
79 01 08 00 00 00 31 35 3A 30 30 3A 30 30         y.....15:00:00
```

A data type value of **0x1** indicates that this property's data type is "**String**".

The value of this property is "**15:00:00**".

8.3.5 Part

```
00000640  50 52 4F 50 18 00 00 00 16 00 00 00 00 00 00 00   PROP............
00000650  F0 07 12 00 00 00 08 00 00 00 41 6E 63 68 6F 72   ð.........Anchor
00000660  65 64 02 01 00 00 00 00 50 52 4F 50 22 00 00 00   ed......PROP"...
00000670  27 00 00 00 00 00 00 00 F0 05 12 00 00 00 0A 00   '.......ð.......
00000680  00 00 42 61 63 6B 50 61 72 61 6D 41 04 7E 01 00   ..BackParamA.~..
00000690  15 00 01 00 50 01 01 01 01 01 50 52 4F 50 22 00   ....P.....PROP".
000006A0  00 00 27 00 00 00 00 00 00 00 F0 05 12 00 00 00   ..'.......ð.....
000006B0  0A 00 00 00 42 61 63 6B 50 61 72 61 6D 42 04 7E   ....BackParamB.~
000006C0  01 00 15 00 01 00 50 00 00 00 00 00 50 52 4F 50   ......P.....PROP
000006D0  20 00 00 00 28 00 00 00 00 00 00 00 F0 04 12 00    ...(.......ð...
000006E0  00 00 0B 00 00 00 42 61 63 6B 53 75 72 66 61 63   ......BackSurfac
000006F0  65 13 00 08 02 00 50 00 00 00 05 00 50 52 4F 50   e.....P.....PROP
00000700  25 00 00 00 2D 00 00 00 00 00 00 00 F0 09 12 00   %...-.......ð...
00000710  00 00 10 00 00 00 42 61 63 6B 53 75 72 66 61 63   ......BackSurfac
00000720  65 49 6E 70 75 74 18 00 08 02 00 50 00 00 00 00   eInput.....P....
00000730  00 50 52 4F 50 24 00 00 00 29 00 00 00 00 00 00   .PROP$...)......
00000740  00 F0 07 12 00 00 00 0C 00 00 00 42 6F 74 74 6F   .ð.........Botto
00000750  6D 50 61 72 61 6D 41 04 7E 01 00 15 00 01 00 50   mParamA.~......P
00000760  01 01 01 01 01 50 52 4F 50 24 00 00 00 29 00 00   .....PROP$...)..
00000770  00 00 00 00 00 F0 07 12 00 00 00 0C 00 00 00 42   .....ð.........B
00000780  6F 74 74 6F 6D 50 61 72 61 6D 42 04 7E 01 00 15   ottomParamB.~...
00000790  00 01 00 50 00 00 00 00 00 50 52 4F 50 22 00 00   ...P.....PROP"..
000007A0  00 2A 00 00 00 00 00 00 00 F0 06 12 00 00 00 0D   .*.......ð......
000007B0  00 00 00 42 6F 74 74 6F 6D 53 75 72 66 61 63 65   ...BottomSurface
000007C0  15 00 08 02 00 50 04 00 00 05 02 50 52 4F 50 26   .....P.....PROP&
000007D0  00 00 00 2F 00 00 00 00 00 00 00 40 12 00 00 00   .../.......@....
000007E0  04 00 F0 03 42 6F 74 74 6F 6D 53 75 72 66 61 63   ..ð.BottomSurfac
```

```
000007F0  65 49 6E 70 75 74 16 00 08 02 00 50 00 00 00 00  eInput.....P....
00000800  00 50 52 4F 50 20 00 00 00 27 00 00 00 00 00 00  .PROP ...'......
00000810  00 F8 05 12 00 00 00 0A 00 00 00 42 72 69 63 6B  .ø.........Brick
00000820  43 6F 6C 6F 72 0B 00 01 00 70 03 00 C7 C7 18 EE  Color....p..ÇÇ.î
00000830  01 50 52 4F 50 68 00 00 00 74 00 00 00 00 00 00  .PROPh...t......
00000840  00 F6 03 12 00 00 00 06 00 00 00 43 46 72 61 6D  .ö.........CFram
00000850  65 10 02 02 00 01 00 66 80 3F 00 00 80 BF 10 00  e......f€?..€¿..
00000860  00 02 00 F0 35 80 BF 2E BD 3B B3 02 02 00 80 83  ...ð5€¿.½;³...€ƒ
00000870  83 7F 00 00 0C 0C 66 00 00 CC CC 66 00 00 D1 C7  ƒ.....f..ÌÌf..ÑÇ
00000880  00 7E 81 81 80 7C 38 72 0C 0B 85 51 8F 28 85 1E  .~..€|8r..…Q.(….
00000890  ED 5C F4 20 BA 00 83 83 81 81 00 90 A0 CC CC 00  í\ô º.ƒƒ.... ÌÌ.
000008A0  00 00 CC CD 00 01 01 B5 01 50 52 4F 50 1A 00 00  ..ÌÍ...µ.PROP...
000008B0  00 18 00 00 00 00 00 00 00 F0 09 12 00 00 00 0A  .........ð......
000008C0  00 00 00 43 61 6E 43 6F 6C 6C 69 64 65 02 01 01  ...CanCollide...
000008D0  01 01 01 50 52 4F 50 22 00 00 00 27 00 00 00 00  ...PROP"...'....
000008E0  00 00 00 F0 05 12 00 00 00 0A 00 00 00 45 6C 61  ...ð.........Ela
000008F0  73 74 69 63 69 74 79 04 7E 01 00 15 00 01 00 50  sticity.~......P
00000900  00 00 00 00 00 50 52 4F 50 20 00 00 00 25 00 00  .....PROP ...%..
00000910  00 00 00 00 00 F0 03 12 00 00 00 08 00 00 00 46  .....ð.........F
00000920  72 69 63 74 69 6F 6E 04 7D 01 00 15 33 01 00 50  riction.}...3..P
00000930  34 34 34 34 34 50 52 4F 50 23 00 00 00 28 00 00  44444PROP#...(..
00000940  00 00 00 00 00 F0 06 12 00 00 00 0B 00 00 00 46  .....ð.........F
00000950  72 6F 6E 74 50 61 72 61 6D 41 04 7E 01 00 15 00  rontParamA.~....
00000960  01 00 50 01 01 01 01 01 50 52 4F 50 23 00 00 00  ..P.....PROP#...
00000970  28 00 00 00 00 00 00 00 F0 06 12 00 00 00 0B 00  (.......ð.......
00000980  00 00 46 72 6F 6E 74 50 61 72 61 6D 42 04 7E 01  ..FrontParamB.~.
00000990  00 15 00 01 00 50 00 00 00 00 00 50 52 4F 50 21  .....P.....PROP!
000009A0  00 00 00 29 00 00 00 00 00 00 00 F0 05 12 00 00  ...).......ð....
000009B0  00 0C 00 00 00 46 72 6F 6E 74 53 75 72 66 61 63  .....FrontSurfac
000009C0  65 14 00 08 02 00 50 00 00 00 05 00 50 52 4F 50  e.....P.....PROP
000009D0  26 00 00 00 2E 00 00 00 00 00 00 F0 0A 12 00     &..........ð...
000009E0  00 00 11 00 00 00 46 72 6F 6E 74 53 75 72 66 61  ......FrontSurfa
000009F0  63 65 49 6E 70 75 74 19 00 08 02 00 50 00 00 00  ceInput.....P...
00000A00  00 00 50 52 4F 50 22 00 00 00 27 00 00 00 00 00  ..PROP"...'.....
```

