Preface

This document describes an algorithm which corrects an issue in “APPENDIX E Encoding and Decoding Algorithms for Multimedia Objects” which was included as part of the GEDCOM 5.5 specification but later depreciated in GEDCOM 5.5.1. The issue causes data corruption if the original binary data has trailing 0xFF bytes. The corrected algorithm is currently used by several applications.

Encoding and Decoding Algorithms for Multimedia Objects (corrected)

Introduction:

Embedded multimedia objects in GEDCOM require special handling. These objects are normally represented by binary files which interfere with data transmission protocols. This document describes how the binary data is encoded for transmission and then decoded to rebuild the multimedia file.

Encoding:

Binary multimedia files are read in segments of 54 bytes. Each 54-byte segment is encoded into a GEDCOM line value of 2 to 72 characters in length. This encoded value becomes the <ENCODED_MULTIMEDIA_LINE> used in the MULTIMEDIA_RECORD (see page 26 of the GEDCOM 5.5 specification).

The encoding algorithm can be accomplished using the following steps:

1. The segment of the binary multimedia file is read in chunks of 3-bytes.

2. Each 3-byte (24-bit) chunk of the segment is divided into four 6-bit encoding keys in the range 0x00..0x3F.

3. The four 6-bit encoding keys are used to obtain four replacement characters from the Encoding Table which are appended to the encoded line segment.

4. Special processing may be required for the last chunk which may contain fewer than 3 bytes.

	Retrieved	Action
a.	0 bytes:	Do nothing. The encoding is complete.
b.	1 byte:	Append 0-bits to the right of the chunk until **two** 6-bit encoding keys can be obtained. Obtain replacement characters for each 6-bit key and append to the encoded line segment. The encoding is complete.
c.	2 bytes:	Append 0-bits to the right of the chunk until **three** 6-bit encoding keys can be obtained. Obtain replacement characters for each 6-bit key and append to the encoded line segment. The encoding is complete.

Decoding:

The decoding routine converts an encoded line value back into a segment of the original binary multimedia file.

The decoding algorithm can be accomplished using the following steps:

1. Each encoded line segment is read in chunks of four 8-bit characters.

2. Each character in the group becomes a decoding key used to look up a corresponding byte from the Decoding Table. A new 24-bit group is formed by concatenating the low-order 6 bits from each of the 4 bytes obtained from the Decoding Table.

3. This new 24-bit group is split into three bytes which are appended to the decoded line segment.

4. Special processing may be required for the last chunk of the encoded line segment which may contain fewer than four 8-bit characters.

	Retrieved	Action
a.	0 characters:	Do nothing. The conversion is complete.
b.	1 character:	The encoded line segment in invalid.
c.	2 characters:	Obtain replacement bytes for each character from the Decoding Table. Form a new 12-bit group by concatenating the low-order 6 bits from each of the 2 bytes obtained from the Decoding Table. The first 8 bits on the left form **one** decoded byte which is appended to the decoded line segment. The decoding is complete.
d.	3 characters:	Obtain replacement bytes for each character from the Decoding Table. Form a new 18-bit group by concatenating the low-order 6 bits from each of the 3 bytes obtained from the Decoding Table. The first 16 bits on the left form **two** decoded bytes which are appended to the decoded line segment. The decoding is complete.

GEDCOM 5.5 errata

Page 43 of the GEDCOM 5.5 specification defines the length constraint of the <ENCODED_MULTIMEDIA_LINE> to be {1:87}. The original and corrected algorithm will always yield an <ENCODED_MULTIMEDIA_LINE> line value between 2 and 72 characters in length. The correct length constraint is therefore {2:72}.

Encoding Table

Encoding key	Replacement character	Encoding key	Replacement character	Encoding key	Replacement character
0x00	.	0x0C	A	0x26	a
0x01	/	0x0D	B	0x27	b
0x02	0	0x0E	C	0x28	c
0x03	1	0x0F	D	0x29	d
0x04	2	0x10	E	0x2A	e
0x05	3	0x11	F	0x2B	f
0x06	4	0x12	G	0x2C	g
0x07	5	0x13	H	0x2D	h
0x08	6	0x14	I	0x2E	i
0x09	7	0x15	J	0x2F	j
0x0A	8	0x16	K	0x30	k
0x0B	9	0x17	L	0x31	l
		0x18	M	0x32	m
		0x19	N	0x33	n
		0x1A	O	0x34	o
		0x1B	P	0x35	p
		0x1C	Q	0x36	q
		0x1D	R	0x37	r
		0x1E	S	0x38	s
		0x1F	T	0x39	t
		0x20	U	0x3A	u
		0x21	V	0x3B	v
		0x22	W	0x3C	w
		0x23	X	0x3D	x
		0x24	Y	0x3E	y
		0x25	Z	0x3F	z

Decoding Table

Decoding key	Replacement byte	Decoding key	Replacement byte	Decoding key	Replacement byte
.	0x00	A	0x0C	a	0x26
/	0x01	B	0x0D	b	0x27
0	0x02	C	0x0E	c	0x28
1	0x03	D	0x0F	d	0x29
2	0x04	E	0x10	e	0x2A
3	0x05	F	0x11	f	0x2B
4	0x06	G	0x12	g	0x2C
5	0x07	H	0x13	h	0x2D
6	0x08	I	0x14	i	0x2E
7	0x09	J	0x15	j	0x2F
8	0x0A	K	0x16	k	0x30
9	0x0B	L	0x17	l	0x31
		M	0x18	m	0x32
		N	0x19	n	0x33
		O	0x1A	o	0x34
		P	0x1B	p	0x35
		Q	0x1C	q	0x36
		R	0x1D	r	0x37
		S	0x1E	s	0x38
		T	0x1F	t	0x39
		U	0x20	u	0x3A
		V	0x21	v	0x3B
		W	0x22	w	0x3C
		X	0x23	x	0x3D
		Y	0x24	y	0x3E
		Z	0x25	z	0x3F