```
00000A10  00 00 F0 05 12 00 00 00 0A 00 00 00 4C 65 66 74  ..ð.........Left
00000A20  50 61 72 61 6D 41 04 7E 01 00 15 00 01 00 50 01  ParamA.~......P.
00000A30  01 01 01 01 50 52 4F 50 22 00 00 00 27 00 00 00  ....PROP"...'...
00000A40  00 00 00 00 F0 05 12 00 00 00 0A 00 00 00 4C 65  ....ð.........Le
00000A50  66 74 50 61 72 61 6D 42 04 7E 01 00 15 00 01 00  ftParamB.~......
00000A60  50 00 00 00 00 00 50 52 4F 50 20 00 00 00 28 00  P.....PROP ...(.
00000A70  00 00 00 00 00 00 F0 04 12 00 00 00 0B 00 00 00  ......ð.........
00000A80  4C 65 66 74 53 75 72 66 61 63 65 13 00 08 02 00  LeftSurface.....
00000A90  50 00 00 00 05 00 50 52 4F 50 25 00 00 00 2D 00  P.....PROP%...-.
00000AA0  00 00 00 00 00 00 F0 09 12 00 00 00 10 00 00 00  ......ð.........
00000AB0  4C 65 66 74 53 75 72 66 61 63 65 49 6E 70 75 74  LeftSurfaceInput
00000AC0  18 00 08 02 00 50 00 00 00 00 00 50 52 4F 50 16  .....P.....PROP.
00000AD0  00 00 00 14 00 00 00 00 00 00 00 F0 05 12 00 00  ...........ð....
00000AE0  00 06 00 00 00 4C 6F 63 6B 65 64 02 01 00 00 00  .....Locked.....
00000AF0  00 50 52 4F 50 22 00 00 00 25 00 00 00 00 00 00  .PROP"...%......
00000B00  00 F0 01 12 00 00 00 08 00 00 00 4D 61 74 65 72  .ð.........Mater
00000B10  69 61 6C 10 00 03 02 00 A0 01 03 01 01 01 00 20  ial..... ...... 
00000B20  00 00 00 50 52 4F 50 42 00 00 00 41 00 00 00 00  ...PROPB...A....
00000B30  00 00 00 F0 22 12 00 00 00 04 00 00 00 4E 61 6D  ...ð".......Nam
00000B40  65 01 09 00 00 00 42 61 73 65 50 6C 61 74 65 06  e.....BasePlate.
00000B50  00 00 00 53 70 68 65 72 65 08 00 00 00 43 79 6C  ...Sphere....Cyl
00000B60  69 6E 64 65 72 05 23 00 C0 6C 6F 63 6B 04 00 00  inder.#.Àlock...
00000B70  00 46 6C 61 74 50 52 4F 50 24 00 00 00 28 00 00  .FlatPROP$...(..
00000B80  00 00 00 00 00 F5 0B 12 00 00 00 0B 00 00 00 52  .....õ.........R
00000B90  65 66 6C 65 63 74 61 6E 63 65 04 00 00 7E 00 7E  eflectance...~.~
00000BA0  00 01 00 50 00 00 00 00 00 50 52 4F 50 23 00 00  ...P.....PROP#..
00000BB0  00 28 00 00 00 00 00 00 00 F0 06 12 00 00 00 0B  .(.......ð......
00000BC0  00 00 00 52 69 67 68 74 50 61 72 61 6D 41 04 7E  ...RightParamA.~
00000BD0  01 00 15 00 01 00 50 01 01 01 01 01 50 52 4F 50  ......P.....PROP
00000BE0  23 00 00 00 28 00 00 00 00 00 00 00 F0 06 12 00  #...(.......ð...
00000BF0  00 00 0B 00 00 00 52 69 67 68 74 50 61 72 61 6D  ......RightParam
00000C00  42 04 7E 01 00 15 00 01 00 50 00 00 00 00 00 50  B.~......P.....P
00000C10  52 4F 50 21 00 00 00 29 00 00 00 00 00 00 00 F0  ROP!...).......ð
00000C20  05 12 00 00 00 0C 00 00 00 52 69 67 68 74 53 75  .........RightSu
```

```
00000C30  72 66 61 63 65 14 00 08 02 00 50 00 00 00 05 00  rface.....P.....
00000C40  50 52 4F 50 26 00 00 00 2E 00 00 00 00 00 00 00  PROP&...........
00000C50  F0 0A 12 00 00 00 11 00 00 00 52 69 67 68 74 53  ð.........RightS
00000C60  75 72 66 61 63 65 49 6E 70 75 74 19 00 08 02 00  urfaceInput.....
00000C70  50 00 00 00 00 00 50 52 4F 50 20 00 00 00 50 00  P.....PROP ...P.
00000C80  00 00 00 00 00 00 FF 06 12 00 00 00 0B 00 00 00  ......ÿ.........
00000C90  52 6F 74 56 65 6C 6F 63 69 74 79 0E 00 01 00 23  RotVelocity....#
00000CA0  50 00 00 00 00 00 50 52 4F 50 21 00 00 00 26 00  P.....PROP!...&.
00000CB0  00 00 00 00 00 00 F0 04 12 00 00 00 09 00 00 00  ......ð.........
00000CC0  54 6F 70 50 61 72 61 6D 41 04 7E 01 00 15 00 01  TopParamA.~.....
00000CD0  00 50 01 01 01 01 01 50 52 4F 50 21 00 00 00 26  .P.....PROP!...&
00000CE0  00 00 00 00 00 00 00 F0 04 12 00 00 00 09 00 00  .......ð........
00000CF0  00 54 6F 70 50 61 72 61 6D 42 04 7E 01 00 15 00  .TopParamB.~....
00000D00  01 00 50 00 00 00 00 00 50 52 4F 50 1F 00 00 00  ..P.....PROP....
00000D10  27 00 00 00 00 00 00 00 F0 03 12 00 00 00 0A 00  '.......ð.......
00000D20  00 00 54 6F 70 53 75 72 66 61 63 65 12 00 08 02  ..TopSurface....
00000D30  00 50 03 00 00 05 02 50 52 4F 50 24 00 00 00 2C  .P.....PROP$...,
00000D40  00 00 00 00 00 00 00 F0 08 12 00 00 00 0F 00 00  .......ð........
00000D50  00 54 6F 70 53 75 72 66 61 63 65 49 6E 70 75 74  .TopSurfaceInput
00000D60  17 00 08 02 00 50 00 00 00 00 00 50 52 4F 50 20  .....P.....PROP
00000D70  00 00 00 29 00 00 00 00 00 00 00 FA 07 12 00 00  ...).......ú....
00000D80  00 0C 00 00 00 54 72 61 6E 73 70 61 72 65 6E 63  .....Transparenc
00000D90  79 04 00 01 00 50 00 00 00 00 00 50 52 4F 50 1D  y....P.....PROP.
00000DA0  00 00 00 4D 00 00 00 00 00 00 00 FF 03 12 00 00  ...M.......ÿ....
00000DB0  00 08 00 00 00 56 65 6C 6F 63 69 74 79 0E 00 01  .....Velocity...
00000DC0  00 23 50 00 00 00 00 00 50 52 4F 50 22 00 00 00  .#P.....PROP"...
00000DD0  2A 00 00 00 00 00 00 00 F0 06 12 00 00 00 0D 00  *.......ð.......
00000DE0  00 00 66 6F 72 6D 46 61 63 74 6F 72 52 61 77 15  ..formFactorRaw.
00000DF0  00 08 02 00 50 01 00 00 01 02 50 52 4F 50 19 00  ....P.....PROP..
00000E00  00 00 22 00 00 00 00 00 00 00 D0 12 00 00 00 05  .."........Ð.....
00000E10  00 00 00 73 68 61 70 65 0D 00 08 02 00 50 01 00  ...shape.....P..
00000E20  02 01 01 50 52 4F 50 46 00 00 00 49 00 00 00 00  ...PROPF...I....
00000E30  00 00 00 F4 26 12 00 00 00 04 00 00 00 73 69 7A  ...ô&........siz
00000E40  65 0E 88 82 82 81 82 00 73 0C 33 00 00 33 CC 33  e.ˆ‚‚.‚.s.3..3Ì3
```

```
00000E50  00 00 34 CC 34 00 7F 82 82 81 7D 33 73 0C 0C 99  ..4Ì4..‚‚.}3s..™
00000E60  33 33 CC CC 99 34 34 CC CE 9A 28 00 C0 66 00 00  33ÌÌ™44ÌÎš(.Àf..
00000E70  33 CC 66 00 00 34 CC 68                          3Ìf..4Ìh.
```

Decompressed:

```
12 00 00 00 08 00 00 00 41 6E 63 68 6F 72 65 64  ........Anchored
02 01 00 00 00 00 50 52 4F 50 12 00 00 00 0A 00  ......PROP......
00 00 42 61 63 6B 50 61 72 61 6D 41 04 7E 7E 7E  ..BackParamA.~~~
7E 7E 00 00 00 00 00 00 00 00 00 00 01 01 01 01  ~~..............
01 50 52 4F 50 12 00 00 00 0A 00 00 00 42 61 63  .PROP........Bac
6B 50 61 72 61 6D 42 04 7E 7E 7E 7E 7E 00 00 00  kParamB.~~~~~...
00 00 00 00 00 00 00 00 00 00 00 00 50 52 4F 50  ............PROP
12 00 00 00 0B 00 00 00 42 61 63 6B 53 75 72 66  ........BackSurf
61 63 65 12 00 00 00 00 00 00 00 00 00 00 00 00  ace.............
00 00 00 00 00 00 05 00 50 52 4F 50 12 00 00 00  ........PROP....
10 00 00 00 42 61 63 6B 53 75 72 66 61 63 65 49  ....BackSurfaceI
6E 70 75 74 12 00 00 00 00 00 00 00 00 00 00 00  nput............
00 00 00 00 00 00 00 00 00 50 52 4F 50 12 00 00  .........PROP...
00 0C 00 00 00 42 6F 74 74 6F 6D 50 61 72 61 6D  .....BottomParam
41 04 7E 7E 7E 7E 7E 00 00 00 00 00 00 00 00 00  A.~~~~~.........
00 01 01 01 01 01 50 52 4F 50 12 00 00 00 0C 00  ......PROP......
00 00 42 6F 74 74 6F 6D 50 61 72 61 6D 42 04 7E  ..BottomParamB.~
7E 7E 7E 7E 00 00 00 00 00 00 00 00 00 00 00 00  ~~~~............
00 00 00 50 52 4F 50 12 00 00 00 0D 00 00 00 42  ...PROP........B
6F 74 74 6F 6D 53 75 72 66 61 63 65 12 00 00 00  ottomSurface....
00 00 00 00 00 00 00 00 00 00 00 00 04 00 00 05  ................
02 50 52 4F 50 12 00 00 00 12 00 00 00 42 6F 74  .PROP........Bot
74 6F 6D 53 75 72 66 61 63 65 49 6E 70 75 74 12  tomSurfaceInput.
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 50 52 4F 50 12 00 00 00 0A 00 00 00  ....PROP........
42 72 69 63 6B 43 6F 6C 6F 72 0B 00 00 00 00 00  BrickColor......
00 00 00 00 00 00 00 00 03 00 C7 C7 18 EE 01 50  ..........ÇÇ.î.P
52 4F 50 12 00 00 00 06 00 00 00 43 46 72 61 6D  ROP........CFram
65 10 02 02 00 00 00 00 00 00 00 00 00 00 00 80  e..............€
```

```
3F 00 00 80 BF 00 00 00 00 00 00 00 00 00 00 00  ?..€¿...........
00 00 00 80 BF 2E BD 3B B3 02 02 00 80 83 83 7F  ...€¿.½;³...€ƒƒ.
00 00 0C 0C 66 00 00 CC CC 66 00 00 D1 C7 00 7E  ....f..ÌÌf..ÑÇ.~
81 81 80 7C 38 72 0C 0B 85 51 8F 28 85 1E ED 5C  ..€|8r..…Q.(….í\
F4 20 BA 00 83 83 81 81 00 90 A0 CC CC 00 00 00  ô º.ƒƒ.... ÌÌ...
CC CD 00 01 01 B5 01 50 52 4F 50 12 00 00 00 0A  ÌÍ...µ.PROP.....
00 00 00 43 61 6E 43 6F 6C 6C 69 64 65 02 01 01  ...CanCollide...
01 01 01 50 52 4F 50 12 00 00 00 0A 00 00 00 45  ...PROP........E
6C 61 73 74 69 63 69 74 79 04 7E 7E 7E 7E 7E 00  lasticity.~~~~~.
00 00 00 00 00 00 00 00 00 00 00 00 00 00 50 52  ..............PR
4F 50 12 00 00 00 08 00 00 00 46 72 69 63 74 69  OP........Fricti
6F 6E 04 7D 7D 7D 7D 7D 33 33 33 33 33 33 33 33  on.}}}}}33333333
33 33 34 34 34 34 34 50 52 4F 50 12 00 00 00 0B  3344444PROP.....
00 00 00 46 72 6F 6E 74 50 61 72 61 6D 41 04 7E  ...FrontParamA.~
7E 7E 7E 7E 00 00 00 00 00 00 00 00 00 00 01 01  ~~~~............
01 01 01 50 52 4F 50 12 00 00 00 0B 00 00 00 46  ...PROP........F
72 6F 6E 74 50 61 72 61 6D 42 04 7E 7E 7E 7E 7E  rontParamB.~~~~~
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 50  ...............P
52 4F 50 12 00 00 00 0C 00 00 00 46 72 6F 6E 74  ROP........Front
53 75 72 66 61 63 65 12 00 00 00 00 00 00 00 00  Surface.........
00 00 00 00 00 00 00 00 00 00 05 00 50 52 4F 50  ............PROP
12 00 00 00 11 00 00 00 46 72 6F 6E 74 53 75 72  ........FrontSur
66 61 63 65 49 6E 70 75 74 12 00 00 00 00 00 00  faceInput.......
00 00 00 00 00 00 00 00 00 00 00 00 00 00 50 52  ..............PR
4F 50 12 00 00 00 0A 00 00 00 4C 65 66 74 50 61  OP........LeftPa
72 61 6D 41 04 7E 7E 7E 7E 7E 00 00 00 00 00 00  ramA.~~~~~......
00 00 00 00 01 01 01 01 01 50 52 4F 50 12 00 00  .........PROP...
00 0A 00 00 00 4C 65 66 74 50 61 72 61 6D 42 04  .....LeftParamB.
7E 7E 7E 7E 7E 00 00 00 00 00 00 00 00 00 00 00  ~~~~~...........
00 00 00 00 50 52 4F 50 12 00 00 00 0B 00 00 00  ....PROP........
4C 65 66 74 53 75 72 66 61 63 65 12 00 00 00 00  LeftSurface.....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 05 00  ................
50 52 4F 50 12 00 00 00 10 00 00 00 4C 65 66 74  PROP........Left
53 75 72 66 61 63 65 49 6E 70 75 74 12 00 00 00  SurfaceInput....
```

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 50 52 4F 50 12 00 00 00 06 00 00 00 4C 6F 63  .PROP........Loc
6B 65 64 02 01 00 00 00 00 50 52 4F 50 12 00 00  ked......PROP...
00 08 00 00 00 4D 61 74 65 72 69 61 6C 12 00 00  .....Material...
00 00 00 00 00 00 00 00 01 03 01 01 01 00 20 00  .............. .
00 00 50 52 4F 50 12 00 00 00 04 00 00 00 4E 61  ..PROP........Na
6D 65 01 09 00 00 00 42 61 73 65 50 6C 61 74 65  me.....BasePlate
06 00 00 00 53 70 68 65 72 65 08 00 00 00 43 79  ....Sphere....Cy
6C 69 6E 64 65 72 05 00 00 00 42 6C 6F 63 6B 04  linder....Block.
00 00 00 46 6C 61 74 50 52 4F 50 12 00 00 00 0B  ...FlatPROP.....
00 00 00 52 65 66 6C 65 63 74 61 6E 63 65 04 00  ...Reflectance..
00 7E 00 7E 00 00 00 00 00 00 00 00 00 00 00 00  .~.~............
00 00 00 50 52 4F 50 12 00 00 00 0B 00 00 00 52  ...PROP........R
69 67 68 74 50 61 72 61 6D 41 04 7E 7E 7E 7E 7E  ightParamA.~~~~~
00 00 00 00 00 00 00 00 00 00 01 01 01 01 01 50  ...............P
52 4F 50 12 00 00 00 0B 00 00 00 52 69 67 68 74  ROP........Right
50 61 72 61 6D 42 04 7E 7E 7E 7E 7E 00 00 00 00  ParamB.~~~~~....
00 00 00 00 00 00 00 00 00 00 00 50 52 4F 50 12  ...........PROP.
00 00 00 0C 00 00 00 52 69 67 68 74 53 75 72 66  .......RightSurf
61 63 65 12 00 00 00 00 00 00 00 00 00 00 00 00  ace.............
00 00 00 00 00 00 05 00 50 52 4F 50 12 00 00 00  ........PROP....
11 00 00 00 52 69 67 68 74 53 75 72 66 61 63 65  ....RightSurface
49 6E 70 75 74 12 00 00 00 00 00 00 00 00 00 00  Input...........
00 00 00 00 00 00 00 00 00 00 50 52 4F 50 12 00  ..........PROP..
00 00 0B 00 00 00 52 6F 74 56 65 6C 6F 63 69 74  ......RotVelocit
79 0E 00 00 00 00 00 00 00 00 00 00 00 00 00 00  y...............
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 50 52  ..............PR
4F 50 12 00 00 00 09 00 00 00 54 6F 70 50 61 72  OP........TopPar
61 6D 41 04 7E 7E 7E 7E 7E 00 00 00 00 00 00 00  amA.~~~~~.......
00 00 00 01 01 01 01 01 50 52 4F 50 12 00 00 00  ........PROP....
09 00 00 00 54 6F 70 50 61 72 61 6D 42 04 7E 7E  ....TopParamB.~~
7E 7E 7E 00 00 00 00 00 00 00 00 00 00 00 00 00  ~~~.............
```

```
00 00 50 52 4F 50 12 00 00 00 0A 00 00 00 54 6F   ..PROP........To
70 53 75 72 66 61 63 65 12 00 00 00 00 00 00 00   pSurface........
00 00 00 00 00 00 00 00 03 00 00 05 02 50 52 4F   .............PRO
50 12 00 00 00 0F 00 00 00 54 6F 70 53 75 72 66   P........TopSurf
61 63 65 49 6E 70 75 74 12 00 00 00 00 00 00 00   aceInput........
00 00 00 00 00 00 00 00 00 00 00 00 00 50 52 4F   .............PRO
50 12 00 00 00 0C 00 00 00 54 72 61 6E 73 70 61   P........Transpa
72 65 6E 63 79 04 00 00 00 00 00 00 00 00 00 00   rency...........
00 00 00 00 00 00 00 00 00 00 50 52 4F 50 12 00   ..........PROP..
00 00 08 00 00 00 56 65 6C 6F 63 69 74 79 0E 00   ......Velocity..
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 50 52 4F 50 12   ...........PROP.
00 00 00 0D 00 00 00 66 6F 72 6D 46 61 63 74 6F   .......formFacto
72 52 61 77 12 00 00 00 00 00 00 00 00 00 00 00   rRaw............
00 00 00 00 01 00 00 01 02 50 52 4F 50 12 00 00   .........PROP...
00 05 00 00 00 73 68 61 70 65 12 00 00 00 00 00   .....shape......
00 00 00 00 00 00 00 00 00 00 01 00 02 01 01 50   ...............P
52 4F 50 12 00 00 00 04 00 00 00 73 69 7A 65 0E   ROP........size.
88 82 82 81 82 00 73 0C 33 00 00 33 CC 33 00 00   ˆ‚‚.‚.s.3..3Ì3..
34 CC 34 00 7F 82 82 81 7D 33 73 0C 0C 99 33 33   4Ì4..‚‚.}3s..™33
CC CC 99 34 34 CC CE 9A 88 82 82 81 82 00 73 0C   ÌÌ™44ÌÎšˆ‚‚.‚.s.
66 00 00 33 CC 66 00 00 34 CC 68 00               f..3Ìf..4Ìh.
```

8.3.5.1 Anchored

Decompressed:

```
12 00 00 00 08 00 00 00 41 6E 63 68 6F 72 65 64   ........Anchored
02 01 00 00 00 00                                 ......
```

A data type value of **0x2** indicates that this property's data type is "**Boolean**".

The data array for this property is { **True**, **False**, **False**, **False**, **False** }.

8.3.5.2 BackParamA

Decompressed:

```
12 00 00 00 0A 00 00 00 42 61 63 6B 50 61 72 61  ........BackPara
6D 41 04 7E 7E 7E 7E 7E 00 00 00 00 00 00 00 00  mA.~~~~~........
00 00 01 01 01 01 01                             .......
```

A data type value of **0x4** indicates that this property's data type is "**Float**".

The data array for this property is { **-.5**, **-.5**, **-.5**, **-.5**, **-.5** }.

8.3.5.3 BackParamB

Decompressed:

```
12 00 00 00 0A 00 00 00 42 61 63 6B 50 61 72 61  ........BackPara
6D 42 04 7E 7E 7E 7E 7E 00 00 00 00 00 00 00 00  mB.~~~~~........
00 00 00 00 00 00 00                             .......
```

A data type value of **0x4** indicates that this property's data type is "**Float**".

The data array for this property is { **.5**, **.5**, **.5**, **.5**, **.5** }.

8.3.5.4 BackSurface

Decompressed:

```
12 00 00 00 0B 00 00 00 42 61 63 6B 53 75 72 66  ........BackSurf
61 63 65 12 00 00 00 00 00 00 00 00 00 00 00 00  ace.............
00 00 00 00 00 00 05 00                          ........
```

A data type value of **0x12** indicates that this property's data type is "**Enumeration**".

The raw data array for this property is { **0x0**, **0x0**, **0x0**, **0x5**, **0x0** }.

These correspond[4] to enum values of { "**Smooth**", "**Smooth**", "**Smooth**", "**Universal**", "**Smooth**" }.

8.3.5.5 BackSurfaceInput

Decompressed:

```
12 00 00 00 10 00 00 00 42 61 63 6B 53 75 72 66  ........BackSurf
```

[4] See Roblox Wiki page on SurfaceType (http://wiki.roblox.com/index.php?title=API:Enum/SurfaceType).

```
61 63 65 49 6E 70 75 74 12 00 00 00 00 00 00 00  aceInput........
00 00 00 00 00 00 00 00 00 00 00 00 00           .............
```

A data type value of **0x12** indicates that this property's data type is "**Enumeration**".

The raw data array for this property is { **0x0**, **0x0**, **0x0**, **0x0**, **0x0** }.

These correspond[5] to enum values of { "**NoInput**", "**NoInput**", "**NoInput**", "**NoInput**", "**NoInput**" }.

8.3.5.6 BottomParamA

```
Decompressed:
        12 00 00 00 0C 00 00 00 42 6F 74 74 6F 6D 50 61  ........BottomPa
        72 61 6D 41 04 7E 7E 7E 7E 7E 00 00 00 00 00 00  ramA.~~~~~......
        00 00 00 00 01 01 01 01 01                       .........
```

A data type value of **0x4** indicates that this property's data type is "**Float**".

The data array for this property is { **-.5**, **-.5**, **-.5**, **-.5**, **-.5** }.

8.3.5.7 BottomParamB

```
Decompressed:
        12 00 00 00 0C 00 00 00 42 6F 74 74 6F 6D 50 61  ........BottomPa
        72 61 6D 42 04 7E 7E 7E 7E 7E 00 00 00 00 00 00  ramB.~~~~~......
        00 00 00 00 00 00 00 00 00                       .........
```

A data type value of **0x4** indicates that this property's data type is "**Float**".

The data array for this property is { **.5**, **.5**, **.5**, **.5**, **.5** }.

8.3.5.8 BottomSurface

```
Decompressed:

        12 00 00 00 0D 00 00 00 42 6F 74 74 6F 6D 53 75  ........BottomSu
        72 66 61 63 65 12 00 00 00 00 00 00 00 00 00 00  rface...........
        00 00 00 00 00 04 00 00 05 02                    ..........
```

[5] See Roblox Wiki page on InputType (http://wiki.roblox.com/index.php?title=API:Enum/InputType)

A data type value of **0x12** indicates that this property's data type is "**Enumeration**".

The raw data array for this property is { **0x4**, **0x0**, **0x0**, **0x5**, **0x2** }.

These correspond to enum values of { "**Inlet**", "**Smooth**", "**Smooth**", "**Universal**", "**Weld**" }.

8.3.5.9 BottomSurfaceInput

```
Decompressed:
          12 00 00 00 12 00 00 00 42 6F 74 74 6F 6D 53 75   ........BottomSu
          72 66 61 63 65 49 6E 70 75 74 12 00 00 00 00 00   rfaceInput......
          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      ...............
```

A data type value of **0x12** indicates that this property's data type is "**Enumeration**".

The raw data array for this property is { **0x0**, **0x0**, **0x0**, **0x0**, **0x0** }.

These correspond to enum values of { "**NoInput**", "**NoInput**", "**NoInput**", "**NoInput**", "**NoInput**" }.

8.3.5.10 BrickColor

```
Decompressed:
          12 00 00 00 0A 00 00 00 42 72 69 63 6B 43 6F 6C   ........BrickCol
          6F 72 0B 00 00 00 00 00 00 00 00 00 00 00 00 00   or..............
          03 00 C7 C7 18 EE 01                              ..ÇÇ.î.
```

A data type value of **0x0B** indicates that this property's data type is "**BrickColor**".

The raw data array for this property is { **199**, **199**, **24**, **1006**, **1** }.

These correspond[6] to BrickColor names of { "**Dark stone grey**", "**Dark stone grey**", "**Bright yellow**", "**Alder**", "**White**" }.

8.3.5.11 CFrame

```
Decompressed:
          12 00 00 00 06 00 00 00 43 46 72 61 6D 65 10 02   ........CFrame..
          02 00 00 00 00 00 00 00 00 00 00 00 80 3F 00 00   ............€?..
          80 BF 00 00 00 00 00 00 00 00 00 00 00 00 00 00   €¿..............
          80 BF 2E BD 3B B3 02 02 00 80 83 83 7F 00 00 0C   €¿.½;³...€ƒƒ....
```

[6] See Roblox Wiki page on BrickColor Codes (http://wiki.roblox.com/index.php?title=BrickColor_codes)

```
0C 66 00 00 CC CC 66 00 00 D1 C7 00 7E 81 81 80   .f..ÌÌf..ÑÇ.~..€
7C 38 72 0C 0B 85 51 8F 28 85 1E ED 5C F4 20 BA   |8r..…Q.(….í\ô º
00 83 83 81 81 00 90 A0 CC CC 00 00 00 CC CD 00   .ƒƒ.... ÌÌ...ÌÍ.
01 01 B5 01                                       ..µ.
```

A data type value of **0x10** indicates that this property's data type is "**CFrame**".

The position data values for this property are the following:
$\{(\mathbf{0}, -.\mathbf{61}, \mathbf{0}), (\mathbf{2}, \mathbf{5.79}, -\mathbf{25}), (-\mathbf{16.8}, \mathbf{4.19}, -\mathbf{26}), (-\mathbf{16.8}, \mathbf{2.09}, -\mathbf{7.2}), (\mathbf{1.4}, .\mathbf{19}, -\mathbf{7.2})\}$.

The rotation matrices are the following:

$$\left\{\begin{bmatrix}\mathbf{1} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{1} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix}, \begin{bmatrix}\mathbf{1} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{1} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix}, \begin{bmatrix}\mathbf{0} & \mathbf{0} & \mathbf{1}\\ -\mathbf{1} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & -\mathbf{1} & -\mathbf{4.37}\times\mathbf{10}^{-\mathbf{8}}\end{bmatrix}, \begin{bmatrix}\mathbf{1} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{1} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix}, \begin{bmatrix}\mathbf{1} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{1} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix}\right\}.$$

8.3.5.12 CanCollide

Decompressed:

```
12 00 00 00 0A 00 00 00 43 61 6E 43 6F 6C 6C 69   ........CanColli
64 65 02 01 01 01 01 01                           de......
```

A data type value of **0x2** indicates that this property's data type is "**Boolean**".

The data array for this property is { **0x1**, **0x1**, **0x1**, **0x1**, **0x1** }.

These correspond to Boolean values { **True**, **True**, **True**, **True**, **True** }.

8.3.5.13 Elasticity

Decompressed:

```
12 00 00 00 0A 00 00 00 45 6C 61 73 74 69 63 69   ........Elastici
74 79 04 7E 7E 7E 7E 7E 00 00 00 00 00 00 00 00   ty.~~~~~........
00 00 00 00 00 00 00                              .......
```

A data type value of **0x4** indicates that this property's data type is "**Float**".

The value array for this property is { **.5**, **.5**, **.5**, **.5**, **.5** }.

8.3.5.14 Friction

Decompressed:

```
12 00 00 00 08 00 00 00 46 72 69 63 74 69 6F 6E  ........Friction
04 7D 7D 7D 7D 7D 33 33 33 33 33 33 33 33 33 33  .}}}}}3333333333
34 34 34 34 34                                   44444
```

A data type value of **0x4** indicates that this property's data type is "**Float**".

The value array for this property is { **.3**, **.3**, **.3**, **.3**, **.3** }.

8.3.5.15 FrontParamA

```
Decompressed:
    12 00 00 00 0B 00 00 00 46 72 6F 6E 74 50 61 72  ........FrontPar
    61 6D 41 04 7E 7E 7E 7E 7E 00 00 00 00 00 00 00  amA.~~~~~.......
    00 00 00 01 01 01 01 01                          ........
```

A data type value of **0x4** indicates that this property's data type is "**Float**".

The data array for this property is { **-.5**, **-.5**, **-.5**, **-.5**, **-.5** }.

8.3.5.16 FrontParamB

```
Decompressed:
    12 00 00 00 0B 00 00 00 46 72 6F 6E 74 50 61 72  ........FrontPar
    61 6D 42 04 7E 7E 7E 7E 7E 00 00 00 00 00 00 00  amB.~~~~~.......
    00 00 00 00 00 00 00 00                          ........
```

A data type value of **0x4** indicates that this property's data type is "**Float**".

The data array for this property is { **.5**, **.5**, **.5**, **.5**, **.5** }.

8.3.5.17 FrontSurface

```
Decompressed:
    12 00 00 00 0C 00 00 00 46 72 6F 6E 74 53 75 72  ........FrontSur
    66 61 63 65 12 00 00 00 00 00 00 00 00 00 00 00  face............
    00 00 00 00 00 00 00 05 00                       .........
```

A data type value of **0x12** indicates that this property's data type is "**Enumeration**".

The raw data array for this property is { **0x0**, **0x0**, **0x0**, **0x5**, **0x0** }.

These correspond to enum values of { “**Smooth**”, “**Smooth**”, “**Smooth**”, “**Universal**”, “**Smooth**” }.

8.3.5.18 FrontSurfaceInput

```
Decompressed:
        12 00 00 00 11 00 00 00 46 72 6F 6E 74 53 75 72   ........FrontSur
        66 61 63 65 49 6E 70 75 74 12 00 00 00 00 00 00   faceInput.......
        00 00 00 00 00 00 00 00 00 00 00 00 00 00         ..............
```

A data type value of **0x12** indicates that this property’s data type is “**Enumeration**”.

The raw data array for this property is { **0x0**, **0x0**, **0x0**, **0x0**, **0x0** }.

These correspond to enum values of { “**NoInput**”, “**NoInput**”, “**NoInput**”, “**NoInput**”, “**NoInput**” }.

8.3.5.19 LeftParamA

```
Decompressed:
        12 00 00 00 0A 00 00 00 4C 65 66 74 50 61 72 61   ........LeftPara
        6D 41 04 7E 7E 7E 7E 7E 00 00 00 00 00 00 00 00   mA.~~~~~........
        00 00 01 01 01 01 01                              .......
```

A data type value of **0x4** indicates that this property’s data type is “**Float**”.

The data array for this property is { **-.5**, **-.5**, **-.5**, **-.5**, **-.5** }.

8.3.5.20 LeftParamB

```
Decompressed:
        12 00 00 00 0A 00 00 00 4C 65 66 74 50 61 72 61   ........LeftPara
        6D 42 04 7E 7E 7E 7E 7E 00 00 00 00 00 00 00 00   mB.~~~~~........
        00 00 00 00 00 00 00                              .......
```

A data type value of **0x4** indicates that this property’s data type is “**Float**”.

The data array for this property is { **.5**, **.5**, **.5**, **.5**, **.5** }.

8.3.5.21 LeftSurface

```
Decompressed:
        12 00 00 00 0B 00 00 00 4C 65 66 74 53 75 72 66   ........LeftSurf
```

```
        61 63 65 12 00 00 00 00 00 00 00 00 00 00 00 00  ace.............
        00 00 00 00 00 00 05 00                          ........
```

A data type value of **0x12** indicates that this property's data type is "**Enumeration**".

The raw data array for this property is { **0x0**, **0x0**, **0x0**, **0x5**, **0x0** }.

These correspond to enum values of { "**Smooth**", "**Smooth**", "**Smooth**", "**Universal**", "**Smooth**" }.

8.3.5.22 LeftSurfaceInput

```
Decompressed:
        12 00 00 00 10 00 00 00 4C 65 66 74 53 75 72 66  ........LeftSurf
        61 63 65 49 6E 70 75 74 12 00 00 00 00 00 00 00  aceInput........
        00 00 00 00 00 00 00 00 00 00 00 00 00           .............
```

A data type value of **0x12** indicates that this property's data type is "**Enumeration**".

The raw data array for this property is { **0x0**, **0x0**, **0x0**, **0x0**, **0x0** }.

These correspond to enum values of { "**NoInput**", "**NoInput**", "**NoInput**", "**NoInput**", "**NoInput**" }.

8.3.5.23 Locked

```
Decompressed:
        12 00 00 00 06 00 00 00 4C 6F 63 6B 65 64 02 01  ........Locked..
        00 00 00 00                                      ....
```

A data type value of **0x2** indicates that this property's data type is "**Boolean**".

The data array for this property is { **0x1**, **0x0**, **0x0**, **0x0**, **0x0**}.

These correspond to Boolean values { **True**, **False**, **False**, **False**, **False** }.

8.3.5.24 Material

```
Decompressed:
        12 00 00 00 08 00 00 00 4D 61 74 65 72 69 61 6C  ........Material
        12 00 00 00 00 00 00 00 00 00 00 01 03 01 01 01  ................
        00 20 00 00 00                                   . ...
```

A data type value of **0x12** indicates that this property's data type is "**Enumeration**".

The raw data array for this property is { **0x100**, **0x320**, **0x100**, **0x100**, **0x100** }.

These correspond[7] to material values of { "**Plastic**", "**Slate**", "**Plastic**", "**Plastic**", "**Plastic**" }.

8.3.5.25 Name

```
Decompressed:
            12 00 00 00 04 00 00 00 4E 61 6D 65 01 09 00 00  ........Name....
            00 42 61 73 65 50 6C 61 74 65 06 00 00 00 53 70  .BasePlate....Sp
            68 65 72 65 08 00 00 00 43 79 6C 69 6E 64 65 72  here....Cylinder
            05 00 00 00 42 6C 6F 63 6B 04 00 00 00 46 6C 61  ....Block....Fla
            74                                               t
```

A data type value of **0x1** indicates that this property's data type is "**String**".

The data array for this property is { "**BasePlate**", "**Sphere**", "**Cylinder**", "**Block**", "**Flat**" }.

8.3.5.26 Reflectance

```
Decompressed:
            12 00 00 00 0B 00 00 00 52 65 66 6C 65 63 74 61  ........Reflecta
            6E 63 65 04 00 00 7E 00 7E 00 00 00 00 00 00 00  nce...~.~.......
            00 00 00 00 00 00 00 00                          ........
```

A data type value of **0x4** indicates that this property's data type is "**Float**".

The data array for this property is { **0**, **0**, **.5**, **0**, **.5** }.

8.3.5.27 RightParamA

```
Decompressed:
            12 00 00 00 0B 00 00 00 52 69 67 68 74 50 61 72  ........RightPar
            61 6D 41 04 7E 7E 7E 7E 7E 00 00 00 00 00 00 00  amA.~~~~~.......
            00 00 00 01 01 01 01 01                          ........
```

A data type value of **0x4** indicates that this property's data type is "**Float**".

The data array for this property is { **-.5**, **-.5**, **-.5**, **-.5**, **-.5** }.

[7] See Roblox Wiki page on Material values (http://wiki.roblox.com/index.php?title=Material_(Enum)).

8.3.5.28 RightParamB

Decompressed:

```
12 00 00 00 0B 00 00 00 52 69 67 68 74 50 61 72  ........RightPar
61 6D 42 04 7E 7E 7E 7E 7E 00 00 00 00 00 00 00  amB.~~~~~.......
00 00 00 00 00 00 00 00                          ........
```

A data type value of **0x4** indicates that this property's data type is "**Float**".

The data array for this property is { **.5**, **.5**, **.5**, **.5**, **.5** }.

8.3.5.29 RightSurface

Decompressed:

```
12 00 00 00 0C 00 00 00 52 69 67 68 74 53 75 72  ........RightSur
66 61 63 65 12 00 00 00 00 00 00 00 00 00 00 00  face............
00 00 00 00 00 00 00 05 00                       .........
```

A data type value of **0x12** indicates that this property's data type is "**Enumeration**".

The raw data array for this property is { **0x0**, **0x0**, **0x0**, **0x5**, **0x0** }.

These correspond to enum values of { "**Smooth**", "**Smooth**", "**Smooth**", "**Universal**", "**Smooth**" }.

8.3.5.30 RightSurfaceInput

Decompressed:

```
12 00 00 00 11 00 00 00 52 69 67 68 74 53 75 72  ........RightSur
66 61 63 65 49 6E 70 75 74 12 00 00 00 00 00 00  faceInput.......
00 00 00 00 00 00 00 00 00 00 00 00 00 00        ..............
```

A data type value of **0x12** indicates that this property's data type is "**Enumeration**".

The raw data array for this property is { **0x0**, **0x0**, **0x0**, **0x0**, **0x0** }.

These correspond to enum values of { "**NoInput**", "**NoInput**", "**NoInput**", "**NoInput**", "**NoInput**" }.

8.3.5.31 RotVelocity

Decompressed:

```
12 00 00 00 0B 00 00 00 52 6F 74 56 65 6C 6F 63  ........RotVeloc
```

```
69 74 79 0E 00 00 00 00 00 00 00 00 00 00 00 00   ity.............
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

A data type value of **0x0E** indicates that this property's data type is "**Vector3**".

The value array for this property is { $(0, 0, 0), (0, 0, 0), (0, 0, 0), (0, 0, 0), (0, 0, 0)$ }.

8.3.5.32 TopParamA

```
Decompressed:
          12 00 00 00 09 00 00 00 54 6F 70 50 61 72 61 6D   ........TopParam
          41 04 7E 7E 7E 7E 7E 00 00 00 00 00 00 00 00 00   A.~~~~~.........
          00 01 01 01 01 01                                 ......
```

A data type value of **0x4** indicates that this property's data type is "**Float**".

The data array for this property is { **-.5**, **-.5**, **-.5**, **-.5**, **-.5** }.

8.3.5.33 TopParamB

```
Decompressed:
          12 00 00 00 09 00 00 00 54 6F 70 50 61 72 61 6D   ........TopParam
          42 04 7E 7E 7E 7E 7E 00 00 00 00 00 00 00 00 00   B.~~~~~.........
          00 00 00 00 00 00                                 ......
```

A data type value of **0x4** indicates that this property's data type is "**Float**".

The data array for this property is { **.5**, **.5**, **.5**, **.5**, **.5** }.

8.3.5.34 TopSurface

```
Decompressed:
          12 00 00 00 0A 00 00 00 54 6F 70 53 75 72 66 61   ........TopSurfa
          63 65 12 00 00 00 00 00 00 00 00 00 00 00 00 00   ce..............
          00 00 03 00 00 05 02                              .......
```

A data type value of **0x12** indicates that this property's data type is "**Enumeration**".

The raw data array for this property is { **0x3**, **0x0**, **0x0**, **0x5**, **0x2** }.

These correspond to enum values of { "**Studs**", "**Smooth**", "**Smooth**", "**Universal**", "**Weld**" }.

8.3.5.35 TopSurfaceInput

```
Decompressed:
        12 00 00 00 0F 00 00 00 54 6F 70 53 75 72 66 61  ........TopSurfa
        63 65 49 6E 70 75 74 12 00 00 00 00 00 00 00 00  ceInput.........
        00 00 00 00 00 00 00 00 00 00 00 00              ............
```

A data type value of **0x12** indicates that this property's data type is "**Enumeration**".

The raw data array for this property is { **0x0**, **0x0**, **0x0**, **0x0**, **0x0** }.

These correspond to enum values of { "**NoInput**", "**NoInput**", "**NoInput**", "**NoInput**", "**NoInput**" }.

8.3.5.36 Transparency

```
Decompressed:
        12 00 00 00 0C 00 00 00 54 72 61 6E 73 70 61 72  ........Transpar
        65 6E 63 79 04 00 00 00 00 00 00 00 00 00 00 00  ency............
        00 00 00 00 00 00 00 00 00                       .........
```

A data type value of **0x4** indicates that this property's data type is "**Float**".

The data array for this property is { **0**, **0**, **0**, **0**, **0** }.

8.3.5.37 Velocity

```
Decompressed:
        12 00 00 00 08 00 00 00 56 65 6C 6F 63 69 74 79  ........Velocity
        0E 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
        00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
        00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
        00 00 00 00 00 00 00 00 00 00 00 00 00           .............
```

A data type value of **0xE** indicates that this property's data type is "**Vector3**".

The value array for this property is { $(\mathbf{0},\ \mathbf{0},\ \mathbf{0}), (\mathbf{0},\ \mathbf{0},\ \mathbf{0}), (\mathbf{0},\ \mathbf{0},\ \mathbf{0}), (\mathbf{0},\ \mathbf{0},\ \mathbf{0}), (\mathbf{0},\ \mathbf{0},\ \mathbf{0})$ }.

8.3.5.38 formFactorRaw

Decompressed:

```
12 00 00 00 0D 00 00 00 66 6F 72 6D 46 61 63 74   ........formFact
6F 72 52 61 77 12 00 00 00 00 00 00 00 00 00 00   orRaw...........
00 00 00 00 00 01 00 00 01 02                     ..........
```

A data type value of 0x12 indicates that this property's data type is "Enumeration".

The raw data array for this property is { **0x1**, **0x0**, **0x0**, **0x1**, **0x2** }.

These correspond[8] to enum values of { "**Brick**", "**Symmetric**", "**Symmetric**", "**Brick**", "**Plate**" }.

8.3.5.39 Shape

Decompressed:

```
12 00 00 00 05 00 00 00 73 68 61 70 65 12 00 00   ........shape...
00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 02   ................
01 01                                             ..
```

A data type value of **0x12** indicates that this property's data type is "**Enumeration**".

The raw data array for this property is { **0x1**, **0x0**, **0x2**, **0x1**, **0x1** }.

These correspond[9] to enum values of { "**Block**", "**Ball**", "**Cylinder**", "**Block**", "**Block**" }.

8.3.5.40 Size

Decompressed:

```
12 00 00 00 04 00 00 00 73 69 7A 65 0E 88 82 82   ........size.ˆ,,
81 82 00 73 0C 33 00 00 33 CC 33 00 00 34 CC 34   .,.s.3..3Ì3..4Ì4
00 7F 82 82 81 7D 33 73 0C 0C 99 33 33 CC CC 99   ..,,.}3s..™33ÌÌ™
34 34 CC CE 9A 88 82 82 81 82 00 73 0C 66 00 00   44ÌÎšˆ,,.,.s.f..
33 CC 66 00 00 34 CC 68 00                        3Ìf..4Ìh.
```

A data type value of **0xE** indicates that this property's data type is "**Vector3**".

[8]See Roblox Wiki page on FormFactor (http://wiki.roblox.com/index.php?title=API:Enum/FormFactor).
[9]See Roblox Wiki page on FormFactor (http://wiki.roblox.com/index.php?title=API:Enum/FormFactor).

The data array for this property is
$\{ (512, 1.2, 512), (11.6, 11.6, 11.6), (8.4, 8.4, 8.4), (4.8, 4.2, 5.6), (8, .4, 8) \}$

8.3.6 Players

```
00000E70                               50 52 4F 50 1A 00 00            PROP...
00000E80  00 18 00 00 00 00 00 00 00 F0 09 15 00 00 00 04   .........ð......
00000E90  00 00 00 4E 61 6D 65 01 07 00 00 00 50 6C 61 79   ...Name.....Play
00000EA0  65 72 73                                          ers
```

```
Decompressed:
          15 00 00 00 12 00 00 00 43 68 61 72 61 63 74 65   ........Characte
          72 41 75 74 6F 4C 6F 61 64 73 02 01 50 52 4F 50   rAutoLoads..PROP
          15 00 00 00 04 00 00 00 4E 61 6D 65 01 07 00 00   ........Name....
          00 50 6C 61 79 65 72 73                           .Players
```

8.3.6.1 CharacterAutoLoads

```
Decompressed:
          15 00 00 00 12 00 00 00 43 68 61 72 61 63 74 65   ........Characte
          72 41 75 74 6F 4C 6F 61 64 73 02 01               rAutoLoads..
```

A data type value of **0x2** indicates that this property's data type is "**Boolean**".

The raw data array for this property is { **0x1** }, which indicates a value of { **True** }.

8.3.6.2 Name

```
Decompressed:
          15 00 00 00 04 00 00 00 4E 61 6D 65 01 07 00 00   ........Name....
          00 50 6C 61 79 65 72 73                           .Players
```

A data type value of **0x1** indicates that this property's data type is "**String**".

The raw data array for this property is { "**Players**" }.

8.3.7 Workspace

```
00000EA0                    1C 00 00 00 1A 00 00 00 00            .........
```

```
00000EB0  00 00 00 F0 0B 29 00 00 00 0D 00 00 00 43 75 72  ...ð.).......Cur
00000EC0  72 65 6E 74 43 61 6D 65 72 61 13 00 00 00 06 50  rentCamera.....P
00000ED0  52 4F 50 26 00 00 00 24 00 00 00 00 00 00 00 F0  ROP&...$.......ð
00000EE0  15 29 00 00 00 13 00 00 00 44 69 73 74 72 69 62  .).......Distrib
00000EF0  75 74 65 64 47 61 6D 65 54 69 6D 65 05 00 00 00  utedGameTime....
00000F00  00 00 00 00 00 50 52 4F 50 1C 00 00 00 1A 00 00  .....PROP.......
00000F10  00 00 00 00 00 F0 0B 29 00 00 00 10 00 00 00 46  .....ð.).......F
00000F20  69 6C 74 65 72 69 6E 67 45 6E 61 62 6C 65 64 02  ilteringEnabled.
00000F30  00 50 52 4F 50 26 00 00 00 24 00 00 00 00 00 00  .PROP&...$......
00000F40  00 F0 15 29 00 00 00 0E 00 00 00 4D 6F 64 65 6C  .ð.).......Model
00000F50  49 6E 50 72 69 6D 61 72 79 10 02 00 00 00 00 00  InPrimary.......
00000F60  00 00 00 00 00 00 00 50 52 4F 50 1C 00 00 00 1A  .......PROP.....
00000F70  00 00 00 00 00 00 00 F0 0B 29 00 00 00 04 00 00  .......ð.)......
00000F80  00 4E 61 6D 65 01 09 00 00 00 57 6F 72 6B 73 70  .Name.....Worksp
00000F90  61 63 65 50 52 4F 50 1A 00 00 00 18 00 00 00 00  acePROP.........
00000FA0  00 00 00 F0 09 29 00 00 00 0B 00 00 00 50 72 69  ...ð.).......Pri
00000FB0  6D 61 72 79 50 61 72 74 13 00 00 00 01 50 52 4F  maryPart.....PRO
00000FC0  50 1C 00 00 00 1A 00 00 00 00 00 00 00 F0 0B 29  P............ð.)
00000FD0  00 00 00 10 00 00 00 53 74 72 65 61 6D 69 6E 67  .......Streaming
00000FE0  45 6E 61 62 6C 65 64 02 00                       Enabled..
```

Decompressed:

```
29 00 00 00 0D 00 00 00 43 75 72 72 65 6E 74 43  ).......CurrentC
61 6D 65 72 61 13 00 00 00 06 50 52 4F 50 29 00  amera.....PROP).
00 00 13 00 00 00 44 69 73 74 72 69 62 75 74 65  ......Distribute
64 47 61 6D 65 54 69 6D 65 05 00 00 00 00 00 00  dGameTime.......
00 00 50 52 4F 50 29 00 00 00 10 00 00 00 46 69  ..PROP).......Fi
6C 74 65 72 69 6E 67 45 6E 61 62 6C 65 64 02 00  lteringEnabled..
50 52 4F 50 29 00 00 00 0E 00 00 00 4D 6F 64 65  PROP).......Mode
6C 49 6E 50 72 69 6D 61 72 79 10 02 00 00 00 00  lInPrimary......
00 00 00 00 00 00 00 00 50 52 4F 50 29 00 00 00  ........PROP)...
04 00 00 00 4E 61 6D 65 01 09 00 00 00 57 6F 72  ....Name.....Wor
6B 73 70 61 63 65 50 52 4F 50 29 00 00 00 0B 00  kspacePROP).....
00 00 50 72 69 6D 61 72 79 50 61 72 74 13 00 00  ..PrimaryPart...
```

```
00 01 50 52 4F 50 29 00 00 00 10 00 00 00 53 74  ..PROP).......St
72 65 61 6D 69 6E 67 45 6E 61 62 6C 65 64 02 00  reamingEnabled..
```

8.3.7.1 CurrentCamera

Decompressed:

```
29 00 00 00 0D 00 00 00 43 75 72 72 65 6E 74 43  ).......CurrentC
61 6D 65 72 61 13 00 00 00 06                    amera.....
```

A data type value of **0x13** indicates that this property's data type is "**Referent**".

The raw data for this property is **0x6**, which indicates the actual value is **0x3**. Referencing the referent table (see section 7.2), this value refers to the instance named "**Camera**".

8.3.7.2 DistributedGameTime

Decompressed:

```
29 00 00 00 13 00 00 00 44 69 73 74 72 69 62 75  ).......Distribu
74 65 64 47 61 6D 65 54 69 6D 65 05 00 00 00 00  tedGameTime.....
00 00 00 00                                      ....
```

A data type value of **0x5** indicates that this property's data type is "**Double**".

The raw data for this property is **0**.

8.3.7.3 FilteringEnabled

Decompressed:

```
29 00 00 00 10 00 00 00 46 69 6C 74 65 72 69 6E  ).......Filterin
67 45 6E 61 62 6C 65 64 02 00                    gEnabled..
```

A data type value of **0x2** indicates that this property's data type is "**Boolean**".

The raw data array for this property is { **0x0** }, which indicates a value of { **False** }.

8.3.7.4 ModelInPrimary

Decompressed:

```
29 00 00 00 0E 00 00 00 4D 6F 64 65 6C 49 6E 50  ).......ModelInP
72 69 6D 61 72 79 10 02 00 00 00 00 00 00 00 00  rimary..........
00 00 00 00                                      ....
```

A data type value of **0x10** indicates that this property's data type is "**CFrame**".

The position stored here is { $(\mathbf{0}, \mathbf{0}, \mathbf{0})$ }.

The rotation matrix is a special value (stored as **0x2**). This indicates an identity matrix: $\left\{\begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}\right\}$.

8.3.7.5 Name

Decompressed:

```
29 00 00 00 04 00 00 00 4E 61 6D 65 01 09 00 00  ).......Name....
00                                               .
```

A data type value of **0x1** indicates that this property's data type is "**String**".

The data array for this property is { "**Workspace**" }.

8.3.7.6 PrimaryPart

Decompressed:

```
29 00 00 00 0B 00 00 00 50 72 69 6D 61 72 79 50  ).......PrimaryP
61 72 74 13 00 00 00 01                          art.....
```

A data type value of **0x13** indicates that this property's data type is "**Referent**".

The raw data for this property is **0x1**, which indicates the actual value is **-1**. This means that there is no PrimaryPart.

8.3.7.7 StreamingEnabled

Decompressed:

```
29 00 00 00 10 00 00 00 53 74 72 65 61 6D 69 6E  ).......Streamin
67 45 6E 61 62 6C 65 64 02 00                    gEnabled..
```

A data type value of **0x2** indicates that this property's data type is "**Boolean**".

The raw data array for this property is { **0x0** }, which indicates a value of { **False** }.

8.4 Parent Data

```
00000FE0                                50 52 4E 54 22 00 00          PRNT"..
00000FF0  00 6C 00 00 00 00 00 00 00 2F 0D 00 01 00 17 21  .l......./.....!
00001000  04 02 01 00 5F 08 32 04 0A 02 36 00 14 D0 01 02  ...._.2...6..Ð..
00001010  00 00 00 00 00 0E 0F 00 00 00 00 22 00 00 00 6C  ..........."...l
00001020  00 00 00 00 00 00 00 2F 0D 00 01 00 17 21 04 02  ......./.....!..
00001030  01 00 5F 08 32 04 0A 02 36 00 14 D0 01 02 00 00  .._.2...6..Ð....
00001040  00 00 00 0E 0F 00 00 00 00                       .........
```

```
Decompressed:
          0D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
          00 00 00 00 00 00 00 00 00 00 00 00 04 02 02 02  ................
          02 02 02 08 32 04 0A 02 00 00 00 00 00 00 00 00  ....2...........
          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01  ................
          02 00 00 00 00 00 0E 0F 00 00 00 00              ............
```

8.4.1.1 Object Count

The first four bytes of this section indicate the total length of the two parent data arrays. The value is **0xD**.

8.4.1.2 Object Array

The raw object array is { **0x0**, **0x4**, **0x2**, **0x2**, **0x2**, **0x2**, **0x2**, **0x2**, **0x8**, **0x32**, **0x4**, **0xA**, **0x2** }.

The values are relative. The true values (summed and untransformed, see sections 3.3.2 and 3.4) are { **0x0**, **0x2**, **0x3**, **0x4**, **0x5, 0x6**, **0x7**, **0x8**, **0xC**, **0x25**, **0x27**, **0x2C**, **0x2D** }.

8.4.1.3 Parent Array

8.4.1.4 Parent Data

Raw	Referent	Object		Raw	Referent	Parent
0x0	0x0	Workspace		0x1	-0x1	Game
0x4	0x2	BasePlate		0x2	0x0	Workspace
0x2	0x3	Camera		0x0	0x0	Workspace
0x2	0x4	Sphere		0x0	0x0	Workspace
0x2	0x5	Cylinder		0x0	0x0	Workspace
0x2	0x6	Block		0x0	0x0	Workspace
0x2	0x7	Flat		0x0	0x0	Workspace

0x2	0x8	Decal		0xE	0x7	Flat
0x8	0xC	Players		0xF	0x0	Workspace
0x32	0x25	FilteredSelection		0x0	0x0	Workspace
0x4	0x27	FilteredSelection		0x0	0x0	Workspace
0xA	0x2C	FilteredSelection		0x0	0x0	Workspace
0x2	0x2D	Lighting		0x0	0x0	Workspace

8.5 Ending Data

```
00001040                             45 4E 44 00 00 00 00            END....
00001050  00 09 00 00 00 00 00 00 00 3C 2F 72 6F 62 6C 6F  .........</roblo
00001060  78 3E                                             x>
```

9 A Note On Solid Modeling

Due to the fact that Solid Modeling is not completely finished at the point in time in which I am writing this, I am not including my findings on its internal format. CSG physics are currently in development and Roblox employees have stated that the data stored by unions will change to include Physics data. I plan to document everything related to Solid Modeling when it is completed.

10 Ending Notes

I hope you learned something from this document! It represents months of work. So now what? I'm hoping you go out and write some tools that interop with Roblox! I plan to release an open source code library in the near future that provides an implementation of the format described herein, which should ease the burden of writing code that works with Roblox. If you want to write your own, that's cool too. Go develop!